



**Figure 1.4** 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.”

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 1.4. 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 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**. The **perceptron convergence theorem** (Block *et al.*, 1962) says that the learning algorithm can adjust the connection strengths of a perceptron to match any input data, provided such a match exists. These topics are covered in Chapter 20.

### 1.3.4 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 more concrete predictions: 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 overconfidence 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 knew nothing 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 from an electronic dictionary, would suffice to preserve the exact meanings of sentences. The fact is that accurate translation requires background knowledge in order to resolve ambiguity and establish the content of the sentence. The famous retranslation 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 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.*



MACHINE EVOLUTION  
GENETIC ALGORITHM

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 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 (restricted to be simpler than the form Rosenblatt originally studied) 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).

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

WEAK METHOD

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<sub>6</sub>H<sub>13</sub>NO<sub>2</sub>) 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 (CH<sub>3</sub>) 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 even moderate-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 (C=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.

EXPERT SYSTEMS

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.

CERTAINTY FACTOR

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, 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),

FRAMES

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.

### 1.3.6 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. DuPont 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. In Britain, the Alvey report reinstated the funding that was cut by the Lighthill report.<sup>13</sup> In all three countries, however, the projects never met their ambitious goals.

Overall, the AI industry boomed from a few million dollars in 1980 to billions of dollars in 1988, including hundreds of companies building expert systems, vision systems, robots, and software and hardware specialized for these purposes. Soon after that came a period called the “AI Winter,” in which many companies fell by the wayside as they failed to deliver on extravagant promises.

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

BACK-PROPAGATION

In the mid-1980s 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*

<sup>13</sup> To save embarrassment, a new field called IKBS (Intelligent Knowledge-Based Systems) was invented because Artificial Intelligence had been officially canceled.

*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. As occurred with the separation of AI and cognitive science, modern neural network research has bifurcated into two fields, one concerned with creating effective network architectures and algorithms and understanding their mathematical properties, the other concerned with careful modeling of the empirical properties of actual neurons and ensembles of neurons.

### 1.3.8 AI adopts the scientific method (1987–present)

Recent years have seen a revolution in both the content and the methodology of work in artificial intelligence.<sup>14</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 be analyzed statistically for their importance (Cohen, 1995). It is now possible to replicate experiments by using shared repositories of test data and code.

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

<sup>14</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.

and consumer applications. Note that there is no scientific claim that humans use HMMs to recognize speech; rather, HMMs provide a mathematical framework for understanding the problem and support the engineering claim that they work well in practice.

Machine translation follows the same course as speech recognition. In the 1950s there was initial enthusiasm for an approach based on sequences of words, with models learned according to the principles of information theory. That approach fell out of favor in the 1960s, but returned in the late 1990s and now dominates the field.

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.

DATA MINING

BAYESIAN NETWORK

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™ 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. Although increased formalization and specialization led fields such as vision and robotics to become somewhat isolated from “mainstream” AI in the 1990s, this trend has reversed in recent years as tools from machine learning in particular have proved effective for many problems. The process of reintegration is already yielding significant benefits.

### 1.3.9 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. 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 aggregators.

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. Recent progress in the control of robotic cars has derived from a mixture of approaches ranging from better sensors, control-theoretic integration of sensing, localization and mapping, as well as a degree of high-level planning.

Despite these successes, some influential founders of AI, including John McCarthy (2007), Marvin Minsky (2007), Nils Nilsson (1995, 2005) and Patrick Winston (Beal and Winston, 2009), have expressed discontent with the progress of AI. They think that AI should put less emphasis on creating ever-improved versions of applications that are good at a specific task, such as driving a car, playing chess, or recognizing speech. Instead, they believe AI should return to its roots of striving for, in Simon's words, "machines that think, that learn and that create." They call the effort **human-level AI** or HLAI; their first symposium was in 2004 (Minsky *et al.*, 2004). The effort will require very large knowledge bases; Helder *et al.* (1995) discuss where these knowledge bases might come from.

