

Systems that think like humans	Systems that think rationally
"The exciting new effort to make computers think . . . <i>machines with minds</i> , in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Chamiak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Systems that act like humans	Systems that act rationally
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole <i>et al.</i> , 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)
<b>Figure 1.1</b> Some definitions of artificial intelligence, organized into four categories.	

Historically, all four approaches to AI have been followed. As one might expect, a tension exists between approaches centered around humans and approaches centered around rationality.<sup>1</sup> A human-centered approach must be an empirical science, involving hypothesis and experimental confirmation. A rationalist approach involves a combination of mathematics and engineering. Each group has both disparaged and helped the other. Let us look at the four approaches in more detail.

### Acting humanly: The Turing Test approach

#### TURING TEST

The **Turing Test**, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence. Rather than proposing a long and perhaps controversial list of qualifications required for intelligence, he suggested a test based on indistinguishability from undeniably intelligent entities—human beings. The computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or not. Chapter 26 discusses the details of the test and whether a computer is really intelligent if it passes. For now, we note that programming a computer to pass the test provides plenty to work on. The computer would need to possess the following capabilities:

#### NATURAL LANGUAGE PROCESSING

◇ **natural language processing** to enable it to communicate successfully in English.

<sup>1</sup> We should point out that, by distinguishing between *human* and rational behavior, we are not suggesting that humans are necessarily "irrational" in the sense of "emotionally unstable" or "insane." One merely need note that we are not perfect: we are not all chess grandmasters, even those of us who know all the rules of chess; and, unfortunately, not everyone gets an A on the exam. Some systematic errors in human reasoning are cataloged by Kahneman *et al.* (1982).

KNOWLEDGE  
REPRESENTATION◇ **knowledge representation** to store what it knows or hears;AUTOMATED  
REASONING◇ **automated reasoning** to use the stored information to answer questions and to draw new conclusions;

MACHINE LEARNING

◇ **machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

TOTAL TURING TEST

Turing's test deliberately avoided direct physical interaction between the interrogator and the computer, because physical simulation of a person is unnecessary for intelligence. However, the so-called **total Turing Test** includes a video signal so that the interrogator can test the subject's perceptual abilities, as well as the opportunity for the interrogator to pass physical objects "through the hatch." To pass the total Turing Test, the computer will need

COMPUTER VISION

◇ **computer vision** to perceive objects, and

ROBOTICS

◇ **robotics** to manipulate objects and move about.

These six disciplines compose most of AI, and Turing deserves credit for designing a test that remains relevant 50 years later. Yet AI researchers have devoted little effort to passing the Turing test, believing that it is more important to study the underlying principles of intelligence than to duplicate an exemplar. The quest for "artificial flight" succeeded when the Wright brothers and others stopped imitating birds and learned about aerodynamics. Aeronautical engineering texts do not define the goal of their field as making "machines that fly so exactly like pigeons that they can fool even other pigeons."

### Thinking humanly: The cognitive modeling approach

COGNITIVE SCIENCE

If we are going to say that a given program thinks like a human, we must have some way of determining how humans think. We need to get *inside* the actual workings of human minds. There are two ways to do this: through introspection—trying to catch our own thoughts as they go by—and through psychological experiments. Once we have a sufficiently precise theory of the mind, it becomes possible to express the theory as a computer program. If the program's input/output and timing behaviors match corresponding human behaviors, that is evidence that some of the program's mechanisms could also be operating in humans. For example, Allen Newell and Herbert Simon, who developed GPS, the "General Problem Solver" (Newell and Simon, 1961), were not content to have their program solve problems correctly. They were more concerned with comparing the trace of its reasoning steps to traces of human subjects solving the same problems. The interdisciplinary field of **cognitive science** brings together computer models from AI and experimental techniques from psychology to try to construct precise and testable theories of the workings of the human mind.

Cognitive science is a fascinating field, worthy of an encyclopedia in itself (Wilson and Keil, 1999). We will not attempt to describe what is known of human cognition in this book. We will occasionally comment on similarities or differences between AI techniques and human cognition. Real cognitive science, however, is necessarily based on experimental investigation of actual humans or animals, and we assume that the reader has access only to a computer for experimentation.

In the early days of AI there was often confusion between the approaches: an author would argue that an algorithm performs well on a task and that it is *therefore* a good model

of human performance, or vice versa. Modern authors separate the two kinds of claims; this distinction has allowed both AI and cognitive science to develop more rapidly. The two fields continue to fertilize each other, especially in the areas of vision and natural language. Vision in particular has recently made advances via an integrated approach that considers neurophysiological evidence and computational models.

### Thinking rationally: The "laws of thought" approach

SYLLOGISMS

The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking," that is, irrefutable reasoning processes. His **syllogisms** provided patterns for argument structures that always yielded correct conclusions when given correct premises—for example, "Socrates is a man; all men are mortal; therefore, Socrates is mortal." These laws of thought were supposed to govern the operation of the mind; their study initiated the field called **logic**.

LOGIC

Logicians in the 19th century developed a precise notation for statements about all kinds of things in the world and about the relations among them. (Contrast this with ordinary arithmetic notation, which provides mainly for equality and inequality statements about numbers.) By 1965, programs existed that could, in principle, solve *any* solvable problem described in logical notation.<sup>2</sup> The so-called **logicist** tradition within artificial intelligence hopes to build on such programs to create intelligent systems.

LOGICIST

There are two main obstacles to this approach. First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain. Second, there is a big difference between being able to solve a problem "in principle" and doing so in practice. Even problems with just a few dozen facts can exhaust the computational resources of any computer unless it has some guidance as to which reasoning steps to try first. Although both of these obstacles apply to *any* attempt to build computational reasoning systems, they appeared first in the logicist tradition.

### Acting rationally: The rational agent approach

AGENT

An **agent** is just something that acts (*agent* comes from the Latin *agere*, to do). But computer agents are expected to have other attributes that distinguish them from mere "programs," such as operating under autonomous control, perceiving their environment, persisting over a prolonged time period, adapting to change, and being capable of taking on another's goals. A **rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.

RATIONAL AGENT

In the "laws of thought" approach to AI, the emphasis was on correct inferences. Making correct inferences is sometimes *part* of being a rational agent, because one way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals and then to act on that conclusion. On the other hand, correct inference is not all of rationality, because there are often situations where there is no provably correct thing to do, yet something must still be done. There are also ways of acting rationally that cannot be said to involve inference. For example, recoiling from a hot stove is a reflex action that is usually more successful than a slower action taken after careful deliberation.

<sup>2</sup> If there is no solution, the program might never stop looking for one.

All the skills needed for the Turing Test are there to allow rational actions. Thus, we need the ability to represent knowledge and reason with it because this enables us to reach good decisions in a wide variety of situations. We need to be able to generate comprehensible sentences in natural language because saying those sentences helps us get by in a complex society. We need learning not just for erudition, but because having a better idea of how the world works enables us to generate more effective strategies for dealing with it. We need visual perception not just because seeing is fun, but to get a better idea of what an action might achieve—for example, being able to see a tasty morsel helps one to move toward it.

For these reasons, the study of AI as rational-agent design has at least two advantages. First, it is more general than the "laws of thought" approach, because correct inference is just one of several possible mechanisms for achieving rationality. Second, it is more amenable to scientific development than are approaches based on human behavior or human thought because the standard of rationality is clearly defined and completely general. Human behavior, on the other hand, is well-adapted for one specific environment and is the product, in part, of a complicated and largely unknown evolutionary process that still is far from producing perfection. *This book will therefore concentrate on general principles of rational agents and on components for constructing them.* We will see that despite the apparent simplicity with which the problem can be stated, an enormous variety of issues come up when we try to solve it. Chapter 2 outlines some of these issues in more detail.

One important point to keep in mind: We will see before too long that achieving perfect rationality—always doing the right thing—is not feasible in complicated environments. The computational demands are just too high. For most of the book, however, we will adopt the working hypothesis that perfect rationality is a good starting point for analysis. It simplifies the problem and provides the appropriate setting for most of the foundational material in the field. Chapters 6 and 17 deal explicitly with the issue of limited rationality—acting appropriately when there is not enough time to do all the computations one might like.



LIMITED  
RATIONALITY

## 1.2 THE FOUNDATIONS OF ARTIFICIAL INTELLIGENCE

In this section, we provide a brief history of the disciplines that contributed ideas, viewpoints, and techniques to AI. Like any history, this one is forced to (concentrate on a small number of people, events, and ideas and to ignore others that also were important. We organize the history around a series of questions. We certainly would not wish to give the impression that these questions are the only ones the disciplines address or that the disciplines have all been working toward AI as their ultimate fruition.

### Philosophy (428 B.C.–present)

- Can formal rules be used to draw valid conclusions?
- How does the mental mind arise from a physical brain?
- Where does knowledge come from?
- How does knowledge lead to action?

Aristotle (384–322 B.C.) was the first to formulate a precise set of laws governing the rational part of the mind. He developed an informal system of syllogisms for proper reasoning, which in principle allowed one to generate conclusions mechanically, given initial premises. Much later, Ramon Lull (d. 1315) had the idea that useful reasoning could actually be carried out by a mechanical artifact. His "concept wheels" are on the cover of this book. Thomas Hobbes (1588–1679) proposed that reasoning was like numerical computation, that "we add and subtract in our silent thoughts." The automation of computation itself was already well under way; around 1500, Leonardo da Vinci (1452–1519) designed but did not build a mechanical calculator; recent reconstructions have shown the design to be functional. The first known calculating machine was constructed around 1623 by the German scientist Wilhelm Schickard (1592–1635), although the Pascaline, built in 1642 by Blaise Pascal (1623–1662), is more famous. Pascal wrote that "the arithmetical machine produces effects which appear nearer to thought than all the actions of animals." Gottfried Wilhelm Leibniz (1646–1716) built a mechanical device intended to carry out operations on concepts rather than numbers, but its scope was rather limited.

Now that we have the idea of a set of rules that can describe the formal, rational part of the mind, the next step is to consider the mind as a physical system. René Descartes (1596–1650) gave the first clear discussion of the distinction between mind and matter and of the problems that arise. One problem with a purely physical conception of the mind is that it seems to leave little room for free will: if the mind is governed entirely by physical laws, then it has no more free will than a rock "deciding" to fall toward the center of the earth. Although a strong advocate of the power of reasoning, Descartes was also a proponent of **dualism**. He held that there is a part of the human mind (or soul or spirit) that is outside of nature, exempt from physical laws. Animals, on the other hand, did not possess this dual quality; they could be treated as machines. An alternative to dualism is **materialism**, which holds that the brain's operation according to the laws of physics *constitutes* the mind. Free will is simply the way that the perception of available choices appears to the choice process.

Given a physical mind that manipulates knowledge, the next problem is to establish the source of knowledge. The **empiricism** movement, starting with Francis Bacon's (1561–1626) *Novum Organum*,<sup>3</sup> is characterized by a dictum of John Locke (1632–1704): "Nothing is in the understanding, which was not first in the senses." David Hume's (1711–1776) *A Treatise of Human Nature* (Hume, 1739) proposed what is now known as the principle of **induction**: that general rules are acquired by exposure to repeated associations between their elements. Building on the work of Ludwig Wittgenstein (1889–1951) and Bertrand Russell (1872–1970), the famous Vienna Circle, led by Rudolf Carnap (1891–1970), developed the doctrine of **logical positivism**. This doctrine holds that all knowledge can be characterized by logical theories connected, ultimately, to **observation sentences** that correspond to sensory inputs.<sup>4</sup> The **confirmation theory** of Carnap and Carl Hempel (1905–1997) attempted to understand how knowledge can be acquired from experience. Carnap's book *The Logical Structure of*

<sup>3</sup> An update of Aristotle's *Organon*, or instrument of thought.

<sup>4</sup> In this picture, all meaningful statements can be verified or falsified either by analyzing the meaning of the words or by carrying out experiments. Because this rules out most of metaphysics, as was the intention, logical positivism was unpopular in some circles.

DUALISM

MATERIALISM

EMPIRICISM

INDUCTION

LOGICAL POSITIVISM  
OBSERVATION  
SENTENCES  
CONFIRMATION  
THEORY

*the World* (1928) defined an explicit computational procedure for extracting knowledge from elementary experiences. It was probably the first theory of mind as a computational process.

The final element in the philosophical picture of the mind is the connection between knowledge and action. This question is vital to AI, because intelligence requires action as well as reasoning. Moreover, only by understanding how actions are justified can we understand how to build an agent whose actions are justifiable (or rational). Aristotle argued that actions are justified by a logical connection between goals and knowledge of the action's outcome (the last part of this extract also appears on the front cover of this book):

But how does it happen that thinking is sometimes accompanied by action and sometimes not, sometimes by motion, and sometimes not? It looks as if almost the same thing happens as in the case of reasoning and making inferences about unchanging objects. But in that case the end is a speculative proposition . . . whereas here the conclusion which results from the two premises is an action. . . . I need covering; a cloak is a covering. I need a cloak. What I need, I have to make; I need a cloak. I have to make a cloak. And the conclusion, the "I have to make a cloak" is an action. (Nussbaum, 1978, p. 40)

In the *Nicomachean Ethics* (Book III. 3, 1112b), Aristotle further elaborates on this topic, suggesting an algorithm:

We deliberate not about ends, but about means. For a doctor does not deliberate whether he shall heal, nor an orator whether he shall persuade, . . . They assume the end and consider how and by what means it is attained, and if it seems easily and best produced thereby; while if it is achieved by one means only they consider *how* it will be achieved by this and by what means *this* will be achieved, till they come to the first cause, . . . and what is last in the order of analysis seems to be first in the order of becoming. And if we come on an impossibility, we give up the search, e.g. if we need money and this cannot be got; but if a thing appears possible we try to do it.

Aristotle's algorithm was implemented 2300 years later by Newell and Simon in their GPS program. We would now call it a regression planning system. (See Chapter 11.)

Goal-based analysis is useful, but does not say what to do when several actions will achieve the goal, or when no action will achieve it completely. Antoine Arnauld (1612–1694) correctly described a quantitative formula for deciding what action to take in cases like this (see Chapter 16). John Stuart Mill's (1806–1873) book *Utilitarianism* (Mill, 1863) promoted the idea of rational decision criteria in all spheres of human activity. The more formal theory of decisions is discussed in the following section.

### Mathematics (c. 800–present)

- What are the formal rules to draw valid conclusions?
- What can be computed?
- How do we reason with uncertain information?

Philosophers staked out most of the important ideas of AI, but the leap to a formal science required a level of mathematical formalization in three fundamental areas: logic, computation, and probability.

The idea of formal logic can be traced back to the philosophers of ancient Greece (see Chapter 7), but its mathematical development really began with the work of George Boole

(1815–1864), who worked out the details of propositional, or Boolean, logic (Boole, 1847). In 1879, Gottlob Frege (1848–1925) extended Boole's logic to include objects and relations, creating the first-order logic that is used today as the most basic knowledge representation system.<sup>5</sup> Alfred Tarski (1902–1983) introduced a theory of reference that shows how to relate the objects in a logic to objects in the real world. The next step was to determine the limits of what could be done with logic and computation.

## ALGORITHM

The first nontrivial **algorithm** is thought to be Euclid's algorithm for computing greatest common denominators. The study of algorithms as objects in themselves goes back to al-Khowarazmi, a Persian mathematician of the 9th century, whose writings also introduced Arabic numerals and algebra to Europe. Boole and others discussed algorithms for logical deduction, and, by the late 19th century, efforts were under way to formalize general mathematical reasoning as logical deduction. In 1900, David Hilbert (1862–1943) presented a list of 23 problems that he correctly predicted would occupy mathematicians for the bulk of the century. The final problem asks whether there is an algorithm for deciding the truth of any logical proposition involving the natural numbers—the famous *Entscheidungsproblem*, or decision problem. Essentially, Hilbert was asking whether there were fundamental limits to the power of effective proof procedures. In 1930, Kurt Gödel (1906–1978) showed that there exists an effective procedure to prove any true statement in the first-order logic of Frege and Russell, but that first-order logic could not capture the principle of mathematical induction needed to characterize the natural numbers. In 1931, he showed that real limits do exist. His **incompleteness theorem** showed that in any language expressive enough to describe the properties of the natural numbers, there are true statements that are undecidable in the sense that their truth cannot be established by any algorithm.

INCOMPLETENESS  
THEOREM

This fundamental result can also be interpreted as showing that there are some functions on the integers that cannot be represented by an algorithm—that is, they cannot be computed. This motivated Alan Turing (1912–1954) to try to characterize exactly which functions are capable of being computed. This notion is actually slightly problematic, because the notion of a computation or effective procedure really cannot be given a formal definition. However, the Church–Turing thesis, which states that the Turing machine (Turing, 1936) is capable of computing any computable function, is generally accepted as providing a sufficient definition. Turing also showed that there were some functions that no Turing machine can compute. For example, no machine can tell in general whether a given program will return an answer on a given input or run forever.

## INTRACTABILITY

Although undecidability and noncomputability are important to an understanding of computation, the notion of **intractability** has had a much greater impact. Roughly speaking, a problem is called intractable if the time required to solve instances of the problem grows exponentially with the size of the instances. The distinction between polynomial and exponential growth in complexity was first emphasized in the mid-1960s (Cobham, 1964; Edmonds, 1965). It is important because exponential growth means that even moderately large instances cannot be solved in any reasonable time. Therefore, one should strive to divide

<sup>5</sup> Frege's proposed notation for first-order logic never became popular, for reasons that are apparent immediately from the example on the front cover.

the overall problem of generating intelligent behavior into tractable subproblems rather than intractable ones.