HUMAN-LEVEL AI

ARTIFICIAL GENERAL  
INTELLIGENCE

FRIENDLY AI

A related idea is the subfield of **Artificial General Intelligence** or AGI (Goertzel and Pennachin, 2007), which held its first conference and organized the *Journal of Artificial General Intelligence* in 2008. AGI looks for a universal algorithm for learning and acting in any environment, and has its roots in the work of Ray Solomonoff (1964), one of the attendees of the original 1956 Dartmouth conference. Guaranteeing that what we create is really **Friendly AI** is also a concern (Yudkowsky, 2008; Omohundro, 2008), one we will return to in Chapter 26.

### 1.3.10 The availability of very large data sets (2001–present)

Throughout the 60-year history of computer science, the emphasis has been on the *algorithm* as the main subject of study. But some recent work in AI suggests that for many problems, it makes more sense to worry about the *data* and be less picky about what algorithm to apply. This is true because of the increasing availability of very large data sources: for example, trillions of words of English and billions of images from the Web (Kilgarriff and Grefenstette, 2006); or billions of base pairs of genomic sequences (Collins *et al.*, 2003).

One influential paper in this line was Yarowsky's (1995) work on word-sense disambiguation: given the use of the word "plant" in a sentence, does that refer to flora or factory? Previous approaches to the problem had relied on human-labeled examples combined with machine learning algorithms. Yarowsky showed that the task can be done, with accuracy above 96%, with no labeled examples at all. Instead, given a very large corpus of unannotated text and just the dictionary definitions of the two senses—"works, industrial plant" and "flora, plant life"—one can label examples in the corpus, and from there **bootstrap** to learn

new patterns that help label new examples. Banko and Brill (2001) show that techniques like this perform even better as the amount of available text goes from a million words to a billion and that the increase in performance from using more data exceeds any difference in algorithm choice; a mediocre algorithm with 100 million words of unlabeled training data outperforms the best known algorithm with 1 million words.

As another example, Hays and Efros (2007) discuss the problem of filling in holes in a photograph. Suppose you use Photoshop to mask out an ex-friend from a group photo, but now you need to fill in the masked area with something that matches the background. Hays and Efros defined an algorithm that searches through a collection of photos to find something that will match. They found the performance of their algorithm was poor when they used a collection of only ten thousand photos, but crossed a threshold into excellent performance when they grew the collection to two million photos.

Work like this suggests that the “knowledge bottleneck” in AI—the problem of how to express all the knowledge that a system needs—may be solved in many applications by learning methods rather than hand-coded knowledge engineering, provided the learning algorithms have enough data to go on (Halevy *et al.*, 2009). Reporters have noticed the surge of new applications and have written that “AI Winter” may be yielding to a new Spring (Havenstein, 2005). As Kurzweil (2005) writes, “today, many thousands of AI applications are deeply embedded in the infrastructure of every industry.”

## 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.

**Robotic vehicles:** A driverless robotic car named STANLEY sped through the rough terrain of the Mojave dessert at 22 mph, finishing the 132-mile course first to win the 2005 DARPA Grand Challenge. STANLEY is a Volkswagen Touareg outfitted with cameras, radar, and laser rangefinders to sense the environment and onboard software to command the steering, braking, and acceleration (Thrun, 2006). The following year CMU’s BOSS won the Urban Challenge, safely driving in traffic through the streets of a closed Air Force base, obeying traffic rules and avoiding pedestrians and other vehicles.

**Speech recognition:** A traveler calling United Airlines to book a flight can have the entire conversation guided by an automated speech recognition and dialog management system.

**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 monitored the execution of those plans—detecting, diagnosing, and recovering from problems as they occurred. Successor program MAPGEN (Al-Chang *et al.*, 2004) plans the daily operations for NASA’s Mars Exploration Rovers, and MEXAR2 (Cesta *et al.*, 2007) did mission planning—both logistics and science planning—for the European Space Agency’s Mars Express mission in 2008.