## NP-COMPLETENESS

How can one recognize an intractable problem? The theory of **NP-completeness**, pioneered by Steven Cook (1971) and Richard Karp (1972), provides a method. Cook and Karp showed the existence of large classes of canonical combinatorial search and reasoning problems that are NP-complete. Any problem class to which the class of NP-complete problems can be reduced is likely to be intractable. (Although it has not been proved that NP-complete problems are necessarily intractable, most theoreticians believe it.) These results contrast with the optimism with which the popular press greeted the first computers—"Electronic Super-Brains" that were "Faster than Einstein!" Despite the increasing speed of computers, careful use of resources will characterize intelligent systems. Put crudely, the world is an *extremely* large problem instance! In recent years, AI has helped explain why some instances of NP-complete problems are hard, yet others are easy (Cheeseman *et al.*, 1991).

## PROBABILITY

Besides logic and computation, the third great contribution of mathematics to AI is the theory of **probability**. The Italian Gerolamo Cardano (1501–1576) first framed the idea of probability, describing it in terms of the possible outcomes of gambling events. Probability quickly became an invaluable part of all the quantitative sciences, helping to deal with uncertain measurements and incomplete theories. Pierre Fermat (1601–1665), Blaise Pascal (1623–1662), James Bernoulli (1654–1705), Pierre Laplace (1749–1827), and others advanced the theory and introduced new statistical methods. Thomas Bayes (1702–1761) proposed a rule for updating probabilities in the light of new evidence. Bayes' rule and the resulting field called Bayesian analysis form the basis of most modern approaches to uncertain reasoning in AI systems.

### Economics (1776–present)

- How should we make decisions so as to maximize payoff?
- How should we do this when others may not go along?
  - a How should we do this when the payoff may be far in the future?

The science of economics got its start in 1776, when Scottish philosopher Adam Smith (1723–1790) published *An Inquiry into the Nature and Causes of the Wealth of Nations*. While the ancient Greeks and others had made contributions to economic thought, Smith was the first to treat it as a science, using the idea that economies can be thought of as consisting of individual agents maximizing their own economic well-being. Most people think of economics as being about money, but economists will say that they are really studying how people make choices that lead to preferred outcomes. The mathematical treatment of "preferred outcomes" or **utility** was first formalized by Léon Walras (pronounced "Valrasse") (1834–1910) and was improved by Frank Ramsey (1931) and later by John von Neumann and Oskar Morgenstern in their book *The Theory of Games and Economic Behavior* (1944).

## DECISION THEORY

**Decision theory**, which combines probability theory with utility theory, provides a formal and complete framework for decisions (economic or otherwise) made under uncertainty—that is, in cases where probabilistic descriptions appropriately capture the decision-maker's environment. This is suitable for "large" economies where each agent need pay no attention



GAMETHEORY

to the actions of other agents as individuals. For "small" economies, the situation is much more like a game: the actions of one player can significantly affect the utility of another (either positively or negatively). Von Neumann and Morgenstern's development of game theory (see also Luce and Raiffa, 1957) included the surprising result that, for some games, a rational agent should act in a random fashion, or at least in a way that appears random to the adversaries.

OPERATIONS  
RESEARCH

For the most part, economists did not address the third question listed above, namely, how to make rational decisions when payoffs from actions are not immediate but instead result from several actions taken in *sequence*. This topic was pursued in the field of operations research, which emerged in World War II from efforts in Britain to optimize radar installations, and later found civilian applications in complex management decisions. The work of Richard Bellman (1957) formalized a class of sequential decision problems called Markov decision processes, which we study in Chapters 17 and 21.

SATISFICING

Work in economics and operations research has contributed much to our notion of rational agents, yet for many years AI research developed along entirely separate paths. One reason was the apparent complexity of making rational decisions. Herbert Simon (1916–2001), the pioneering AI researcher, won the Nobel prize in economics in 1978 for his early work showing that models based on satisficing—making decisions that are "good enough," rather than laboriously calculating an optimal decision—gave a better description of actual human behavior (Simon, 1947). In the 1990s, there has been a resurgence of interest in decision-theoretic techniques for agent systems (Wellman, 1995).

## Neuroscience (1861–present)

- How do brains process information?

NEUROSCIENCE

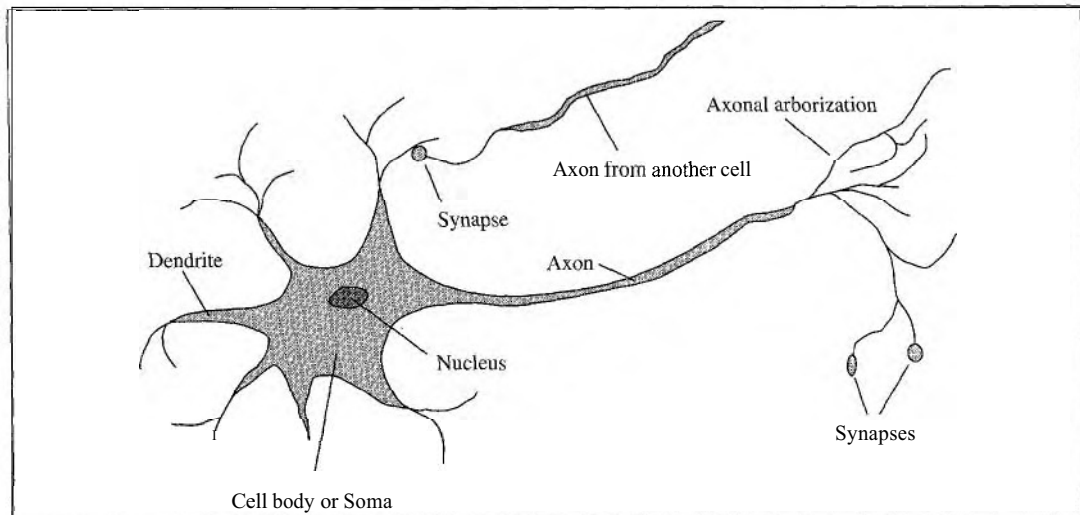
Neuroscience is the study of the nervous system, particularly the brain. The exact way in which the brain enables thought is one of the great mysteries of science. It has been appreciated for thousands of years that the brain is somehow involved in thought, because of the evidence that strong blows to the head can lead to mental incapacitation. It has also long been known that human brains are somehow different; in about 335 B.C. Aristotle wrote, "Of all the animals, man has the largest brain in proportion to his size."<sup>6</sup> Still, it was not until the middle of the 18th century that the brain was widely recognized as the seat of consciousness. Before then, candidate locations included the heart, the spleen, and the pineal gland.

NEURONS

Paul Broca's (1824–1880) study of aphasia (speech deficit) in brain-damaged patients in 1861 reinvigorated the field and persuaded the medical establishment of the existence of localized areas of the brain responsible for specific cognitive functions. In particular, he showed that speech production was localized to a portion of the left hemisphere now called Broca's area.<sup>7</sup> By that time, it was known that the brain consisted of nerve cells or neurons, but it was not until 1873 that Camillo Golgi (1843–1926) developed a staining technique allowing the observation of individual neurons in the brain (see Figure 1.2). This technique

<sup>6</sup> Since then, it has been discovered that some species of dolphins and whales have relatively larger brains. The large size of human brains is now thought to be enabled in part by recent improvements in its cooling system.

<sup>7</sup> Many cite Alexander Hood (1824) as a possible prior source.



**Figure 1.2** The parts of a nerve cell or neuron. Each neuron consists of a cell body, or soma, that contains a cell nucleus. Branching out from the cell body are a number of fibers called dendrites and a single long fiber called the axon. The axon stretches out for a long distance, much longer than the scale in this diagram indicates. Typically they are 1 cm long (100 times the diameter of the cell body), but can reach up to 1 meter. A neuron makes connections with 10 to 100,000 other neurons at junctions called synapses. Signals are propagated from neuron to neuron by a complicated electrochemical reaction. The signals control brain activity in the short term, and also enable long-term changes in the position and connectivity of neurons. These mechanisms are thought to form the basis for learning in the brain. Most information processing goes on in the cerebral cortex, the outer layer of the brain. The basic organizational unit appears to be a column of tissue about 0.5 mm in diameter, extending the full depth of the cortex, which is about 4 mm in humans. A column contains about 20,000 neurons.

was used by Santiago Ramon y Cajal (1852–1934) in his pioneering studies of the brain's neuronal structures.'

We now have some data on the mapping between areas of the brain and the parts of the body that they control or from which they receive sensory input. Such mappings are able to change radically over the course of a few weeks, and some animals seem to have multiple maps. Moreover, we do not fully understand how other areas can take over functions when one area is damaged. There is almost no theory on how an individual memory is stored.

The measurement of intact brain activity began in 1929 with the invention by Hans Berger of the electroencephalograph (EEG). The recent development of functional magnetic resonance imaging (fMRI) (Ogawa *et al.*, 1990) is giving neuroscientists unprecedentedly detailed images of brain activity, enabling measurements that correspond in interesting ways to ongoing cognitive processes. These are augmented by advances in single-cell recording of

<sup>8</sup> Golgi persisted in his belief that the brain's functions were carried out primarily in a continuous medium in which neurons were embedded, whereas Cajal propounded the "neuronal doctrine." The two shared the Nobel prize in 1906 but gave rather antagonistic acceptance speeches.

	Computer	Human Brain
Computational units	1 CPU, $10^8$ gates	$10^{11}$ neurons
Storage units	$10^{10}$ bits RAM $10^{11}$ bits disk	$10^{11}$ neurons $10^{14}$ synapses
Cycle time	$10^{-9}$ sec	$10^{-3}$ sec
Bandwidth	$10^{10}$ bits/sec	$10^{14}$ bits/sec
Memory updates/sec	$10^9$	$10^{14}$

**Figure 1.3** A crude comparison of the raw computational resources available to computers (circa 2003) and brains. The computer's numbers have all increased by at least a factor of 10 since the first edition of this book, and are expected to do so again this decade. The brain's numbers have not changed in the last 10,000 years.

neuron activity. Despite these advances, we are still a long way from understanding how any of these cognitive processes actually work.

The truly amazing conclusion is that *a collection of simple cells can lead to thought, action, and consciousness* or, in other words, that *brains cause minds* (Searle, 1992). The only real alternative theory is mysticism: that there is some mystical realm in which minds operate that is beyond physical science.

Brains and digital computers perform quite different tasks and have different properties. Figure 1.3 shows that there are 1000 times more neurons in the typical human brain than there are gates in the CPU of a typical high-end computer. Moore's Law<sup>9</sup> predicts that the CPU's gate count will equal the brain's neuron count around 2020. Of course, little can be inferred from such predictions; moreover, the difference in storage capacity is minor compared to the difference in switching speed and in parallelism. Computer chips can execute an instruction in a nanosecond, whereas neurons are millions of times slower. Brains more than make up for this, however, because all the neurons and synapses are active simultaneously, whereas most current computers have only one or at most a few CPUs. Thus, *even though a computer is a million times faster in raw switching speed, the brain ends up being 100,000 times faster at what it does.*

## Psychology (1879–present)

- How do humans and animals think and act?

The origins of scientific psychology are usually traced to the work of the German physicist Hermann von Helmholtz (1821–1894) and his student Wilhelm Wundt (1832–1920). Helmholtz applied the scientific method to the study of human vision, and his *Handbook of Physiological Optics* is even now described as "the single most important treatise on the physics and physiology of human vision" (Nalwa, 1993, p.15). In 1879, Wundt opened the first laboratory of experimental psychology at the University of Leipzig. Wundt insisted on carefully controlled experiments in which his workers would perform a perceptual or associa-

<sup>9</sup> Moore's Law says that the number of transistors per square inch doubles every 1 to 1.5 years. Human brain capacity doubles roughly every 2 to 4 million years.

## BEHAVIORISM

tive task while introspecting on their thought processes. The careful controls went a long way toward making psychology a science, but the subjective nature of the data made it unlikely that an experimenter would ever disconfirm his or her own theories. Biologists studying animal behavior, on the other hand, lacked introspective data and developed an objective methodology, as described by H. S. Jennings (1906) in his influential work *Behavior of the Lower Organisms*. Applying this viewpoint to humans, the **behaviorism** movement, led by John Watson (1878–1958), rejected *any* theory involving mental processes on the grounds that introspection could not provide reliable evidence. Behaviorists insisted on studying only objective measures of the percepts (or *stimulus*) given to an animal and its resulting actions (or *response*). Mental constructs such as knowledge, beliefs, goals, and reasoning steps were dismissed as unscientific "folk psychology." Behaviorism discovered a lot about rats and pigeons, but had less success at understanding humans. Nevertheless, it exerted a strong hold on psychology (especially in the United States) from about 1920 to 1960.

## COGNITIVE PSYCHOLOGY

The view of the brain as an information-processing device, which is a principal characteristic of **cognitive psychology**, can be traced back at least to the works of William James<sup>10</sup> (1842–1910). Helmholtz also insisted that perception involved a form of unconscious logical inference. The cognitive viewpoint was largely eclipsed by behaviorism in the United States, but at Cambridge's Applied Psychology Unit, directed by Frederic Bartlett (1886–1969), cognitive modeling was able to flourish. *The Nature of Explanation*, by Bartlett's student and successor Kenneth Craik (1943), forcefully reestablished the legitimacy of such "mental" terms as beliefs and goals, arguing that they are just as scientific as, say, using pressure and temperature to talk about gases, despite their being made of molecules that have neither. Craik specified the three key steps of a knowledge-based agent: (1) the stimulus must be translated into an internal representation, (2) the representation is manipulated by cognitive processes to derive new internal representations, and (3) these are in turn retranslated back into action. He clearly explained why this was a good design for an agent:

If the organism carries a "small-scale model" of external reality and of its own possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it. (Craik, 1943)

## COGNITIVE SCIENCE

After Craik's death in a bicycle accident in 1945, his work was continued by Donald Broadbent, whose book *Perception and Communication* (1958) included some of the first information-processing models of psychological phenomena. Meanwhile, in the United States, the development of computer modeling led to the creation of the field of **cognitive science**. The field can be said to have started at a workshop in September 1956 at MIT. (We shall see that this is just two months after the conference at which AI itself was "born.") At the workshop, George Miller presented *The Magic Number Seven*, Noam Chomsky presented *Three Models of Language*, and Allen Newell and Herbert Simon presented *The Logic Theory Machine*. These three influential papers showed how computer models could be used to

<sup>10</sup> William James was the brother of novelist Henry James. It is said that Henry wrote fiction as if it were psychology and William wrote psychology as if it were fiction.

address the psychology of memory, language, and logical thinking, respectively. It is now a common view among psychologists that "a cognitive theory should be like a computer program" (Anderson, 1980), that is, it should describe a detailed information-processing mechanism whereby some cognitive function might be implemented.

### Computer engineering (1940–present)

- How can we build an efficient computer?

For artificial intelligence to succeed, we need two things: intelligence and an artifact. The computer has been the artifact of choice. The modern digital electronic computer was invented independently and almost simultaneously by scientists in three countries embattled in World War II. The first *operational* computer was the electromechanical Heath Robinson,<sup>11</sup> built in 1940 by Alan Turing's team for a single purpose: deciphering German messages. In 1943, the same group developed the Colossus, a powerful general-purpose machine based on vacuum tubes.<sup>12</sup> The first operational *programmable* computer was the Z-3, the invention of Konrad Zuse in Germany in 1941. Zuse also invented floating-point numbers and the first high-level programming language, Plankalkül. The first *electronic* computer, the ABC, was assembled by John Atanasoff and his student Clifford Berry between 1940 and 1942 at Iowa State University. Atanasoff's research received little support or recognition; it was the ENIAC, developed as part of a secret military project at the University of Pennsylvania by a team including John Mauchly and John Eckert, that proved to be the most influential forerunner of modern computers.

In the half-century since then, each generation of computer hardware has brought an increase in speed and capacity and a decrease in price. Performance doubles every 18 months or so, with a decade or two to go at this rate of increase. After that, we will need molecular engineering or some other new technology.

Of course, there were calculating devices before the electronic computer. The earliest automated machines, dating from the 17th century, were discussed on page 6. The first *programmable* machine was a loom devised in 1805 by Joseph Marie Jacquard (1752–1834) that used punched cards to store instructions for the pattern to be woven. In the mid-19th century, Charles Babbage (1792–1871) designed two machines, neither of which he completed. The "Difference Engine," which appears on the cover of this book, was intended to compute mathematical tables for engineering and scientific projects. It was finally built and shown to work in 1991 at the Science Museum in London (Swade, 1993). Babbage's "Analytical Engine" was far more ambitious: it included addressable memory, stored programs, and conditional jumps and was the first artifact capable of universal computation. Babbage's colleague Ada Lovelace, daughter of the poet Lord Byron, was perhaps the world's first programmer. (The programming language Ada is named after her.) She wrote programs for the unfinished Analytical Engine and even speculated that the machine could play chess or compose music.

<sup>11</sup> Heath Robinson was a cartoonist famous for his depictions of whimsical and absurdly complicated contraptions for everyday tasks such as buttering toast.

<sup>12</sup> In the postwar period, Turing wanted to use these computers for AI research—for example, one of the first chess programs (Turing *et al.*, 1953). His efforts were blocked by the British government.

AI also owes a debt to the software side of computer science, which has supplied the operating systems, programming languages, and tools needed to write modern programs (and papers about them). But this is one area where the debt has been repaid: work in AI has pioneered many ideas that have made their way back to mainstream computer science, including time sharing, interactive interpreters, personal computers with windows and mice, rapid development environments, the linked list data type, automatic storage management, and key concepts of symbolic, functional, dynamic, and object-oriented programming.

### Control theory and Cybernetics (1948–present)

- How can artifacts operate under their own control?

Ktesibios of Alexandria (c. 250 B.C.) built the first self-controlling machine: a water clock with a regulator that kept the flow of water running through it at a constant, predictable pace. This invention changed the definition of what an artifact could do. Previously, only living things could modify their behavior in response to changes in the environment. Other examples of self-regulating feedback control systems include the steam engine governor, created by James Watt (1736–1819), and the thermostat, invented by Cornelis Drebbel (1572–1633), who also invented the submarine. The mathematical theory of stable feedback systems was developed in the 19th century.

CONTROL THEORY

The central figure in the creation of what is now called **control theory** was Norbert Wiener (1894–1964). Wiener was a brilliant mathematician who worked with Bertrand Russell, among others, before developing an interest in biological and mechanical control systems and their connection to cognition. Like Craik (who also used control systems as psychological models), Wiener and his colleagues Arturo Rosenblueth and Julian Bigelow challenged the behaviorist orthodoxy (Rosenblueth *et al.*, 1943). They viewed purposive behavior as arising from a regulatory mechanism trying to minimize "error"—the difference between current state and goal state. In the late 1940s, Wiener, along with Warren McCulloch, Walter Pitts, and John von Neumann, organized a series of conferences that explored the new mathematical and computational models of cognition and influenced many other researchers in the behavioral sciences. Wiener's book *Cybernetics* (1948) became a bestseller and avoked the public to the possibility of artificially intelligent machines.

CYBERNETICS

OBJECTIVE  
FUNCTION

Modern control theory, especially the branch known as stochastic optimal control, has as its goal the design of systems that maximize an **objective function** over time. This roughly matches our view of AI designing systems that behave optimally. Why, then, are AI and control theory two different fields, especially given the close connections among their founders? The answer lies in the close coupling between the mathematical techniques that were familiar to the participants and the corresponding sets of problems that were encompassed in each world view. Calculus and matrix algebra, the tools of control theory, lend themselves to systems that are describable by fixed sets of continuous variables; furthermore, exact analysis is typically feasible only for linear systems. AI was founded in part as a way to escape from the limitations of the mathematics of control theory in the 1950s. The tools of logical inference and computation allowed AI researchers to consider some problems such as language, vision, and planning, that fell completely outside the control theorist's purview.

### Linguistics (1957–present)

- How does language relate to thought?

In 1957, B. F. Skinner published *Verbal Behavior*. This was a comprehensive, detailed account of the behaviorist approach to language learning, written by the foremost expert in the field. But curiously, a review of the book became as well known as the book itself, and served to almost kill off interest in behaviorism. The author of the review was Noam Chomsky, who had just published a book on his own theory, *Syntactic Structures*. Chomsky showed how the behaviorist theory did not address the notion of creativity in language—it did not explain how a child could understand and make up sentences that he or she had never heard before. Chomsky's theory—based on syntactic models going back to the Indian linguist Panini (c. 350 B.C.)—could explain this, and unlike previous theories, it was formal enough that it could in principle be programmed.

COMPUTATIONAL  
LINGUISTICS

Modern linguistics and AI, then, were "born" at about the same time, and grew up together, intersecting in a hybrid field called **computational linguistics** or **natural language processing**. The problem of understanding language soon turned out to be considerably more complex than it seemed in 1957. Understanding language requires an understanding of the subject matter and context, not just an understanding of the structure of sentences. This might seem obvious, but it was not widely appreciated until the 1960s. Much of the early work in **knowledge representation** (the study of how to put knowledge into a form that a computer can reason with) was tied to language and informed by research in Linguistics, which was connected in turn to decades of work on the philosophical analysis of language.

## 1.3 THE HISTORY OF ARTIFICIAL INTELLIGENCE

With the background material behind us, we are ready to cover the development of AI itself.

### The gestation of artificial intelligence (1943–1955)

The first work that is now generally recognized as AI was done by Warren McCulloch and Walter Pitts (1943). They drew on three sources: knowledge of the basic physiology and function of neurons in the brain; a formal analysis of propositional logic due to Russell and Whitehead; and Turing's theory of computation. They proposed a model of artificial neurons in which each neuron is characterized as being "on" or "off," with a switch to "on" occurring in response to stimulation by a sufficient number of neighboring neurons. The state of a neuron was conceived of as "factually equivalent to a proposition which proposed its adequate stimulus." They showed, for example, that any computable function could be computed by some network of connected neurons, and that all the logical connectives (and, or, not, etc.) could be implemented by simple net structures. McCulloch and Pitts also suggested that suitably defined networks could learn. Donald Hebb (1949) demonstrated a simple updating rule for modifying the connection strengths between neurons. His rule, now called **Hebbian learning**, remains an influential model to this day.

Two undergraduate students at Harvard, Marvin Minsky and Dean Edmonds, built the first neural network computer in 1950. The SNARC, as it was called, used 3000 vacuum tubes and a surplus automatic pilot mechanism from a B-24 bomber to simulate a network of 40 neurons. Later, at Princeton, Minsky studied universal computation in neural networks. His Ph.D. committee was skeptical about whether this kind of work should be considered mathematics, but von Neumann reportedly said, "If it isn't now, it will be someday." Minsky was later to prove influential theorems showing the limitations of neural network research.

There were a number of early examples of work that can be characterized as AI, but it was Alan Turing who first articulated a complete vision of AI in his 1950 article "Computing Machinery and Intelligence." Therein, he introduced the Turing test, machine learning, genetic algorithms, and reinforcement learning.

### The birth of artificial intelligence (1956)

Princeton was home to another influential figure in AI, John McCarthy. After graduation, McCarthy moved to Dartmouth College, which was to become the official birthplace of the field. McCarthy convinced Minsky, Claude Shannon, and Nathaniel Rochester to help him bring together U.S. researchers interested in automata theory, neural nets, and the study of intelligence. They organized a two-month workshop at Dartmouth in the summer of 1956. There were 10 attendees in all, including Trenchard More from Princeton, Arthur Samuel from IBM, and Ray Solomonoff and Oliver Selfridge from MIT.

Two researchers from Carnegie Tech,<sup>13</sup> Allen Newell and Herbert Simon, rather stole the show. Although the others had ideas and in some cases programs for particular applications such as checkers, Newell and Simon already had a reasoning program, the Logic Theorist (LT), about which Simon claimed, "We have invented a computer program capable of thinking non-numerically, and thereby solved the venerable mind-body problem."<sup>14</sup> Soon after the workshop, the program was able to prove most of the theorems in Chapter 2 of Russell and Whitehead's *Principia Mathematica*. Russell was reportedly delighted when Simon showed him that the program had come up with a proof for one theorem that was shorter than the one in *Principia*. The editors of the *Journal of Symbolic Logic* were less impressed; they rejected a paper coauthored by Newell, Simon, and Logic Theorist.

The Dartmouth workshop did not lead to any new breakthroughs, but it did introduce all the major figures to each other. For the next 20 years, the field would be dominated by these people and their students and colleagues at MIT, CMU, Stanford, and IBM. Perhaps the longest-lasting thing to come out of the workshop was an agreement to adopt McCarthy's new name for the field: **artificial intelligence**. Perhaps "computational rationality" would have been better, but "AI" has stuck.

Looking at the proposal for the Dartmouth workshop (McCarthy *et al.*, 1955), we can see why it was necessary for AI to become a separate field. Why couldn't all the work done

<sup>13</sup> Now Carnegie Mellon University (CMU).

<sup>14</sup> Newell and Simon also invented a list-processing language, IPL, to write LT. They had no compiler, and translated it into machine code by hand. To avoid errors, they worked in parallel, calling out binary numbers to each other as they wrote each instruction to make sure they agreed.



in AI have taken place under the name of control theory, or operations research, or decision theory, which, after all, have objectives similar to those of AI? Or why isn't AI a branch of mathematics? The first answer is that AI from the start embraced the idea of duplicating human faculties like creativity, self-improvement, and language use. None of the other fields were addressing these issues. The second answer is methodology. AI is the only one of these fields that is clearly a branch of computer science (although operations research does share an emphasis on computer simulations), and AI is the only field to attempt to build machines that will function autonomously in complex, changing environments.

### Early enthusiasm, great expectations (1952–1969)

The early years of AI were full of successes—in a limited way. Given the primitive computers and programming tools of the time, and the fact that only a few years earlier computers were seen as things that could do *arithmetic* and no more, it was astonishing whenever a computer did anything remotely clever. The intellectual establishment, by and large, preferred to believe that "a machine can never do X." (See Chapter 26 for a long list of X's gathered by Turing.) AI researchers naturally responded by demonstrating one X after another. John McCarthy referred to this period as the "Look, Ma, no hands!" era.

Newell and Simon's early success was followed up with the General Problem Solver, or GPS. Unlike Logic Theorist, this program was designed from the start to imitate human problem-solving protocols. Within the limited class of puzzles it could handle, it turned out that the order in which the program considered subgoals and possible actions was similar to that in which humans approached the same problems. Thus, GPS was probably the first program to embody the "thinking humanly" approach. The success of GPS and subsequent programs as models of cognition led Newell and Simon (1976) to formulate the famous **physical symbol system** hypothesis, which states that "a physical symbol system has the necessary and sufficient means for general intelligent action." What they meant is that any system (human or machine) exhibiting intelligence must operate by manipulating data structures composed of symbols. We will see later that this hypothesis has been challenged from many directions.

At IBM, Nathaniel Rochester and his colleagues produced some of the first AI programs. Herbert Gelernter (1959) constructed the Geometry Theorem Prover, which was able to prove theorems that many students of mathematics would find quite tricky. Starting in 1952, Arthur Samuel wrote a series of programs for checkers (draughts) that eventually learned to play at a strong amateur level. Along the way, he disproved the idea that computers can do only what they are told to: his program quickly learned to play a better game than its creator. The program was demonstrated on television in February 1956, creating a very strong impression. Like Turing, Samuel had trouble finding computer time. Working at night, he used machines that were still on the testing floor at IBM's manufacturing plant. Chapter 6 covers game playing, and Chapter 21 describes and expands on the learning techniques used by Samuel.

John McCarthy moved from Dartmouth to MIT and there made three crucial contributions in one historic year: 1958. In MIT AI Lab Memo No. 1, McCarthy defined the high-level language **Lisp**, which was to become the dominant AI programming language. Lisp is the

second-oldest major high-level language in current use, one year younger than FORTRAN. With Lisp, McCarthy had the tool he needed, but access to scarce and expensive computing resources was also a serious problem. In response, he and others at MIT invented time sharing. Also in 1958, McCarthy published a paper entitled *Programs with Common Sense*, in which he described the Advice Taker, a hypothetical program that can be seen as the first complete AI system. Like the Logic Theorist and Geometry Theorem Prover, McCarthy's program was designed to use knowledge to search for solutions to problems. But unlike the others, it was to embody general knowledge of the world. For example, he showed how some simple axioms would enable the program to generate a plan to drive to the airport to catch a plane. The program was also designed so that it could accept new axioms in the normal course of operation, thereby allowing it to achieve competence in new areas *without being reprogrammed*. The Advice Taker thus embodied the central principles of knowledge representation and reasoning: that it is useful to have a formal, explicit representation of the world and of the way an agent's actions affect the world and to be able to manipulate these representations with deductive processes. It is remarkable how much of the 1958 paper remains relevant even today.

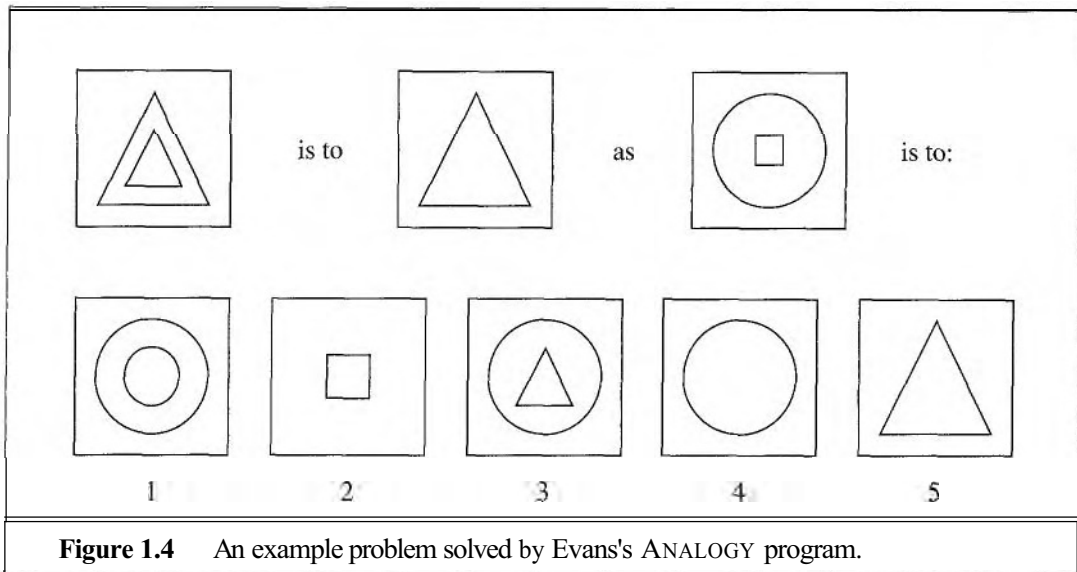
1958 also marked the year that Marvin Minsky moved to MIT. His initial collaboration with McCarthy did not last, however. McCarthy stressed representation and reasoning in formal logic, whereas Minsky was more interested in getting programs to work and eventually developed an anti-logical outlook. In 1963, McCarthy started the AI lab at Stanford. His plan to use logic to build the ultimate Advice Taker was advanced by J. A. Robinson's discovery of the resolution method (a complete theorem-proving algorithm for first-order logic; see Chapter 9). Work at Stanford emphasized general-purpose methods for logical reasoning. Applications of logic included Cordell Green's question-answering and planning systems (Green, 1969b) and the Shakey robotics project at the new Stanford Research Institute (SRI). The latter project, discussed further in Chapter 25, was the first to demonstrate the complete integration of logical reasoning and physical activity.

## MICROWORLDS

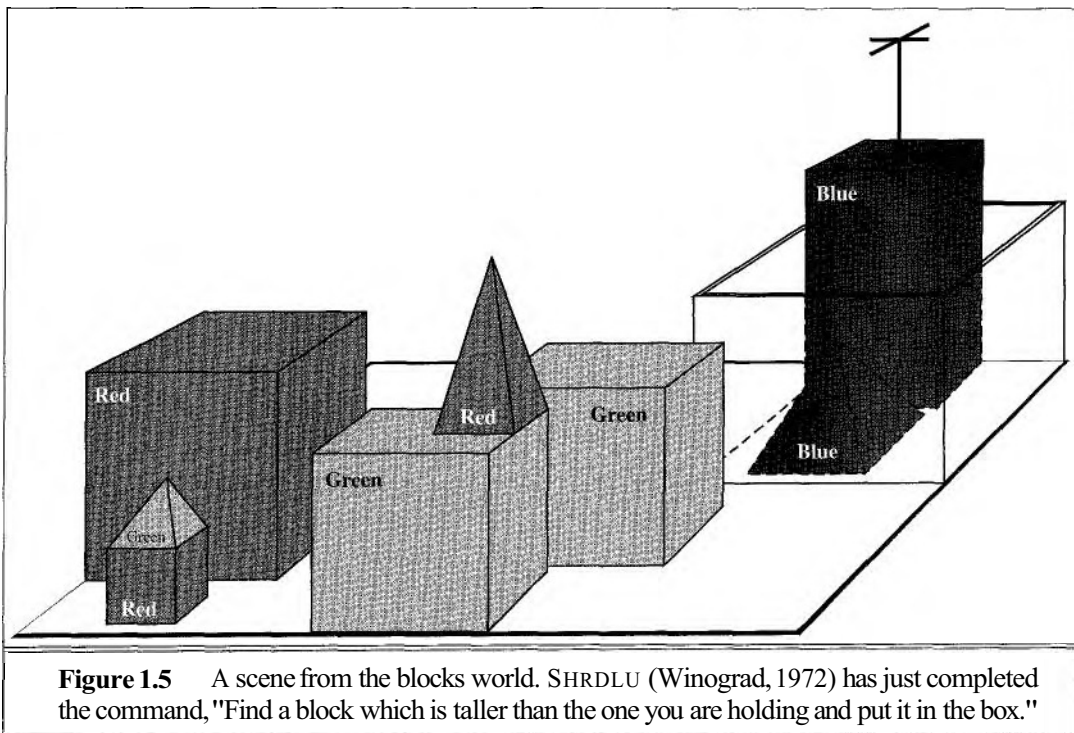
Minsky supervised a series of students who chose limited problems that appeared to require intelligence to solve. These limited domains became known as **microworlds**. James Slagle's SAINT program (1963a) was able to solve closed-form calculus integration problems typical of first-year college courses. Tom Evans's ANALOGY program (1968) solved geometric analogy problems that appear in IQ tests, such as the one in Figure 1.4. Daniel Bobrow's STUDENT program (1967) solved algebra story problems, such as the following:

If the number of customers Tom gets is twice the square of 20 percent of the number of advertisements he runs, and the number of advertisements he runs is 45, what is the number of customers Tom gets?

The most famous microworld was the blocks world, which consists of a set of solid blocks placed on a tabletop (or more often, a simulation of a tabletop), as shown in Figure 1.5. A typical task in this world is to rearrange the blocks in a certain way, using a robot hand that can pick up one block at a time. The blocks world was home to the vision project of David Huffman (1971), the vision and constraint-propagation work of David Waltz (1975), the learning theory of Patrick Winston (1970), the natural language understanding program



**Figure 1.4** An example problem solved by Evans's ANALOGY program.



**Figure 1.5** A scene from the blocks world. SHRDLU (Winograd, 1972) has just completed the command, "Find a block which is taller than the one you are holding and put it in the box."

of Terry Winograd (1972), and the planner of Scott Fahlman (1974).

Early work building on the neural networks of McCulloch and Pitts also flourished. The work of Winograd and Cowan (1963) showed how a large number of elements could collectively represent an individual concept, with a corresponding increase in robustness and parallelism. Hebb's learning methods were enhanced by Bernie Widrow (Widrow and Hoff,

1960; Widrow, 1962), who called his networks adalines, and by Frank Rosenblatt (1962) with his perceptrons. Rosenblatt proved the perceptron **convergence** theorem, showing that his learning algorithm could adjust the connection strengths of a perceptron to match any input data, provided such a match existed. These topics are covered in Chapter 20.

### A dose of reality (1966–1973)

From the beginning, AI researchers were not shy about making predictions of their coming successes. The following statement by Herbert Simon in 1957 is often quoted:

It is not my aim to surprise or shock you—but the simplest way I can summarize is to say that there are now in the world machines that think, that learn and that create. Moreover, their ability to do these things is going to increase rapidly until—in a visible future—the range of problems they can handle will be coextensive with the range to which the human mind has been applied.

Terms such as "visible future" can be interpreted in various ways, but Simon also made a more concrete prediction: that within 10 years a computer would be chess champion, and a significant mathematical theorem would be proved by machine. These predictions came true (or approximately true) within 40 years rather than 10. Simon's over-confidence was due to the promising performance of early AI systems on simple examples. In almost all cases, however, these early systems turned out to fail miserably when tried out on wider selections of problems and on more difficult problems.

The first kind of difficulty arose because most early programs contained little or no knowledge of their subject matter; they succeeded by means of simple syntactic manipulations. A typical story occurred in early machine translation efforts, which were generously funded by the U.S. National Research Council in an attempt to speed up the translation of Russian scientific papers in the wake of the Sputnik launch in 1957. It was thought initially that simple syntactic transformations based on the grammars of Russian and English, and word replacement using an electronic dictionary, would suffice to preserve the exact meanings of sentences. The fact is that translation requires general knowledge of the subject matter in order to resolve ambiguity and establish the content of the sentence. The famous re-translation of "the spirit is willing but the flesh is weak" as "the vodka is good but the meat is rotten" illustrates the difficulties encountered. In 1966, a report by an advisory committee found that "there has been no machine translation of general scientific text, and none is in immediate prospect." All U.S. government funding for academic translation projects was canceled. Today, machine translation is an imperfect but widely used tool for technical, commercial, government, and Internet documents.

The second kind of difficulty was the intractability of many of the problems that AI was attempting to solve. Most of the early AI programs solved problems by trying out different combinations of steps until the solution was found. This strategy worked initially because microworlds contained very few objects and hence very few possible actions and very short solution sequences. Before the theory of computational complexity was developed, it was widely thought that "scaling up" to larger problems was simply a matter of faster hardware and larger memories. The optimism that accompanied the development of resolution theorem



## MACHINE EVOLUTION

proving, for example, was soon dampened when researchers failed to prove theorems involving more than a few dozen facts. *The fact that a program can find a solution in principle does not mean that the program contains any of the mechanisms needed to find it in practice.*

The illusion of unlimited computational power was not confined to problem-solving programs. Early experiments in machine evolution (now called genetic algorithms) (Friedberg, 1958; Friedberg *et al.*, 1959) were based on the undoubtedly correct belief that by making an appropriate series of small mutations to a machine code program, one can generate a program with good performance for any particular simple task. The idea, then, was to try random mutations with a selection process to preserve mutations that seemed useful. Despite thousands of hours of CPU time, almost no progress was demonstrated. Modern genetic algorithms use better representations and have shown more success.

Failure to come to grips with the "combinatorial explosion" was one of the main criticisms of AI contained in the Lighthill report (Lighthill, 1973), which formed the basis for the decision by the British government to end support for AI research in all but two universities. (Oral tradition paints a somewhat different and more colorful picture, with political ambitions and personal animosities whose description is beside the point.)

A third difficulty arose because of some fundamental limitations on the basic structures being used to generate intelligent behavior. For example, Minsky and Papert's book *Perceptrons* (1969) proved that, although perceptrons (a simple form of neural network) could be shown to learn anything they were capable of representing, they could represent very little. In particular, a two-input perceptron could not be trained to recognize when its two inputs were different. Although their results did not apply to more complex, multilayer networks, research funding for neural-net research soon dwindled to almost nothing. Ironically, the new back-propagation learning algorithms for multilayer networks that were to cause an enormous resurgence in neural-net research in the late 1980s were actually discovered first in 1969 (Bryson and Ho, 1969).

### Knowledge-based systems: The key to power? (1969–1979)

## WEAK METHODS

The picture of problem solving that had arisen during the first decade of AI research was of a general-purpose search mechanism trying to string together elementary reasoning steps to find complete solutions. Such approaches have been called weak methods, because, although general, they do not scale up to large or difficult problem instances. The alternative to weak methods is to use more powerful, domain-specific knowledge that allows larger reasoning steps and can more easily handle typically occurring cases in narrow areas of expertise. One might say that to solve a hard problem, you have to almost know the answer already.

The DENDRAL program (Buchanan *et al.*, 1969) was an early example of this approach. It was developed at Stanford, where Ed Feigenbaum (a former student of Herbert Simon), Bruce Buchanan (a philosopher turned computer scientist), and Joshua Lederberg (a Nobel laureate geneticist) teamed up to solve the problem of inferring molecular structure from the information provided by a mass spectrometer. The input to the program consists of the elementary formula of the molecule (e.g.,  $C_6H_{13}NO_2$ ) and the mass spectrum giving the masses of the various fragments of the molecule generated when it is bombarded by an electron beam.

For example, the mass spectrum might contain a peak at  $m = 15$ , corresponding to the mass of a methyl ( $\text{CH}_3$ ) fragment.

The naive version of the program generated all possible structures consistent with the formula, and then predicted what mass spectrum would be observed for each, comparing this with the actual spectrum. As one might expect, this is intractable for decent-sized molecules. The DENDRAL researchers consulted analytical chemists and found that they worked by looking for well-known patterns of peaks in the spectrum that suggested common substructures in the molecule. For example, the following rule is used to recognize a ketone ( $\text{C}=\text{O}$ ) subgroup (which weighs 28):

if there are two peaks at  $x_1$  and  $x_2$  such that  
 (a)  $x_1 + x_2 = M + 28$  ( $M$  is the mass of the whole molecule);  
 (b)  $x_1 - 28$  is a high peak;  
 (c)  $x_2 - 28$  is a high peak;  
 (d) At least one of  $x_1$  and  $x_2$  is high.  
 then there is a ketone subgroup

Recognizing that the molecule contains a particular substructure reduces the number of possible candidates enormously. DENDRAL was powerful because

All the relevant theoretical knowledge to solve these problems has been mapped over from its general form in the [spectrum prediction component] ("first principles") to efficient special forms ("cookbook recipes"). (Feigenbaum *et al.*, 1971)

The significance of DENDRAL was that it was the first successful *knowledge-intensive* system: its expertise derived from large numbers of special-purpose rules. Later systems also incorporated the main theme of McCarthy's Advice Taker approach—the clean separation of the knowledge (in the form of rules) from the reasoning component.

EXPERTSYSTEMS

With this lesson in mind, Feigenbaum and others at Stanford began the Heuristic Programming Project (HPP), to investigate the extent to which the new methodology of **expert systems** could be applied to other areas of human expertise. The next major effort was in the area of medical diagnosis. Feigenbaum, Buchanan, and Dr. Edward Shortliffe developed MYCIN to diagnose blood infections. With about 450 rules, MYCIN was able to perform as well as some experts, and considerably better than junior doctors. It also contained two major differences from DENDRAL. First, unlike the DENDRAL rules, no general theoretical model existed from which the MYCIN rules could be deduced. They had to be acquired from extensive interviewing of experts, who in turn acquired them from textbooks, other experts, and direct experience of cases. Second, the rules had to reflect the uncertainty associated with medical knowledge. MYCIN incorporated a calculus of uncertainty called **certainty factors** (see Chapter 14), which seemed (at the time) to fit well with how doctors assessed the impact of evidence on the diagnosis.

The importance of domain knowledge was also apparent in the area of understanding natural language. Although Winograd's SHRDLU system for understanding natural language had engendered a good deal of excitement, its dependence on syntactic analysis caused some of the same problems as occurred in the early machine translation work. It was able to overcome ambiguity and understand pronoun references, but this was mainly because it was

designed specifically for one area—the blocks world. Several researchers, including Eugene Charniak, a fellow graduate student of Winograd's at MIT, suggested that robust language understanding would require general knowledge about the world and a general method for using that knowledge.

At Yale, the linguist-turned-AI-researcher Roger Schank emphasized this point, claiming, "There is no such thing as syntax," which upset a lot of linguists, but did serve to start a useful discussion. Schank and his students built a series of programs (Schank and Abelson, 1977; Wilensky, 1978; Schank and Riesbeck, 1981; Dyer, 1983) that all had the task of understanding natural language. The emphasis, however, was less on language *per se* and more on the problems of representing and reasoning with the knowledge required for language understanding. The problems included representing stereotypical situations (Cullingford, 1981), describing human memory organization (Rieger, 1976; Kolodner, 1983), and understanding plans and goals (Wilensky, 1983).

The widespread growth of applications to real-world problems caused a concurrent increase in the demands for workable knowledge representation schemes. A large number of different representation and reasoning languages were developed. Some were based on logic—for example, the Prolog language became popular in Europe, and the PLANNER family in the United States. Others, following Minsky's idea of **frames** (1975), adopted a more structured approach, assembling facts about particular object and event types and arranging the types into a large taxonomic hierarchy analogous to a biological taxonomy.

FRAMES

### AI becomes an industry (1980–present)

The first successful commercial expert system, R1, began operation at the Digital Equipment Corporation (McDermott, 1982). The program helped configure orders for new computer systems; by 1986, it was saving the company an estimated \$40 million a year. By 1988, DEC's AI group had 40 expert systems deployed, with more on the way. Du Pont had 100 in use and 500 in development, saving an estimated \$10 million a year. Nearly every major U.S. corporation had its own AI group and was either using or investigating expert systems.

In 1981, the Japanese announced the "Fifth Generation" project, a 10-year plan to build intelligent computers running Prolog. In response the United States formed the Microelectronics and Computer Technology Corporation (MCC) as a research consortium designed to assure national competitiveness. In both cases, AI was part of a broad effort, including chip design and human-interface research. However, the AI components of MCC and the Fifth Generation projects never met their ambitious goals. In Britain, the Alvey report reinstated the funding that was cut by the Lighthill report.<sup>15</sup>

Overall, the AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988. Soon after that came a period called the "AI Winter," in which many companies suffered as they failed to deliver on extravagant promises.

<sup>15</sup> To save embarrassment, a new field called IKBS (Intelligent Knowledge-Based Systems) was invented because Artificial Intelligence had been officially canceled.

### The return of neural networks (1986–present)

Although computer science had largely abandoned the field of neural networks in the late 1970s, work continued in other fields. Physicists such as John Hopfield (1982) used techniques from statistical mechanics to analyze the storage and optimization properties of networks, treating collections of nodes like collections of atoms. Psychologists including David Rumelhart and Geoff Hinton continued the study of neural-net models of memory. As we discuss in Chapter 20, the real impetus came in the mid-1980s when at least four different groups reinvented the back-propagation learning algorithm first found in 1969 by Bryson and Ho. The algorithm was applied to many learning problems in computer science and psychology, and the widespread dissemination of the results in the collection *Parallel Distributed Processing* (Rumelhart and McClelland, 1986) caused great excitement.

CONNECTIONIST

These so-called **connectionist** models of intelligent systems were seen by some as direct competitors both to the symbolic models promoted by Newell and Simon and to the logicist approach of McCarthy and others (Smolensky, 1988). It might seem obvious that at some level humans manipulate symbols—in fact, Terrence Deacon's book *The Symbolic Species* (1997) suggests that this is the *defining characteristic* of humans, but the most ardent connectionists questioned whether symbol manipulation had any real explanatory role in detailed models of cognition. This question remains unanswered, but the current view is that connectionist and symbolic approaches are complementary, not competing.

### AI becomes a science (1987–present)

Recent years have seen a revolution in both the content and the methodology of work in artificial intelligence.<sup>16</sup> It is now more common to build on existing theories than to propose brand new ones, to base claims on rigorous theorems or hard experimental evidence rather than on intuition, and to show relevance to real-world applications rather than toy examples.

AI was founded in part as a rebellion against the limitations of existing fields like control theory and statistics, but now it is embracing those fields. As David McAllester (1998) put it,

In the early period of AI it seemed plausible that new forms of symbolic computation, e.g., frames and semantic networks, made much of classical theory obsolete. This led to a form of isolationism in which AI became largely separated from the rest of computer science. This isolationism is currently being abandoned. There is a recognition that machine learning should not be isolated from information theory, that uncertain reasoning should not be isolated from stochastic modeling, that search should not be isolated from classical optimization and control, and that automated reasoning should not be isolated from formal methods and static analysis.

In terms of methodology, AI has finally come firmly under the scientific method. To be accepted, hypotheses must be subjected to rigorous empirical experiments, and the results must

<sup>16</sup> Some have characterized this change as a victory of the **neats**—those who think that AI theories should be grounded in mathematical rigor—over the **scruffies**—those who would rather try out lots of ideas, write some programs, and then assess what seems to be working. Both approaches are important. A shift toward neatness implies that the field has reached a level of stability and maturity. Whether that stability will be disrupted by a new scruffy idea is another question.



be analyzed statistically for their importance (Cohen, 1995). Through the use of the Internet and shared repositories of test data and code, it is now possible to replicate experiments.

The field of speech recognition illustrates the pattern. In the 1970s, a wide variety of different architectures and approaches were tried. Many of these were rather *ad hoc* and fragile, and were demonstrated on only a few specially selected examples. In recent years, approaches based on **hidden Markov models** (HMMs) have come to dominate the area. Two aspects of HMMs are relevant. First, they are based on a rigorous mathematical theory. This has allowed speech researchers to build on several decades of mathematical results developed in other fields. Second, they are generated by a process of training on a large corpus of real speech data. This ensures that the performance is robust, and in rigorous blind tests the HMMs have been improving their scores steadily. Speech technology and the related field of handwritten character recognition are already making the transition to widespread industrial and consumer applications.

Neural networks also fit this trend. Much of the work on neural nets in the 1980s was done in an attempt to scope out what could be done and to learn how neural nets differ from "traditional" techniques. Using improved methodology and theoretical frameworks, the field arrived at an understanding in which neural nets can now be compared with corresponding techniques from statistics, pattern recognition, and machine learning, and the most promising technique can be applied to each application. As a result of these developments, so-called **data mining** technology has spawned a vigorous new industry.

Judea Pearl's (1988) *Probabilistic Reasoning in Intelligent Systems* led to a new acceptance of probability and decision theory in AI, following a resurgence of interest epitomized by Peter Cheeseman's (1985) article "In Defense of Probability." The **Bayesian network** formalism was invented to allow efficient representation of, and rigorous reasoning with, uncertain knowledge. This approach largely overcomes many problems of the probabilistic reasoning systems of the 1960s and 1970s; it now dominates AI research on uncertain reasoning and expert systems. The approach allows for learning from experience, and it combines the best of classical AI and neural nets. Work by Judea Pearl (1982a) and by Eric Horvitz and David Heckerman (Horvitz and Heckerman, 1986; Horvitz et al., 1986) promoted the idea of *normative* expert systems: ones that act rationally according to the laws of decision theory and do not try to imitate the thought steps of human experts. The windows<sup>TM</sup> operating system includes several normative diagnostic expert systems for correcting problems. Chapters 13 to 16 cover this area.

Similar gentle revolutions have occurred in robotics, computer vision, and knowledge representation. A better understanding of the problems and their complexity properties, combined with increased mathematical sophistication, has led to workable research agendas and robust methods. In many cases, formalization and specialization have also led to fragmentation: topics such as vision and robotics are increasingly isolated from "mainstream" AI work. The unifying view of AI as rational agent design is one that can bring unity back to these disparate fields.

### The emergence of intelligent agents (1995–present)

Perhaps encouraged by the progress in solving the subproblems of AI, researchers have also started to look at the "whole agent" problem again. The work of Allen Newell, John Laird, and Paul Rosenbloom on SOAR (Newell, 1990; Laird *et al.*, 1987) is the best-known example of a complete agent architecture. The so-called situated movement aims to understand the workings of agents embedded in real environments with continuous sensory inputs. One of the most important environments for intelligent agents is the Internet. AI systems have become so common in web-based applications that the “-bot” suffix has entered everyday language. Moreover, AI technologies underlie many Internet tools, such as search engines, recommender systems, and Web site construction systems.

Besides the first edition of this text (Russell and Norvig, 1995), other recent texts have also adopted the agent perspective (Poole *et al.*, 1998; Nilsson, 1998). One consequence of trying to build complete agents is the realization that the previously isolated subfields of AI might need to be reorganized somewhat when their results are to be tied together. In particular, it is now widely appreciated that sensory systems (vision, sonar, speech recognition, etc.) cannot deliver perfectly reliable information about the environment. Hence, reasoning and planning systems must be able to handle uncertainty. A second major consequence of the agent perspective is that AI has been drawn into much closer contact with other fields, such as control theory and economics, that also deal with agents.

## 1.4 THE STATE OF THE ART

---

What can AI do today? A concise answer is difficult, because there are so many activities in so many subfields. Here we sample a few applications; others appear throughout the book.

**Autonomous planning and scheduling:** A hundred million miles from Earth, NASA's Remote Agent program became the first on-board autonomous planning program to control the scheduling of operations for a spacecraft (Jonsson *et al.*, 2000). Remote Agent generated plans from high-level goals specified from the ground, and it monitored the operation of the spacecraft as the plans were executed—detecting, diagnosing, and recovering from problems as they occurred.

**Game playing:** IBM's Deep Blue became the first computer program to defeat the world champion in a chess match when it bested Garry Kasparov by a score of 3.5 to 2.5 in an exhibition match (Goodman and Keene, 1997). Kasparov said that he felt a "new kind of intelligence" across the board from him. *Newsweek* magazine described the match as "The brain's last stand." The value of IBM's stock increased by \$18 billion.

**Autonomous control:** The ALVINN computer vision system was trained to steer a car to keep it following a lane. It was placed in CMU's NAVLAB computer-controlled minivan and used to navigate across the United States—for 2850 miles it was in control of steering the vehicle 98% of the time. A human took over the other 2%, mostly at exit ramps. NAVLAB has video cameras that transmit road images to ALVINN, which then computes the best direction to steer, based on experience from previous training runs.

**Diagnosis:** Medical diagnosis programs based on probabilistic analysis have been able to perform at the level of an expert physician in several areas of medicine. Heckerman (1991) describes a case where a leading expert on lymph-node pathology scoffs at a program's diagnosis of an especially difficult case. The creators of the program suggest he ask the computer for an explanation of the diagnosis. The machine points out the major factors influencing its decision and explains the subtle interaction of several of the symptoms in this case. Eventually, the expert agrees with the program.

**Logistics Planning:** During the Persian Gulf crisis of 1991, U.S. forces deployed a Dynamic Analysis and Replanning Tool, DART (Cross and Walker, 1994), to do automated logistics planning and scheduling for transportation. This involved up to 50,000 vehicles, cargo, and people at a time, and had to account for starting points, destinations, routes, and conflict resolution among all parameters. The AI planning techniques allowed a plan to be generated in hours that would have taken weeks with older methods. The Defense Advanced Research Project Agency (DARPA) stated that this single application more than paid back DARPA's 30-year investment in AI.

**Robotics:** Many surgeons now use robot assistants in microsurgery. HipNav (DiGioia *et al.*, 1996) is a system that uses computer vision techniques to create a three-dimensional model of a patient's internal anatomy and then uses robotic control to guide the insertion of a hip replacement prosthesis.

**Language understanding and problem solving:** PROVERB (Littman *et al.*, 1999) is a computer program that solves crossword puzzles better than most humans, using constraints on possible word fillers, a large database of past puzzles, and a variety of information sources including dictionaries and online databases such as a list of movies and the actors that appear in them. For example, it determines that the clue "Nice Story" can be solved by "ETAGE" because its database includes the clue/solution pair "Story in France/ETAGE" and because it recognizes that the patterns "Nice X" and "X in France" often have the same solution. The program does not know that Nice is a city in France, but it can solve the puzzle.

These are just a few examples of artificial intelligence systems that exist today. Not magic or science fiction—but rather science, engineering, and mathematics, to which this book provides an introduction.

---

## 1.5 SUMMARY

This chapter defines AI and establishes the cultural background against which it has developed. Some of the important points are as follows:

- Different people think of AI differently. Two important questions to ask are: Are you concerned with thinking or behavior? Do you want to model humans or work from an ideal standard?
- In this book, we adopt the view that intelligence is concerned mainly with **rational action**. Ideally, an **intelligent agent** takes the best possible action in a situation. We will study the problem of building agents that are intelligent in this sense.

- Philosophers (going back to 400 B.c.) made AI conceivable by considering the ideas that the mind is in some ways like a machine, that it operates on knowledge encoded in some internal language, and that thought can be used to choose what actions to take.
- Mathematicians provided the tools to manipulate statements of logical certainty as well as uncertain, probabilistic statements. They also set the groundwork for understanding computation and reasoning about algorithms.
- Economists formalized the problem of making decisions that maximize the expected outcome to the decision-maker.
- Psychologists adopted the idea that humans and animals can be considered information-processing machines. Linguists showed that language use fits into this model.
- Computer engineers provided the artifacts that make AI applications possible. AI programs tend to be large, and they could not work without the great advances in speed and memory that the computer industry has provided.
- Control theory deals with designing devices that act optimally on the basis of feedback from the environment. Initially, the mathematical tools of control theory were quite different from AI, but the fields are coming closer together.
- The history of AI has had cycles of success, misplaced optimism, and resulting cutbacks in enthusiasm and funding. There have also been cycles of introducing new creative approaches and systematically refining the best ones.
- AI has advanced more rapidly in the past decade because of greater use of the scientific method in experimenting with and comparing approaches.
- Recent progress in understanding the theoretical basis for intelligence has gone hand in hand with improvements in the capabilities of real systems. The subfields of AI have become more integrated, and AI has found common ground with other disciplines.

---

## BIBLIOGRAPHICAL AND HISTORICAL NOTES

The methodological status of artificial intelligence is investigated in *The Sciences of the Artificial*, by Herb Simon (1981), which discusses research areas concerned with complex artifacts. It explains how AI can be viewed as both science and mathematics. Cohen (1995) gives an overview of experimental methodology within AI. Ford and Hayes (1995) give an opinionated view of the usefulness of the Turing Test.

*Artificial Intelligence: The Very Idea*, by John Haugeland (1985) gives a readable account of the philosophical and practical problems of AI. Cognitive science is well described by several recent texts (Johnson-Laird, 1988; Stillings *et al.*, 1995; Thagard, 1996) and by the *Encyclopedia of the Cognitive Sciences* (Wilson and Keil, 1999). Baker (1989) covers the syntactic part of modern linguistics, and Chierchia and McConnell-Ginet (1990) cover semantics. Jurafsky and Martin (2000) cover computational linguistics.

Early AI is described in Feigenbaum and Feldman's *Computers and Thought* (1963), Minsky's *Semantic Information Processing* (1968), and the *Machine Intelligence* series edited by Donald Michie. A large number of influential papers have been anthologized by Webber

and Nilsson (1981) and by Luger (1995). Early papers on neural networks are collected in *Neurocomputing* (Anderson and Rosenfeld, 1988). The *Encyclopedia of AI* (Shapiro, 1992) contains survey articles on almost every topic in AI. These articles usually provide a good entry point into the research literature on each topic.

The most recent work appears in the proceedings of the major AI conferences: the biennial International Joint Conference on AI (IJCAI), the annual European Conference on AI (ECAI), and the National Conference on AI, more often known as AAAI, after its sponsoring organization. The major journals for general AI are *Artificial Intelligence*, *Computational Intelligence*, the *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *IEEE Intelligent Systems*, and the electronic *Journal of Artificial Intelligence Research*. There are also many conferences and journals devoted to specific areas, which we cover in the appropriate chapters. The main professional societies for AI are the American Association for Artificial Intelligence (AAAI), the ACM Special Interest Group in Artificial Intelligence (SIGART), and the Society for Artificial Intelligence and Simulation of Behaviour (AISB). AAAI's *AI Magazine* contains many topical and tutorial articles, and its website, [aaai.org](http://aaai.org), contains news and background information.

---

## EXERCISES

These exercises are intended to stimulate discussion, and some might be set as term projects. Alternatively, preliminary attempts can be made now, and these attempts can be reviewed after the completion of the book.

**1.1** Define in your own words: (a) intelligence, (b) artificial intelligence, (c) agent.



**1.2** Read Turing's original paper on AI (Turing, 1950). In the paper, he discusses several potential objections to his proposed enterprise and his test for intelligence. Which objections still carry some weight? Are his refutations valid? Can you think of new objections arising from developments since he wrote the paper? In the paper, he predicts that, by the year 2000, a computer will have a 30% chance of passing a five-minute Turing Test with an unskilled interrogator. What chance do you think a computer would have today? In another 50 years?



**1.3** Every year the Loebner prize is awarded to the program that comes closest to passing a version of the Turing test. Research and report on the latest winner of the Loebner prize. What techniques does it use? How does it advance the state of the art in AI?

**1.4** There are well-known classes of problems that are intractably difficult for computers, and other classes that are provably undecidable. Does this mean that AI is impossible?

**1.5** Suppose we extend Evans's ANALOGY program so that it can score 200 on a standard IQ test. Would we then have a program more intelligent than a human? Explain.

**1.6** How could introspection—reporting on one's inner thoughts—be inaccurate? Could I be wrong about what I'm thinking? Discuss.



**1.7** Examine the AI literature to discover whether the following tasks can currently be solved by computers:

- a. Playing a decent game of table tennis (ping-pong).
- b. Driving in the center of Cairo.
- c. Buying a week's worth of groceries at the market.
- d. Buying a week's worth of groceries on the web.
- e. Playing a decent game of bridge at a competitive level.
- f. Discovering and proving new mathematical theorems.
- g. Writing an intentionally funny story.
- h. Giving competent legal advice in a specialized area of law.
  - i. Translating spoken English into spoken Swedish in real time.
- j. Performing a complex surgical operation.

For the currently infeasible tasks, try to find out what the difficulties are and predict when, if ever, they will be overcome.

**1.8** Some authors have claimed that perception and motor skills are the most important part of intelligence, and that "higher level" capacities are necessarily parasitic—simple add-ons to these underlying facilities. Certainly, most of evolution and a large part of the brain have been devoted to perception and motor skills, whereas AI has found tasks such as game playing and logical inference to be easier, in many ways, than perceiving and acting in the real world. Do you think that AI's traditional focus on higher-level cognitive abilities is misplaced?

**1.9** Why would evolution tend to result in systems that act rationally? What goals are such systems designed to achieve?

**1.10** Are reflex actions (such as moving your hand away from a hot stove) rational? Are they intelligent?

**1.11** "Surely computers cannot be intelligent—they can do only what their programmers tell them." Is the latter statement true, and does it imply the former?

**1.12** "Surely animals cannot be intelligent—they can do only what their genes tell them." Is the latter statement true, and does it imply the former?

**1.13** "Surely animals, humans, and computers cannot be intelligent—they can do only what their constituent atoms are told to do by the laws of physics." Is the latter statement true, and does it imply the former?

# 2

## INTELLIGENT AGENTS

*In which we discuss the nature of agents, perfect or otherwise, the diversity of environments, and the resulting menagerie of agent types.*

Chapter 1 identified the concept of **rational agents** as central to our approach to artificial intelligence. In this chapter, we make this notion more concrete. We will see that the concept of rationality can be applied to a wide variety of agents operating in any imaginable environment. Our plan in this book is to use this concept to develop a small set of design principles for building successful agents—systems that can reasonably be called **intelligent**.

We will begin by examining agents, environments, and the coupling between them. The observation that some agents behave better than others leads naturally to the idea of a rational agent—one that behaves as well as possible. How well an agent can behave depends on the nature of the environment; some environments are more difficult than others. We give a crude categorization of environments and show how properties of an environment influence the design of suitable agents for that environment. We describe a number of basic "skeleton" agent designs, which will be fleshed out in the rest of the book.

### 2.1 AGENTS AND ENVIRONMENTS

ENVIRONMENT

SENSOR

ACTUATOR

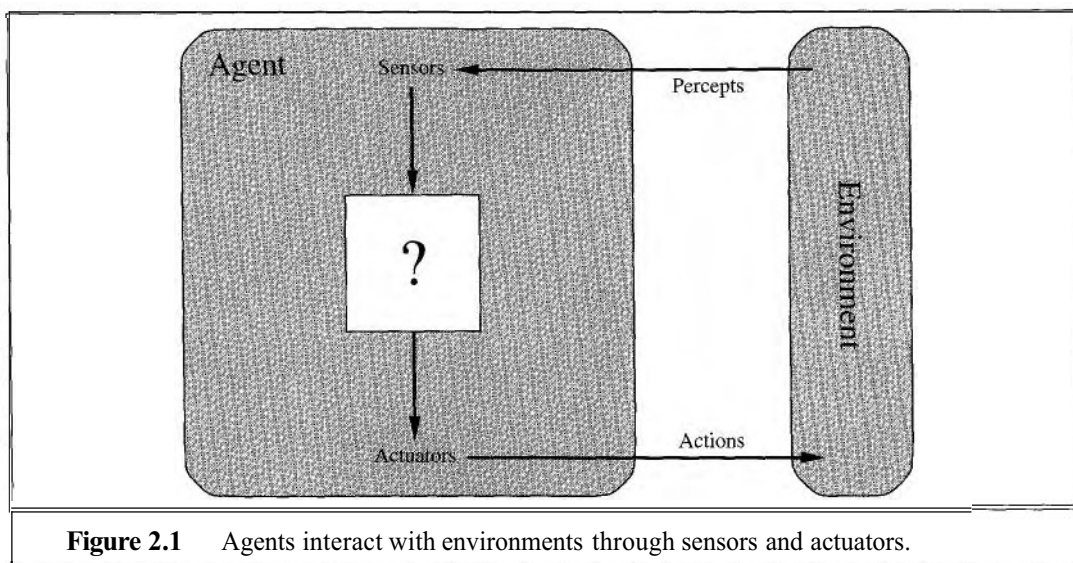
An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators**. This simple idea is illustrated in Figure 2.1. A human agent has eyes, ears, and other organs for sensors and hands, legs, mouth, and other body parts for actuators. A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators. A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets. We will make the general assumption that every agent can perceive its own actions (but not always the effects).

PERCEPT

PERCEPTSEQUENCE



We use the term **percept** to refer to the agent's perceptual inputs at any given instant. An agent's **percept sequence** is the complete history of everything the agent has ever perceived. In general, *an agent's choice of action at any given instant can depend on the entire percept sequence observed to date*. If we can specify the agent's choice of action for every possible



percept sequence, then we have said more or less everything there is to say about the agent. Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action.

We can imagine tabulating the agent function that describes any given agent; for most agents, this would be a very large table—infinite, in fact, unless we place a bound on the length of percept sequences we want to consider. Given an agent to experiment with, we can, in principle, construct this table by trying out all possible percept sequences and recording which actions the agent does in response.<sup>1</sup> The table is, of course, an external characterization of the agent. Internally, the agent function for an artificial agent will be implemented by an **agent program**. It is important to keep these two ideas distinct. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running on the agent architecture.

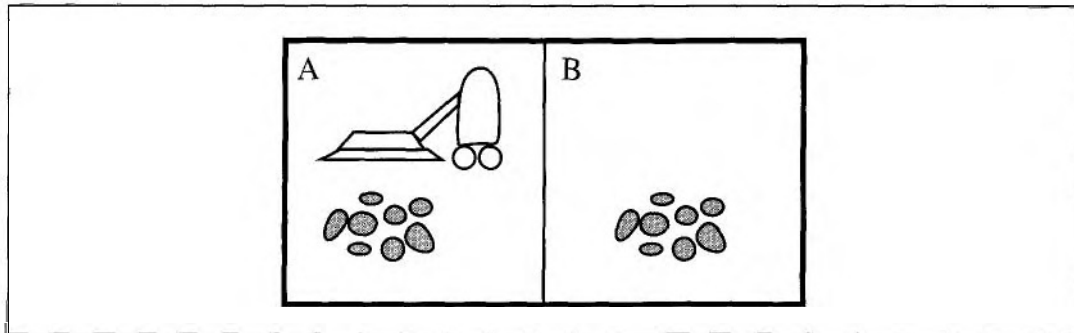
To illustrate these ideas, we will use a very simple example—the vacuum-cleaner world shown in Figure 2.2. This world is so simple that we can describe everything that happens; it's also a made-up world, so we can invent many variations. This particular world has just two locations: squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right, suck up the dirt, or do nothing. One very simple agent function is the following: if the current square is dirty, then suck, otherwise move to the other square. A partial tabulation of this agent function is shown in Figure 2.3. A simple agent program for this agent function is given later in the chapter, in Figure 2.8.

Looking at Figure 2.3, we see that various vacuum-world agents can be defined simply by filling in the right-hand column in various ways. The obvious question, then, is this: What



<sup>1</sup> If the agent uses some randomization to choose its actions, then we would have to try each sequence many times to identify the probability of each action. One might imagine that acting randomly is rather silly, but we'll see later in this chapter that it can be very intelligent.





**Figure 2.2** A vacuum-cleaner world with just two locations.

Percept sequence	Action
<i>[A, Clean]</i>	<i>Right</i>
<i>[A, Dirty]</i>	Suck
<i>[B, Clean]</i>	<i>Left</i>
<i>[B, Dirty]</i>	Suck
<i>[A, Clean], [A, Clean]</i>	<i>Right</i>
<i>[A, Clean], [A, Dirty]</i>	Suck
<i>[A, Clean], [A, Clean], [A, Clean]</i>	<i>Right</i>
<i>[A, Clean], [A, Clean], [A, Dirty]</i>	Suck

**Figure 2.3** Partial tabulation of a simple agent function for the vacuum-cleaner world shown in Figure 2.2.

*is the right way to Jill out the table?* In other words, what makes an agent good or bad, intelligent or stupid? We answer these questions in the next section.

Before closing this section, we will remark that the notion of an agent is meant to be a tool for analyzing systems, not an absolute characterization that divides the world into agents and non-agents. One could view a hand-held calculator as an agent that chooses the action of displaying “4” when given the percept sequence “2 + 2 =,” but such an analysis would hardly aid our understanding of the calculator.

## 2.2 GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

### RATIONAL AGENT

A **rational agent** is one that does the right thing—conceptually speaking, every entry in the table for the agent function is filled out correctly. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing? As a first approximation, we will say that the right action is the one that will cause the agent to be

most successful. Therefore, we will need some way to measure success. Together with the description of the environment and the sensors and actuators of the agent, this will provide a complete specification of the task facing the agent. Given this, we can define more precisely what it means to be rational.

### Performance measures

PERFORMANCE  
MEASURE

A **performance measure** embodies the criterion for success of an agent's behavior. When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives. This sequence of actions causes the environment to go through a sequence of states. If the sequence is desirable, then the agent has performed well. Obviously, there is not one fixed measure suitable for all agents. We could ask the agent for a subjective opinion of how happy it is with its own performance, but some agents would be unable to answer, and others would delude themselves.<sup>2</sup> Therefore, we will insist on an objective performance measure, typically one imposed by the designer who is constructing the agent.

Consider the vacuum-cleaner agent from the preceding section. We might propose to measure performance by the amount of dirt cleaned up in a single eight-hour shift. With a rational agent, of course, what you ask for is what you get. A rational agent can maximize this performance measure by cleaning up the dirt, then dumping it all on the floor, then cleaning it up again, and so on. A more suitable performance measure would reward the agent for having a clean floor. For example, one point could be awarded for each clean square at each time step (perhaps with a penalty for electricity consumed and noise generated). *As a general rule, it is better to design performance measures according to what one actually wants in the environment, rather than according to how one thinks the agent should behave.*



The selection of a performance measure is not always easy. For example, the notion of "clean floor" in the preceding paragraph is based on average cleanliness over time. Yet the same average cleanliness can be achieved by two different agents, one of which does a mediocre job all the time: while the other cleans energetically but takes long breaks. Which is preferable might seem to be a fine point of janitorial science, but in fact it is a deep philosophical question with far-reaching implications. Which is better—a reckless life of highs and lows, or a safe but humdrum existence? Which is better—an economy where everyone lives in moderate poverty, or one in which some live in plenty while others are very poor? We will leave these questions as an exercise for the diligent reader.

### Rationality

What is rational at any given time depends on four things:

- The performance measure that defines the criterion of success.
- The agent's prior knowledge of the environment.
- The actions that the agent can perform.
- The agent's percept sequence to date.

<sup>2</sup> Human agents in particular are notorious for "sour grapes" —believing they did not really want something after not getting it, as in, "Oh well, never mind, I didn't want that stupid Nobel prize anyway."

## DEFINITION OF A RATIONAL AGENT



This leads to a **definition of a rational agent**:

*For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*

Consider the simple vacuum-cleaner agent that cleans a square if it is dirty and moves to the other square if not; this is the agent function tabulated in Figure 2.3. Is this a rational agent? That depends! First, we need to say what the performance measure is, what is known about the environment, and what sensors and actuators the agent has. Let us assume the following:

- The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000 time steps.
- The "geography" of the environment is known *a priori* (Figure 2.2) but the dirt distribution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current square. The *Left* and *Right* actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
- The only available actions are *Left*, *Right*, *Suck*, and *NoOp* (do nothing).
- The agent correctly perceives its location and whether that location contains dirt.

We claim that *under these circumstances* the agent is indeed rational; its expected performance is at least as high as any other agent's. Exercise 2.4 asks you to prove this.

One can see easily that the same agent would be irrational under different circumstances. For example, once all the dirt is cleaned up it will oscillate needlessly back and forth; if the performance measure includes a penalty of one point for each movement left or right, the agent will fare poorly. A better agent for this case would do nothing once it is sure that all the squares are clean. If clean squares can become dirty again, the agent should occasionally check and re-clean them if needed. If the geography of the environment is unknown, the agent will need to explore it rather than stick to squares A and B. Exercise 2.4 asks you to design agents for these cases.

### Omniscience, learning, and autonomy

## OMNISCIENCE

We need to be careful to distinguish between rationality and **omniscience**. An omniscient agent knows the *actual* outcome of its actions and can act accordingly; but omniscience is impossible in reality. Consider the following example: I am walking along the Champs Elysées one day and I see an old friend across the street. There is no traffic nearby and I'm not otherwise engaged, so, being rational, I start to cross the street. Meanwhile, at 33,000 feet, a cargo door falls off a passing airliner,<sup>3</sup> and before I make it to the other side of the street I am flattened. Was I irrational to cross the street? It is unlikely that my obituary would read "Idiot attempts to cross street."

This example shows that rationality is not the same as perfection. Rationality maximizes *expected* performance, while perfection maximizes *actual* performance. Retreating from a requirement of perfection is not just a question of being fair to agents. The point is

<sup>3</sup> See N. Henderson, "New door latches urged for Boeing 747 jumbo jets," *Washington Post*, August 24, 1989.

that if we expect an agent to do what turns out to be the best action after the fact, it will be impossible to design an agent to fulfill this specification—unless we improve the performance of crystal balls or time machines.

Our definition of rationality does not require omniscience, then, because the rational choice depends only on the percept sequence to date. We must also ensure that we haven't inadvertently allowed the agent to engage in decidedly underintelligent activities. For example, if an agent does not look both ways before crossing a busy road, then its percept sequence will not tell it that there is a large truck approaching at high speed. Does our definition of rationality say that it's now OK to cross the road? Far from it! First, it would not be rational to cross the road given this uninformative percept sequence: the risk of accident from crossing without looking is too great. Second, a rational agent should choose the "looking" action before stepping into the street, because looking helps maximize the expected performance. Doing actions in order *to modify* future percepts—sometimes called **information gathering**—is an important part of rationality and is covered in depth in Chapter 16. A second example of information gathering is provided by the exploration that must be undertaken by a vacuum-cleaning agent in an initially unknown environment.

INFORMATION  
GATHERING  
EXPLORATION

LEARNING

Our definition requires a rational agent not only to gather information, but also to learn as much as possible from what it perceives. The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented. There are extreme cases in which the environment is completely known a priori. In such cases, the agent need not perceive or learn; it simply acts correctly. Of course, such agents are very fragile. Consider the lowly dung beetle. After digging its nest and laying its eggs, it fetches a ball of dung from a nearby heap to plug the entrance. If the ball of dung is removed from its grasp en route, the beetle continues on and pantomimes plugging the nest with the nonexistent dung ball, never noticing that it is missing. Evolution has built an assumption into the beetle's behavior, and when it is violated, unsuccessful behavior results. Slightly more intelligent is the sphex wasp. The female sphex will dig a burrow, go out and sting a caterpillar and drag it to the burrow, enter the burrow again to check all is well, drag the caterpillar inside, and lay its eggs. The caterpillar serves as a food source when the eggs hatch. So far so good, but if an entomologist moves the caterpillar a few inches away while the sphex is doing the check, it will revert back to the "drag" step of its plan, and will continue the plan without modification, even after dozens of caterpillar-moving interventions. The sphex is unable to learn that its innate plan is failing, and thus will not change it.

Successful agents split the task of computing the agent function into three different periods: when the agent is being designed, some of the computation is done by its designers; when it is deliberating on its next action, the agent does more computation; and as it learns from experience, it does even more computation to decide how to modify its behavior.

To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy. A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge. For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not. As a practical matter, one seldom requires complete autonomy from the start: when the agent has had little or no experience, it

AUTONOMY

would have to act randomly unless the designer gave some assistance. So, just as evolution provides animals with enough built-in reflexes so that they can survive long enough to learn for themselves, it would be reasonable to provide an artificial intelligent agent with some initial knowledge as well as an ability to learn. After sufficient experience of its environment, the behavior of a rational agent can become effectively *independent* of its prior knowledge. Hence, the incorporation of learning allows one to design a single rational agent that will succeed in a vast variety of environments.

## 2.3 THE NATURE OF ENVIRONMENTS

### TASK ENVIRONMENTS

Now that we have a definition of rationality, we are almost ready to think about building rational agents. First, however, we must think about **task environments**, which are essentially the "problems" to which rational agents are the "solutions." We begin by showing how to specify a task environment, illustrating the process with a number of examples. We then show that task environments come in a variety of flavors. The flavor of the task environment directly affects the appropriate design for the agent program.

### Specifying the task environment

### PEAS

In our discussion of the rationality of the simple vacuum-cleaner agent, we had to specify the performance measure, the environment, and the agent's actuators and sensors. We will group all these together under the heading of the **task environment**. For the acronymically minded, we call this the PEAS (Performance, Environment, Actuators, Sensors) description. In designing an agent, the first step must always be to specify the task environment as fully as possible.

The vacuum world was a simple example; let us consider a more complex problem: an automated taxi driver. We will use this example throughout the rest of the chapter. We should point out, before the reader becomes alarmed, that a fully automated taxi is currently somewhat beyond the capabilities of existing technology. (See page 27 for a description of an existing driving robot, or look at recent proceedings of the conferences on Intelligent Transportation Systems.) The full driving task is extremely *open-ended*. There is no limit to the novel combinations of circumstances that can arise — another reason we chose it as a focus for discussion. Figure 2.4 summarizes the PEAS description for the taxi's task environment. We discuss each element in more detail in the following paragraphs.

First, what is the **performance measure** to which we would like our automated driver to aspire? Desirable qualities include getting to the correct destination; minimizing fuel consumption and wear and tear; minimizing the trip time and/or cost; minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort; maximizing profits. Obviously, some of these goals conflict, so there will be tradeoffs involved.

Next, what is the driving **environment** that the taxi will face? Any taxi driver must deal with a variety of roads, ranging from rural lanes and urban alleys to 12-lane freeways. The roads contain other traffic, pedestrians, stray animals, road works, police cars, puddles,

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe: fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

**Figure 2.4** PEAS description of the task environment for an automated taxi.

and potholes. The taxi must also interact with potential and actual passengers. There are also some optional choices. The taxi might need to operate in Southern California, where snow is seldom a problem, or in Alaska, where it seldom is not. It could always be driving on the right, or we might want it to be flexible enough to drive on the left when in Britain or Japan. Obviously, the more restricted the environment, the easier the design problem.

The **actuators** available to an automated taxi will be more or less the same as those available to a human driver: control over the engine through the accelerator and control over steering and braking. In addition, it will need output to a display screen or voice synthesizer to talk back to the passengers, and perhaps some way to communicate with other vehicles, politely or otherwise.

To achieve its goals in the driving environment, the taxi will need to know where it is, what else is on the road, and how fast it is going. Its basic **sensors** should therefore include one or more controllable TV cameras, the speedometer, and the odometer. To control the vehicle properly, especially on curves, it should have an accelerometer; it will also need to know the mechanical state of the vehicle, so it will need the usual array of engine and electrical system sensors. It might have instruments that are not available to the average human driver: a satellite global positioning system (GPS) to give it accurate position information with respect to an electronic map, and infrared or sonar sensors to detect distances to other cars and obstacles. Finally, it will need a keyboard or microphone for the passenger to request a destination.

In Figure 2.5, we have sketched the basic PEAS elements for a number of additional agent types. Further examples appear in Exercise 2.5. It may come as a surprise to some readers that we include in our list of agent types some programs that operate in the entirely artificial environment defined by keyboard input and character output on a screen. "Surely," one might say, "this is not a real environment, is it?" In fact, what matters is not the distinction between "real" and "artificial" environments, but the complexity of the relationship among the behavior of the agent, the percept sequence generated by the environment, and the performance measure. Some "real" environments are actually quite simple. For example, a robot designed to inspect parts as they come by on a conveyor belt can make use of a number of simplifying assumptions: that the lighting is always just so, that the only thing on the conveyor belt will be parts of a kind that it knows about, and that there are only two actions (accept or reject).

Agent Type	Performance Measure	Environment	Actuators	Sensors
Medical diagnosis system	Healthy patient, minimize costs, lawsuits	Patient, hospital, staff	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers
Satellite image analysis system	Correct image categorization	Downlink from orbiting satellite	Display categorization of scene	Color pixel arrays
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arm and hand	Camera, joint angle sensors
Refinery controller	Maximize purity, yield, safety	Refinery, operators	Valves, pumps, heaters, displays	Temperature, pressure, chemical sensors
Interactive English tutor	Maximize student's score on test	Set of students, testing agency	Display exercises, suggestions, corrections	Keyboard entry

**Figure 2.5** Examples of agent types and their PEAS descriptions.

#### SOFTWARE AGENTS SOFTBOTS

In contrast, some **software agents** (or software robots or **softbots**) exist in rich, unlimited domains. Imagine a softbot designed to fly a flight simulator for a large commercial airplane. The simulator is a very detailed, complex environment including other aircraft and ground operations, and the software agent must choose from a wide variety of actions in real time. Or imagine a softbot designed to scan Internet news sources and show the interesting items to its customers. To do well, it will need some natural language processing abilities, it will need to learn what each customer is interested in, and it will need to change its plans dynamically—for example, when the connection for one news source goes down or when a new one comes online. The Internet is an environment whose complexity rivals that of the physical world and whose inhabitants include many artificial agents.

### Properties of task environments

The range of task environments that might arise in AI is obviously vast. We can, however, identify a fairly small number of dimensions along which task environments can be categorized. These dimensions determine, to a large extent, the appropriate agent design and the

applicability of each of the principal families of techniques for agent implementation. First, we list the dimensions, then we analyze several task environments to illustrate the ideas. The definitions here are informal; later chapters provide more precise statements and examples of each kind of environment.

FULLY OBSERVABLE

◇ **Fully observable vs. partially observable.**

If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.<sup>4</sup> A task environment is effectively fully observable if the sensors detect all aspects that are *relevant* to the choice of action; relevance, in turn, depends on the performance measure. Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world. An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.

DETERMINISTIC

STOCHASTIC

◇ **Deterministic vs. stochastic.**

If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic. In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment. If the environment is partially observable, however, then it could *appear* to be stochastic. This is particularly true if the environment is complex, making it hard to keep track of all the unobserved aspects. Thus, it is often better to think of an environment as deterministic or stochastic *from the point of view of the agent*. Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning. The vacuum world as we described it is deterministic, but variations can include stochastic elements such as randomly appearing dirt and an unreliable suction mechanism (Exercise 2.12). If the environment is deterministic except for the actions of other agents, we say that the environment is **strategic**.

STRATEGIC

EPISODIC

SEQUENTIAL

◇ **Episodic vs. sequential.**<sup>5</sup>

In an episodic task environment, the agent's experience is divided into atomic episodes. Each episode consists of the agent perceiving and then performing a single action. Crucially, the next episode does not depend on the actions taken in previous episodes. In episodic environments, the choice of action in each episode depends only on the episode itself. Many classification tasks are episodic. For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions; moreover, the current decision doesn't affect whether the next

<sup>4</sup> The first edition of this book used the terms **accessible** and **inaccessible** instead of **fully** and **partially observable**; **nondeterministic** instead of **stochastic**; and **nonepisodic** instead of **sequential**. The new terminology is more consistent with established usage.

<sup>5</sup> The word "sequential" is also used in computer science as the antonym of "parallel." The two meanings are largely unrelated.



part is defective. In sequential environments, on the other hand, the current decision could affect all future decisions. Chess and taxi driving are sequential: in both cases, short-term actions can have long-term consequences. Episodic environments are much simpler than sequential environments because the agent does not need to think ahead.

STATIC  
DYNAMIC

◇ **Static vs. dynamic.**

If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static. Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time. Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing. If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is **semidynamic**. Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess, when played with a clock, is semidynamic. Crossword puzzles are static.

SEMIDYNAMIC

DISCRETE  
CONTINUOUS

◇ **Discrete vs. continuous.**

The discrete/continuous distinction can be applied to the *state* of the environment, to the way *time* is handled, and to the *percepts* and *actions* of the agent. For example, a discrete-state environment such as a chess game has a finite number of distinct states. Chess also has a discrete set of percepts and actions. Taxi driving is a **continuous-state** and **continuous-time** problem: the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time. Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

SINGLE AGENT  
MULTIAGENT

◇ **Single agent vs. multiagent.**

The distinction between single-agent and multiagent environments may seem simple enough. For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two-agent environment. There are, however, some subtle issues. First, we have described how an entity *may* be viewed as an agent, but we have not explained which entities *must* be viewed as agents. Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated merely as a stochastically behaving object, analogous to waves at the beach or leaves blowing in the wind? The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior. For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess is a **competitive** multiagent environment. In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially **cooperative** multiagent environment. It is also partially competitive because, for example, only one car can occupy a parking space. The agent-design problems arising in multiagent environments are often

COMPETITIVE

COOPERATIVE

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Chess with a clock	Fully	Strategic	Sequential	Semi	Discrete	Multi
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Medical diagnosis	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Image-analysis	Fully	Deterministic	Episodic	Semi	Continuous	Single
Part-picking robot	Partially	Stochastic	Episodic	Dynamic	Continuous	Single
Refinery controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi

**Figure 2.6** Examples of task environments and their characteristics.

quite different from those in single-agent environments; for example, **communication** often emerges as a rational behavior in multiagent environments; in some partially observable competitive environments, **stochastic behavior** is rational because it avoids the pitfalls of predictability.

As one might expect, the hardest case is *partially observable, stochastic, sequential, dynamic, continuous, and multiagent*. It also turns out that most real situations are so complex that whether they are *really* deterministic is a moot point. For practical purposes, they must be treated as stochastic. Taxi driving is hard in all these senses.

Figure 2.6 lists the properties of a number of familiar environments. Note that the answers are not always cut and dried. For example, we have listed chess as fully observable; strictly speaking, this is false because certain rules about castling, en *passant* capture, and draws by repetition require remembering some facts about the game history that are not observable as part of the board state. These exceptions to observability are of course minor compared to those faced by the taxi driver, the English tutor, or the medical diagnosis system.

Some other answers in the table depend on how the task environment is defined. We have listed the medical-diagnosis task as single-agent because the disease process in a patient is not profitably modeled as an agent; but a medical-diagnosis system might also have to deal with recalcitrant patients and skeptical staff, so the environment could have a multiagent aspect. Furthermore, medical diagnosis is episodic if one conceives of the task as selecting a diagnosis given a list of symptoms; the problem is sequential if the task can include proposing a series of tests, evaluating progress over the course of treatment, and so on. Also, many environments are episodic at higher levels than the agent's individual actions. For example, a chess tournament consists of a sequence of games; each game is an episode, because (by and large) the contribution of the moves in one game to the agent's overall performance is not affected by the moves in its previous game. On the other hand, decision making within a single game is certainly sequential.

ENVIRONMENT  
CLASSENVIRONMENT  
GENERATOR

The code repository associated with this book ([aima.cs.berkeley.edu](http://aima.cs.berkeley.edu)) includes implementations of a number of environments, together with a general-purpose environment simulator that places one or more agents in a simulated environment, observes their behavior over time, and evaluates them according to a given performance measure. Such experiments are often carried out not for a single environment, but for many environments drawn from an **environment class**. For example, to evaluate a taxi driver in simulated traffic, we would want to run many simulations with different traffic, lighting, and weather conditions. If we designed the agent for a single scenario, we might be able to take advantage of specific properties of the particular case but might not identify a good design for driving in general. For this reason, the code repository also includes an **environment generator** for each environment class that selects particular environments (with certain likelihoods) in which to run the agent. For example, the vacuum environment generator initializes the dirt pattern and agent location randomly. We are then interested in the agent's average performance over the environment class. A rational agent for a given environment class maximizes this average performance. Exercises 2.7 to 2.12 take you through the process of developing an environment class and evaluating various agents therein.

## 2.4 THE STRUCTURE OF AGENTS

AGENTPROGRAM

ARCHITECTURE

So far we have talked about agents by describing behavior—the action that is performed after any given sequence of percepts. Now, we will have to bite the bullet and talk about how the insides work. The job of AI is to design the **agent program** that implements the agent function mapping percepts to actions. We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the **architecture**:

agent = architecture + program .

Obviously, the program we choose has to be one that is appropriate for the architecture. If the program is going to recommend actions like Walk, the architecture had better have legs. The architecture might be just an ordinary PC, or it might be a robotic car with several onboard computers, cameras, and other sensors. In general, the architecture makes the percepts from the sensors available to the program, runs the program, and feeds the program's action choices to the actuators as they are generated. Most of this book is about designing agent programs, although Chapters 24 and 25 deal directly with the sensors and actuators.

### Agent programs

The agent programs that we will design in this book all have the same skeleton: they take the current percept as input from the sensors and return an action to the actuators.<sup>6</sup> Notice the difference between the agent program, which takes the current percept as input, and the agent function, which takes the entire percept history. The agent program takes just the current

<sup>6</sup> There are other choices for the agent program skeleton; for example, we could have the agent programs be **coroutines** that run asynchronously with the environment. Each such coroutine has an input and output port and consists of a loop that reads the input port for percepts and writes actions to the output port.

```

function TABLE-DRIVEN-AGENT(percept) returns an action
  static: percepts, a sequence, initially empty
           table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action  $\leftarrow$  LOOKUP(percepts, table)
  return action

```

**Figure 2.7** The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It keeps track of the percept sequence using its own private data structure.

percept as input because nothing more is available from the environment; if the agent's actions depend on the entire percept sequence, the agent will have to remember the percepts.

We will describe the agent programs via the simple pseudocode language that is defined in Appendix B. (The online code repository contains implementations in real programming languages.) For example, Figure 2.7 shows a rather trivial agent program that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do. The table represents explicitly the agent function that the agent program embodies. To build a rational agent in this way, we as designers must construct a table that contains the appropriate action for every possible percept sequence.

It is instructive to consider why the table-driven approach to agent construction is doomed to failure. Let  $\mathcal{P}$  be the set of possible percepts and let  $T$  be the lifetime of the agent (the total number of percepts it will receive). The lookup table will contain  $\sum_{t=1}^T |\mathcal{P}|^t$  entries. Consider the automated taxi: the visual input from a single camera comes in at the rate of roughly 27 megabytes per second (30 frames per second, 640 x 480 pixels with 24 bits of color information). This gives a lookup table with over  $10^{250,000,000,000}$  entries for an hour's driving. Even the lookup table for chess—a tiny, well-behaved fragment of the real world—would have at least  $10^{150}$  entries. The daunting size of these tables (the number of atoms in the observable universe is less than  $10^{80}$ ) means that (a) no physical agent in this universe will have the space to store the table, (b) the designer would not have time to create the table, (c) no agent could ever learn all the right table entries from its experience, and (d) even if the environment is simple enough to yield a feasible table size, the designer still has no guidance about how to fill in the table entries.

Despite all this, TABLE-DRIVEN-AGENT *does* do what we want: it implements the desired agent function. The key challenge for AI is to find out how to write programs that, to the extent possible, produce rational behavior from a small amount of code rather than from a large number of table entries. We have many examples showing that this can be done successfully in other areas: for example, the huge tables of square roots used by engineers and schoolchildren prior to the 1970s have now been replaced by a five-line program for Newton's method running on electronic calculators. The question is, can AI do for general intelligent behavior what Newton did for square roots? We believe the answer is yes.

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
```

```
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

**Figure 2.8** The agent program for a simple reflex agent in the two-state vacuum environment. This program implements the agent function tabulated in Figure 2.3.

In the remainder of this section, we outline four basic kinds of agent program that embody the principles underlying almost all intelligent systems:

- Simple reflex agents;
- Model-based reflex agents;
- Goal-based agents; and
- Utility-based agents.

We then explain in general terms how to convert all these into *learning agents*.

### Simple reflex agents

SIMPLE  
AGENT

The simplest kind of agent is the **simple reflex agent**. These agents select actions on the basis of the *current* percept, ignoring the rest of the percept history. For example, the vacuum agent whose agent function is tabulated in Figure 2.3 is a simple reflex agent, because its decision is based only on the current location and on whether that contains dirt. An agent program for this agent is shown in Figure 2.8.

Notice that the vacuum agent program is very small indeed compared to the corresponding table. The most obvious reduction comes from ignoring the percept history, which cuts down the number of possibilities from  $4^T$  to just 4. A further, small reduction comes from the fact that, when the current square is dirty, the action does not depend on the location.

Imagine yourself as the driver of the automated taxi. If the car in front brakes, and its brake lights come on, then you should notice this and initiate braking. In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking." Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a **condition–action rule**: written as

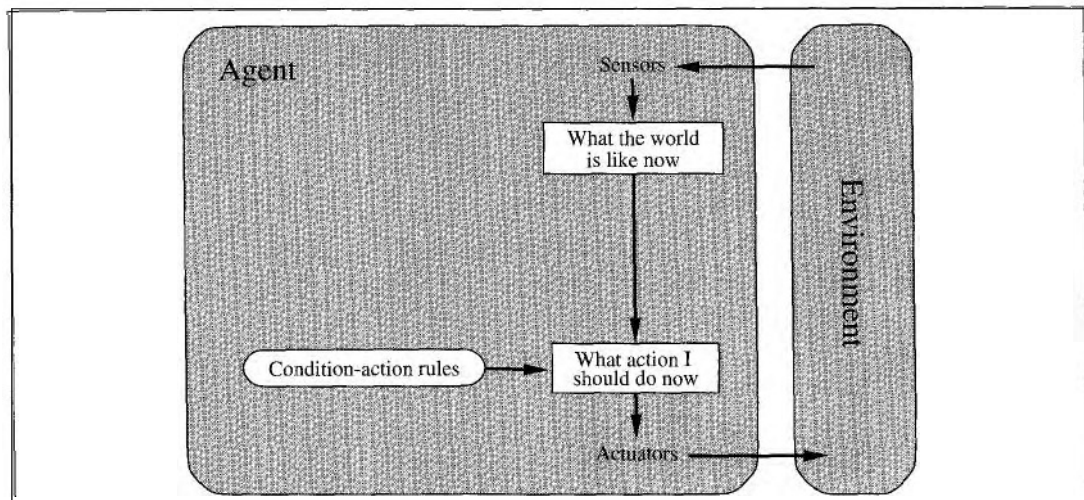
CONDITION–ACTION  
RULE

**if** *car-in-front-is-braking* **then** *initiate-braking*.

Humans also have many such connections, some of which are learned responses (as for driving) and some of which are innate reflexes (such as blinking when something approaches the eye). In the course of the book, we will see several different ways in which such connections can be learned and implemented.

The program in Figure 2.8 is specific to one particular vacuum environment. A more general and flexible approach is first to build a general-purpose interpreter for condition–

<sup>7</sup> Also called **situation–action rules**, **productions**, or **if–then rules**.



**Figure 2.9** Schematic diagram of a simple reflex agent.

**function** SIMPLE-REFLEX-AGENT(*percept*) **returns** an action

**static:** *rules*, a set of condition–action rules

*state*  $\leftarrow$  INTERPRET-INPUT(*percept*)

*rule*  $\leftarrow$  RULE-MATCH(*state*, *rules*)

*action*  $\leftarrow$  RULE-ACTION[*rule*]

**return** *action*

**Figure 2.10** A simple reflex agent. It acts according to a rule whose condition matches the current state, as defined by the percept.

action rules and then to create rule sets for specific task environments. Figure 2.9 gives the structure of this general program in schematic form, showing how the condition–action rules allow the agent to make the connection from percept to action. (Do not worry if this seems trivial; it gets more interesting shortly.) We use rectangles to denote the current internal state of the agent's decision process and ovals to represent the background information used in the process. The agent program, which is also very simple, is shown in Figure 2.10. The INTERPRET-INPUT function generates an abstracted description of the current state from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description. Note that the description in terms of "rules" and "matching" is purely conceptual; actual implementations can be as simple as a collection of logic gates implementing a Boolean circuit.



Simple reflex agents have the admirable property of being simple, but they turn out to be of very limited intelligence. The agent in Figure 2.10 will work *only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable*. Even a little bit of unobservability can cause serious trouble. For example,

the braking rule given earlier assumes that the condition *car-in-front-is-braking* can be determined from the current percept—the current video image—if the car in front has a centrally mounted brake light. Unfortunately, older models have different configurations of taillights, brake lights, and turn-signal lights, and it is not always possible to tell from a single image whether the car is braking. A simple reflex agent driving behind such a car would either brake continuously and unnecessarily, or, worse, never brake at all.

We can see a similar problem arising in the vacuum world. Suppose that a simple reflex vacuum agent is deprived of its location sensor, and has only a dirt sensor. Such an agent has just two possible percepts: *[Dirty]* and *[Clean]*. It can *Suck* in response to *[Dirty]*; what should it do in response to *[Clean]*? Moving *Left* fails (for ever) if it happens to start in square A, and moving *Right* fails (for ever) if it happens to start in square B. Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments.

RANDOMIZATION

Escape from infinite loops is possible if the agent can **randomize** its actions. For example, if the vacuum agent perceives *[Clean]* it might flip a coin to choose between *Left* and *Right*. It is easy to show that the agent will reach the other square in an average of two steps. Then, if that square is dirty, it will clean it and the cleaning task will be complete. Hence, a randomized simple reflex agent might outperform a deterministic simple reflex agent.

We mentioned in Section 2.3 that randomized behavior of the right kind can be rational in some multiagent environments. In single-agent environments, randomization is usually *not* rational. It is a useful trick that helps a simple reflex agent in some situations, but in most cases we can do much better with more sophisticated deterministic agents.

### Model-based reflex agents

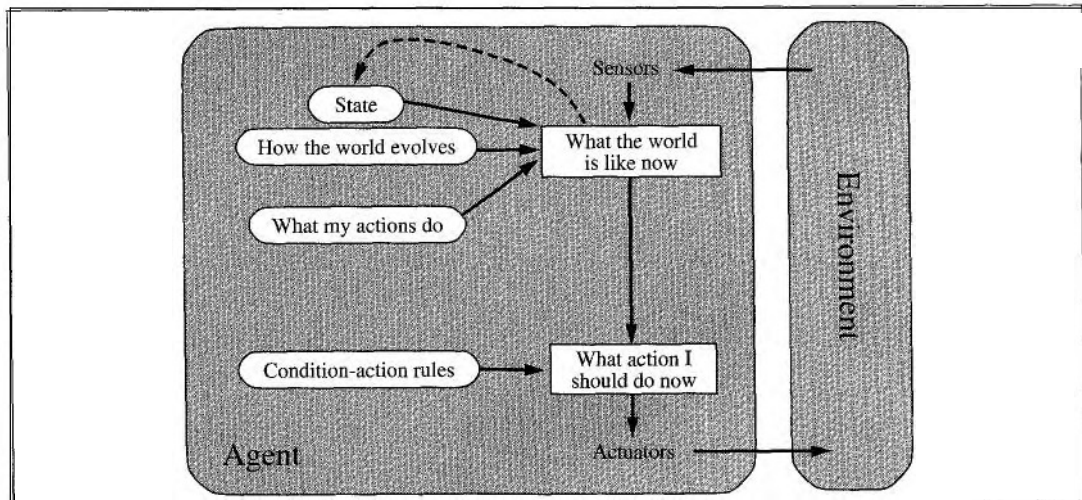
INTERNAL STATE

The most effective way to handle partial observability is for the agent to *keep track of the part of the world it can't see now*. That is, the agent should maintain some sort of **internal state** that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state. For the braking problem, the internal state is not too extensive—just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. For other driving tasks such as changing lanes, the agent needs to keep track of where the other cars are if it can't see them all at once.

Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program. First, we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago. Second, we need some information about how the agent's own actions affect the world—for example, that when the agent turns the steering wheel clockwise, the car turns to the right or that after driving for five minutes northbound on the freeway one is usually about five miles north of where one was five minutes ago. This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a **model** of the world. An agent that uses such a model is called a **model-based agent**.

MODEL-BASED  
AGENT

Figure 2.11 gives the structure of the reflex agent with internal state, showing how the current percept is combined with the old internal state to generate the updated description



**Figure 2.11** A model-based reflex agent.

**function** REFLEX-AGENT-WITH-STATE(*percept*) **returns** an action

**static:** *state*, a description of the current world state

*rules*, a set of condition-action rules

*action*, the most recent action, initially none

*state* ← UPDATE-STATE(*state*, *action*, *percept*)

*rule* ← RULE-MATCH(*state*, *rules*)

*action* ← RULE-ACTION[*rule*]

**return** *action*

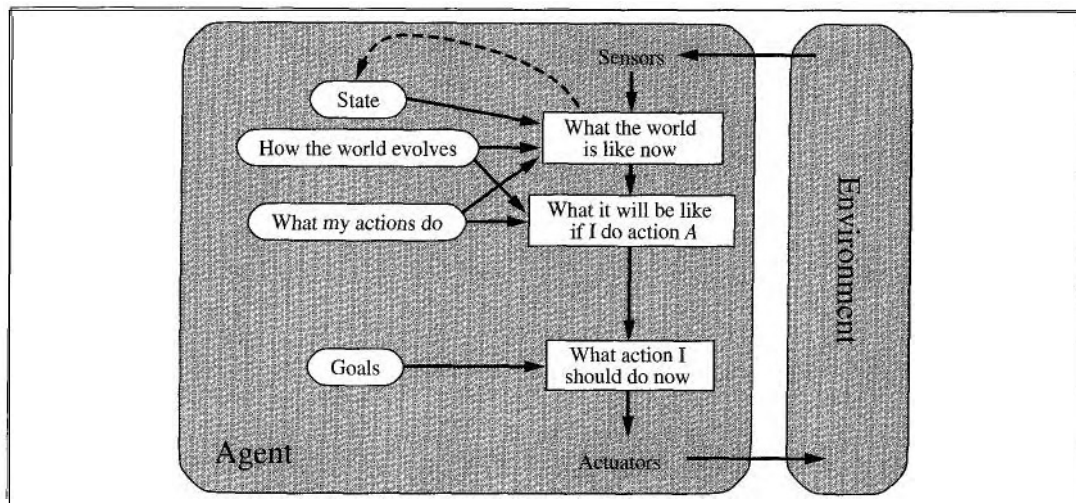
**Figure 2.12** A model-based reflex agent. It keeps track of the current state of the world using an internal model. It then chooses an action in the same way as the reflex agent.

of the current state. The agent program is shown in Figure 2.12. The interesting part is the function UPDATE-STATE, which is responsible for creating the new internal state description. As well as interpreting the new percept in the light of existing knowledge about the state, it uses information about how the world evolves to keep track of the unseen parts of the world, and also must know about what the agent's actions do to the state of the world. Detailed examples appear in Chapters 10 and 17.

### Goal-based agents

Knowing about the current state of the environment is not always enough to decide what to do. For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to. In other words, as well as a current state description, the agent needs some sort of **goal** information that describes situations that are desirable—for example, being at the passenger's destination. The agent





**Figure 2.13** A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

program can combine this with information about the results of possible actions (the same information as was used to update internal state in the reflex agent) in order to choose actions that achieve the goal. Figure 2.13 shows the goal-based agent's structure.

Sometimes goal-based action selection is straightforward, when goal satisfaction results immediately from a single action. Sometimes it will be more tricky, when the agent has to consider long sequences of twists and turns to find a way to achieve the goal. **Search** (Chapters 3 to 6) and **planning** (Chapters 11 and 12) are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

Notice that decision making of this kind is fundamentally different from the condition–action rules described earlier, in that it involves consideration of the future—both “What will happen if I do such-and-such?” and “Will that make me happy?” In the reflex agent designs, this information is not explicitly represented, because the built-in rules map directly from percepts to actions. The reflex agent brakes when it sees brake lights. A goal-based agent, in principle, could reason that if the car in front has its brake lights on, it will slow down. Given the way the world usually evolves, the only action that will achieve the goal of not hitting other cars is to brake.

Although the goal-based agent appears less efficient, it is more flexible because the knowledge that supports its decisions is represented explicitly and can be modified. If it starts to rain, the agent can update its knowledge of how effectively its brakes will operate; this will automatically cause all of the relevant behaviors to be altered to suit the new conditions. For the reflex agent, on the other hand, we would have to rewrite many condition–action rules. The goal-based agent's behavior can easily be changed to go to a different location. The reflex agent's rules for when to turn and when to go straight will work only for a single destination; they must all be replaced to go somewhere new.

## Utility-based agents

Goals alone are not really enough to generate high-quality behavior in most environments. For example, there are many action sequences that will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others. Goals just provide a crude binary distinction between "happy" and "unhappy" states, whereas a more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent if they could be achieved. Because "happy" does not sound very scientific, the customary terminology is to say that if one world state is preferred to another, then it has higher **utility** for the agent.<sup>8</sup>

UTILITY

UTILITY FUNCTION

A **utility function** maps a state (or a sequence of states) onto a real number, which describes the associated degree of happiness. A complete specification of the utility function allows rational decisions in two kinds of cases where goals are inadequate. First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff. Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed up against the importance of the goals.

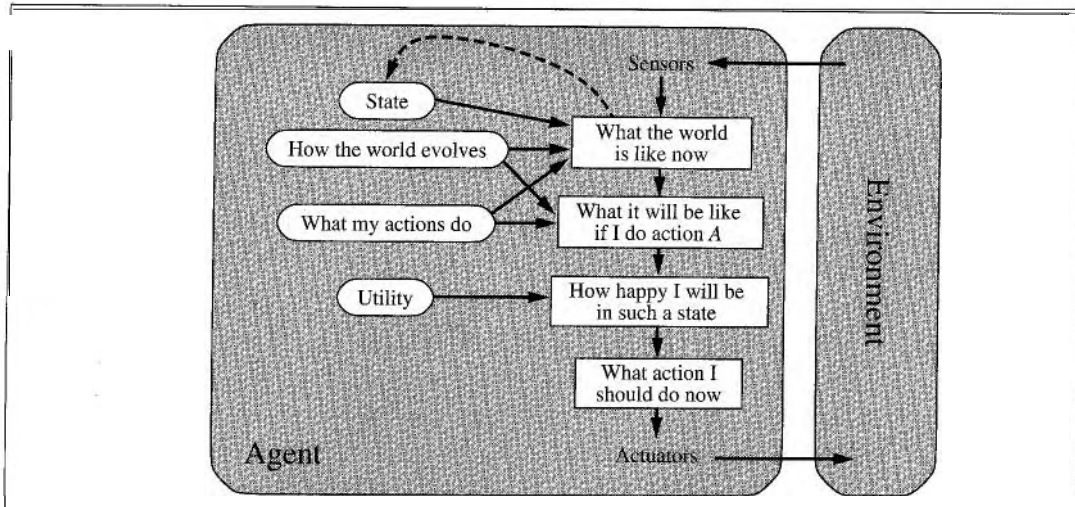
In Chapter 16, we will show that any rational agent must behave as *if* it possesses a utility function whose expected value it tries to maximize. An agent that possesses an *explicit* utility function therefore can make rational decisions, and it can do so via a general-purpose algorithm that does not depend on the specific utility function being maximized. In this way, the "global" definition of rationality—designating as rational those agent functions that have the highest performance—is turned into a "local" constraint on rational-agent designs that can be expressed in a simple program.

The utility-based agent structure appears in Figure 2.14. Utility-based agent programs appear in Part V, where we design decision making agents that must handle the uncertainty inherent in partially observable environments.

## Learning agents

We have described agent programs with various methods for selecting actions. We have not, so far, explained how the agent programs *come into being*. In his famous early paper, Turing (1950) considers the idea of actually programming his intelligent machines by hand. He estimates how much work this might take and concludes "Some more expeditious method seems desirable." The method he proposes is to build learning machines and then to teach them. In many areas of AI, this is now the preferred method for creating state-of-the-art systems. Learning has another advantage, as we noted earlier: it allows the agent to operate in initially unknown environments and to become more competent than its initial knowledge alone might allow. In this section, we briefly introduce the main ideas of learning agents. In almost every chapter of the book, we will comment on opportunities and methods for learning in particular kinds of agents. Part VI goes into much more depth on the various learning algorithms themselves.

<sup>8</sup> The word "utility" here refers to "the quality of being useful," not to the electric company or water works.



**Figure 2.14** A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome states, weighted by the probability of the outcome.

LEARNING ELEMENT  
PERFORMANCE  
ELEMENT

CRITIC

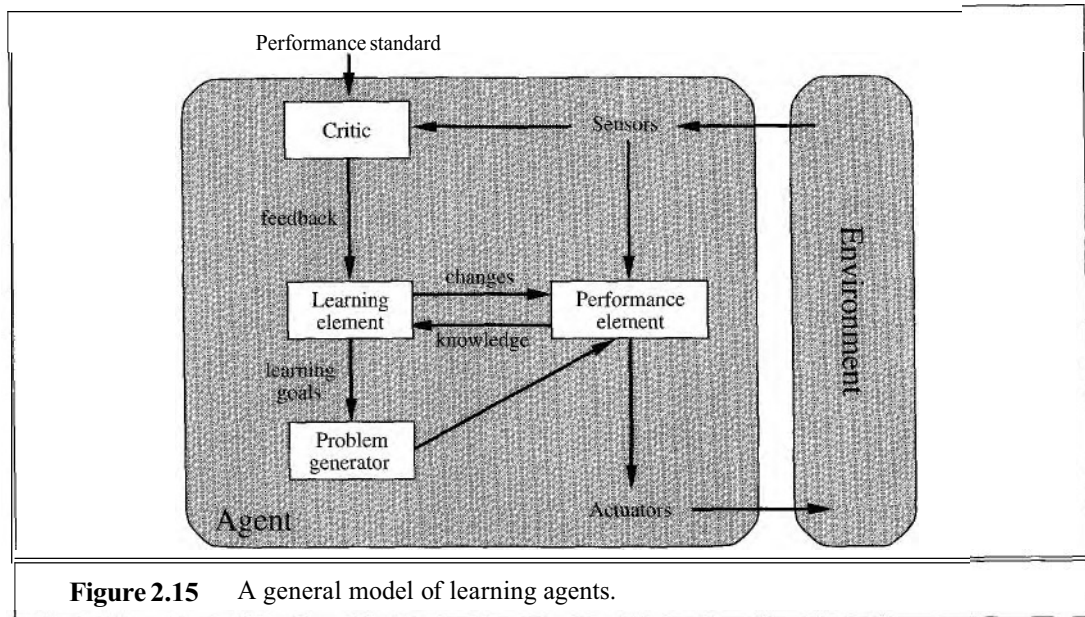
A learning agent can be divided into four conceptual components, as shown in Figure 2.15. The most important distinction is between the learning element, which is responsible for making improvements, and the performance element, which is responsible for selecting external actions. The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions. The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.

The design of the learning element depends very much on the design of the performance element. When trying to design an agent that learns a certain capability, the first question is not "How am I going to get it to learn this?" but "What kind of performance element will my agent need to do this once it has learned how?" Given an agent design, learning mechanisms can be constructed to improve every part of the agent.

The critic tells the learning element how well the agent is doing with respect to a fixed performance standard. The critic is necessary because the percepts themselves provide no indication of the agent's success. For example, a chess program could receive a percept indicating that it has checkmated its opponent, but it needs a performance standard to know that this is a good thing; the percept itself does not say so. It is important that the performance standard be fixed. Conceptually, one should think of it as being outside the agent altogether, because the agent must not modify it to fit its own behavior.

PROBLEM  
GENERATOR

The last component of the learning agent is the problem generator. It is responsible for suggesting actions that will lead to new and informative experiences. The point is that if the performance element had its way, it would keep doing the actions that are best, given what it knows. But if the agent is willing to explore a little, and do some perhaps suboptimal actions in the short run, it might discover much better actions for the long run. The problem



**Figure 2.15** A general model of learning agents.

generator's job is to suggest these exploratory actions. This is what scientists do when they carry out experiments. Galileo did not think that dropping rocks from the top of a tower in Pisa was valuable in itself. He was not trying to break the rocks, nor to modify the brains of unfortunate passers-by. His aim was to modify his own brain, by identifying a better theory of the motion of objects.

To make the overall design more concrete, let us return to the automated taxi example. The performance element consists of whatever collection of knowledge and procedures the taxi has for selecting its driving actions. The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes a quick left turn across three lanes of traffic, the critic observes the shocking language used by other drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installing the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions.

The learning element can make changes to any of the "knowledge" components shown in the agent diagrams (Figures 2.9, 2.11, 2.13, and 2.14). The simplest cases involve learning directly from the percept sequence. Observation of pairs of successive states of the environment can allow the agent to learn "How the world evolves," and observation of the results of its actions can allow the agent to learn "What my actions do." For example, if the taxi exerts a certain braking pressure when driving on a wet road, then it will soon find out how much deceleration is actually achieved. Clearly, these two learning tasks are more difficult if the environment is only partially observable.

The forms of learning in the preceding paragraph do not need to access the external performance standard—in a sense, the standard is the universal one of making predictions

that agree with experiment. The situation is slightly more complex for a utility-based agent that wishes to learn utility information. For example, suppose the taxi-driving agent receives no tips from passengers who have been thoroughly shaken up during the trip. The external performance standard must inform the agent that the loss of tips is a negative contribution to its overall performance; then the agent might be able to learn that violent maneuvers do not contribute to its own utility. In a sense, the performance standard distinguishes part of the incoming percept as a **reward** (or **penalty**) that provides direct feedback on the quality of the agent's behavior. Hard-wired performance standards such as pain and hunger in animals can be understood in this way. This issue is discussed further in Chapter 21.

In summary, agents have a variety of components, and those components can be represented in many ways within the agent program, so there appears to be great variety among learning methods. There is, however, a single unifying theme. Learning in intelligent agents can be summarized as a process of modification of each component of the agent to bring the components into closer agreement with the available feedback information, thereby improving the overall performance of the agent.

---

## 2.5 SUMMARY

---

This chapter has been something of a whirlwind tour of AI, which we have conceived of as the science of agent design. The major points to recall are as follows:

- An **agent** is something that perceives and acts in an environment. The **agent function** for an agent specifies the action taken by the agent in response to any percept sequence.
- The **performance measure** evaluates the behavior of the agent in an environment. A **rational agent** acts so as to maximize the expected value of the performance measure, given the percept sequence it has seen so far.
- A **task environment** specification includes the performance measure, the external environment, the actuators, and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Task environments vary along several significant dimensions. They can be fully or partially observable, deterministic or stochastic, episodic or sequential, static or dynamic, discrete or continuous, and single-agent or multiagent.
- The **agent program** implements the agent function. There exists a variety of basic agent-program designs, reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency, compactness, and flexibility. The appropriate design of the agent program depends on the nature of the environment.
- **Simple reflex agents** respond directly to percepts, whereas **model-based reflex agents** maintain internal state to track aspects of the world that are not evident in the current percept. **Goal-based agents** act to achieve their goals, and **utility-based agents** try to maximize their own expected "happiness."
- All agents can improve their performance through **learning**.

---

BIBLIOGRAPHICAL AND HISTORICAL NOTES

CONTROLLER

The central role of action in intelligence—the notion of practical reasoning—goes back at least as far as Aristotle's *Nicomachean Ethics*. Practical reasoning was also the subject of McCarthy's (1958) influential paper "Programs with Common Sense." The fields of robotics and control theory are, by their very nature, concerned principally with the construction of physical agents. The concept of a **controller** in control theory is identical to that of an agent in AI. Perhaps surprisingly, AI has concentrated for most of its history on isolated components of agents—question-answering systems, theorem-provers, vision systems, and so on—rather than on whole agents. The discussion of agents in the text by Genesereth and Nilsson (1987) was an influential exception. The whole-agent view is now widely accepted in the field and is a central theme in recent texts (Poole *et al.*, 1998; Nilsson, 1998).

Chapter 1 traced the roots of the concept of rationality in philosophy and economics. In AI, the concept was of peripheral interest until the mid-1980s, when it began to suffuse many discussions about the proper technical foundations of the field. A paper by Jon Doyle (1983) predicted that rational agent design would come to be seen as the core mission of AI, while other popular topics would spin off to form new disciplines.

Careful attention to the properties of the environment and their consequences for rational agent design is most apparent in the control theory tradition—for example, classical control systems (Dorf and Bishop, 1999) handle fully observable, deterministic environments; stochastic optimal control (Kumar and Varaiya, 1986) handles partially observable, stochastic environments; and hybrid control (Henzinger and Sastry, 1998) deals with environments containing both discrete and continuous elements. The distinction between fully and partially observable environments is also central in the **dynamic programming** literature developed in the field of operations research (Puterman, 1994), which we will discuss in Chapter 17.

Reflex agents were the primary model for psychological behaviorists such as Skinner (1953), who attempted to reduce the psychology of organisms strictly to input/output or stimulus/response mappings. The advance from behaviorism to functionalism in psychology, which was at least partly driven by the application of the computer metaphor to agents (Putnam, 1960; Lewis, 1966), introduced the internal state of the agent into the picture. Most work in AI views the idea of pure reflex agents with state as too simple to provide much leverage, but work by Rosenschein (1985) and Brooks (1986) questioned this assumption (see Chapter 25). In recent years, a great deal of work has gone into finding efficient algorithms for keeping track of complex environments (Hamscher *et al.*, 1992). The Remote Agent program that controlled the Deep Space One spacecraft (described on page 27) is a particularly impressive example (Muscettola *et al.*, 1998; Jonsson *et al.*, 2000).

Goal-based agents are presupposed in everything from Aristotle's view of practical reasoning to McCarthy's early papers on logical AI. Shakey the Robot (Fikes and Nilsson, 1971; Nilsson, 1984) was the first robotic embodiment of a logical, goal-based agent. A full logical analysis of goal-based agents appeared in Genesereth and Nilsson (1987), and a goal-based programming methodology called agent-oriented programming was developed by Shoham (1993).

The goal-based view also dominates the cognitive psychology tradition in the area of problem solving, beginning with the enormously influential *Human Problem Solving* (Newell and Simon, 1972) and running through all of Newell's later work (Newell, 1990). Goals, further analyzed as *desires* (general) and *intentions* (currently pursued), are central to the theory of agents developed by Bratman (1987). This theory has been influential both in natural language understanding and multiagent systems.

Horvitz *et al.* (1988) specifically suggest the use of rationality conceived as the maximization of expected utility as a basis for AI. The text by Pearl (1988) was the first in AI to cover probability and utility theory in depth; its exposition of practical methods for reasoning and decision making under uncertainty was probably the single biggest factor in the rapid shift towards utility-based agents in the 1990s (see Part V).


The general design for learning agents portrayed in Figure 2.15 is classic in the machine learning literature (Buchanan *et al.*, 1978; Mitchell, 1997). Examples of the design, as embodied in programs, go back at least as far as Arthur Samuel's (1959, 1967) learning program for playing checkers. Learning agents are discussed in depth in Part VI.

Interest in agents and in agent design has risen rapidly in recent years, partly because of the growth of the Internet and the perceived need for automated and mobile **softbots** (Etzioni and Weld, 1994). Relevant papers are collected in *Readings in Agents* (Huhns and Singh, 1998) and *Foundations of Rational Agency* (Wooldridge and Rao, 1999). *Multiagent Systems* (Weiss, 1999) provides a solid foundation for many aspects of agent design. Conferences devoted to agents include the International Conference on Autonomous Agents, the International Workshop on Agent Theories, Architectures, and Languages, and the International Conference on Multiagent Systems. Finally, *Dung Beetle Ecology* (Hanski and Cambefort, 1991) provides a wealth of interesting information on the behavior of dung beetles.

---

## EXERCISES

- 2.1** Define in your own words the following terms: agent, agent function, agent program, rationality, autonomy, reflex agent, model-based agent, goal-based agent, utility-based agent, learning agent.
- 2.2** Both the performance measure and the utility function measure how well an agent is doing. Explain the difference between the two.
- 2.3** This exercise explores the differences between agent functions and agent programs.
  - a.** Can there be more than one agent program that implements a given agent function? Give an example, or show why one is not possible.
  - b.** Are there agent functions that cannot be implemented by any agent program?
  - c.** Given a fixed machine architecture, does each agent program implement exactly one agent function?
  - d.** Given an architecture with  $n$  bits of storage, how many different possible agent programs are there?

- 2.4** Let us examine the rationality of various vacuum-cleaner agent functions.
- a. Show that the simple vacuum-cleaner agent function described in Figure 2.3 is indeed rational under the assumptions listed on page 36.
  - b. Describe a rational agent function for the modified performance measure that deducts one point for each movement. Does the corresponding agent program require internal state?
  - c. Discuss possible agent designs for the cases in which clean squares can become dirty and the geography of the environment is unknown. Does it make sense for the agent to learn from its experience in these cases? If so, what should it learn?
- 2.5** For each of the following agents, develop a PEAS description of the task environment:
- a. Robot soccer player;
  - b. Internet book-shopping agent;
  - c. Autonomous Mars rover;
  - d. Mathematician's theorem-proving assistant.
- 2.6** For each of the agent types listed in Exercise 2.5, characterize the environment according to the properties given in Section 2.3, and select a suitable agent design.
-  The following exercises all concern the implementation of environments and agents for the vacuum-cleaner world.
- 2.7** Implement a performance-measuring environment simulator for the vacuum-cleaner world depicted in Figure 2.2 and specified on page 36. Your implementation should be modular, so that the sensors, actuators, and environment characteristics (size, shape, dirt placement, etc.) can be changed easily. (*Note:* for some choices of programming language and operating system there are already implementations in the online code repository.)
- 2.8** Implement a simple reflex agent for the vacuum environment in Exercise 2.7. Run the environment simulator with this agent for all possible initial dirt configurations and agent locations. Record the agent's performance score for each configuration and its overall average score.
- 2.9** Consider a modified version of the vacuum environment in Exercise 2.7, in which the agent is penalized one point for each movement.
- a. Can a simple reflex agent be perfectly rational for this environment? Explain.
  - b. What about a reflex agent with state? Design such an agent.
  - c. How do your answers to **a** and **b** change if the agent's percepts give it the clean/dirty status of every square in the environment?
- 2.10** Consider a modified version of the vacuum environment in Exercise 2.7, in which the geography of the environment—its extent, boundaries, and obstacles—is unknown, as is the initial dirt configuration. (The agent can go *Up* and *Down* as well as *Left* and *Right*.)
- a. Can a simple reflex agent be perfectly rational for this environment? Explain.



- b.** Can a simple reflex agent with a randomized agent function outperform a simple reflex agent? Design such an agent and measure its performance on several environments.
- c.** Can you design an environment in which your randomized agent will perform very poorly? Show your results.
- d.** Can a reflex agent with state outperform a simple reflex agent? Design such an agent and measure its performance on several environments. Can you design a rational agent of this type?

**2.11** Repeat Exercise 2.10 for the case in which the location sensor is replaced with a "bump" sensor that detects the agent's attempts to move into an obstacle or to cross the boundaries of the environment. Suppose the bump sensor stops working; how should the agent behave?

**2.12** The vacuum environments in the preceding exercises have all been deterministic. Discuss possible agent programs for each of the following stochastic versions:

- a.** Murphy's law: twenty-five percent of the time, the *Suck* action fails to clean the floor if it is dirty and deposits dirt onto the floor if the floor is clean. How is your agent program affected if the dirt sensor gives the wrong answer 10% of the time?
- b.** Small children: At each time step, each clean square has a 10% chance of becoming dirty. Can you come up with a rational agent design for this case?