Chapter 1

Artificial Intelligence and Agents

> *The history of AI is a history of fantasies, possibilities, demonstrations, and promise. Ever since Homer wrote of mechanical "tripods" waiting on the gods at dinner, imagined mechanical assistants have been a part of our culture. However, only in the last half century have we, the AI community, been able to build experimental machines that test hypotheses about the mechanisms of thought and intelligent behavior and thereby demonstrate mechanisms that formerly existed only as theoretical possibilities.*
>
> – Bruce Buchanan [2005]

This book is about artificial intelligence, a field built on centuries of thought, which has been a recognized discipline for over 50 years. As Buchanan points out in the quote above, we now have the tools to test hypotheses about the nature of thought itself, as well as solve practical problems. Deep scientific and engineering problems have already been solved and many more are waiting to be solved. Many practical applications are currently deployed and the potential exists for an almost unlimited number of future applications. In this book, we present the principles that underlie intelligent computational agents. Those principles can help you understand current and future work in AI and equip you to contribute to the discipline yourself.

1.1 What Is Artificial Intelligence?

Artificial intelligence, or **AI**, is the field that studies *the synthesis and analysis of computational agents that act intelligently*. Let us examine each part of this definition.

An **agent** is something that acts in an environment – it does something. Agents include worms, dogs, thermostats, airplanes, robots, humans, companies, and countries.

We are interested in what an agent does; that is, how it **acts**. We judge an agent by its actions.

An agent acts **intelligently** when

- what it does is appropriate for its circumstances and its goals,
- it is flexible to changing environments and changing goals,
- it learns from experience, and
- it makes appropriate choices given its perceptual and computational limitations. An agent typically cannot observe the state of the world directly; it has only a finite memory and it does not have unlimited time to act.

A **computational** agent is an agent whose decisions about its actions can be explained in terms of computation. That is, the decision can be broken down into primitive operation that can be implemented in a physical device. This computation can take many forms. In humans this computation is carried out in "wetware"; in computers it is carried out in "hardware." Although there are some agents that are arguably not computational, such as the wind and rain eroding a landscape, it is an open question whether all intelligent agents are computational.

The central **scientific goal** of AI is to understand the principles that make intelligent behavior possible in natural or artificial systems. This is done by

- the **analysis** of natural and artificial agents;
- formulating and testing hypotheses about what it takes to construct intelligent agents; and
- designing, building, and experimenting with computational systems that perform tasks commonly viewed as requiring intelligence.

As part of science, researchers build empirical systems to test hypotheses or to explore the space of possibilities. These are quite distinct from applications that are built to be useful for an application domain.

Note that the definition is not for intelligent *thought*. We are only interested in **thinking** intelligently insofar as it leads to better performance. The role of thought is to affect action.

The central **engineering goal** of AI is the **design** and **synthesis** of useful, intelligent artifacts. We actually want to build agents that act intelligently. Such agents are useful in many applications.

1.1.1 Artificial and Natural Intelligence

Artificial intelligence (AI) is the established name for the field, but the term "artificial intelligence" is a source of much confusion because artificial intelligence may be interpreted as the opposite of real intelligence.

Interrogator: In the first line of your sonnet which reads "Shall I compare thee to a summer's day," would not "a spring day" do as well or better?

Witness: It wouldn't scan.

Interrogator: How about "a winter's day," That would scan all right.

Witness: Yes, but nobody wants to be compared to a winter's day.

Interrogator: Would you say Mr. Pickwick reminded you of Christmas?

Witness: In a way.

Interrogator: Yet Christmas is a winter's day, and I do not think Mr. Pickwick would mind the comparison.

Witness: I don't think you're serious. By a winter's day one means a typical winter's day, rather than a special one like Christmas.

Figure 1.1: A possible dialog for the Turing test (from Turing [1950])

For any phenomenon, you can distinguish real versus fake, where the fake is non-real. You can also distinguish natural versus artificial. Natural means occurring in nature and artificial means made by people.

Example 1.1 A tsunami is a large wave in an ocean caused by an earthquake or a landslide. Natural tsunamis occur from time to time. You could imagine an artificial tsunami that was made by people, for example, by exploding a bomb in the ocean, yet which is still a real tsunami. One could also imagine fake tsunamis: either artificial, using computer graphics, or natural, for example, a mirage that looks like a tsunami but is not one.

It is arguable that intelligence is different: you cannot have *fake* intelligence. If an agent behaves intelligently, it is intelligent. It is only the external behavior that defines intelligence; acting intelligently is being intelligent. Thus, artificial intelligence, if and when it is achieved, will be real intelligence created artificially.

This idea of intelligence being defined by external behavior was the motivation for a test for intelligence designed by Turing [1950], which has become known as the **Turing test**. The Turing test consists of an imitation game where an interrogator can ask a witness, via a text interface, any question. If the interrogator cannot distinguish the witness from a human, the witness must be intelligent. Figure 1.1 shows a possible dialog that Turing suggested. An agent that is not really intelligent could not fake intelligence for arbitrary topics.

There has been much debate about the Turing test. Unfortunately, although it may provide a test for how to recognize intelligence, it does not provide a way to get there; trying each year to fake it does not seem like a useful avenue of research.

The obvious naturally intelligent agent is the human being. Some people might say that worms, insects, or bacteria are intelligent, but more people would say that dogs, whales, or monkeys are intelligent (see Exercise 1 (page 42)). One class of intelligent agents that may be more intelligent than humans is the class of *organizations*. Ant colonies are a prototypical example of organizations. Each individual ant may not be very intelligent, but an ant colony can act more intelligently than any individual ant. The colony can discover food and exploit it very effectively as well as adapt to changing circumstances. Similarly, companies can develop, manufacture, and distribute products where the sum of the skills required is much more than any individual could master. Modern computers, from low-level hardware to high-level software, are more complicated than any human can understand, yet they are manufactured daily by organizations of humans. Human *society* viewed as an agent is arguably the most intelligent agent known.

It is instructive to consider where human intelligence comes from. There are three main sources:

biology: Humans have evolved into adaptable animals that can survive in various habitats.

culture: Culture provides not only language, but also useful tools, useful concepts, and the wisdom that is passed from parents and teachers to children.

life-long learning: Humans learn throughout their life and accumulate knowledge and skills.

These sources interact in complex ways. Biological evolution has provided stages of growth that allow for different learning at different stages of life. We humans and our culture have evolved together so that humans are helpless at birth, presumably because of our culture of looking after infants. Culture interacts strongly with learning. A major part of lifelong learning is what people are taught by parents and teachers. Language, which is part of culture, provides distinctions in the world that should be noticed for learning.

1.2 A Brief History of AI

Throughout human history, people have used technology to model themselves. There is evidence of this from ancient China, Egypt, and Greece that bears witness to the universality of this activity. Each new technology has, in its turn, been exploited to build intelligent agents or models of mind. Clockwork, hydraulics, telephone switching systems, holograms, analog computers, and digital computers have all been proposed both as technological metaphors for intelligence and as mechanisms for modeling mind.

About 400 years ago people started to write about the nature of thought and reason. Hobbes (1588–1679), who has been described by Haugeland [1985,

p. 85] as the "Grandfather of AI," espoused the position that thinking was symbolic reasoning like talking out loud or working out an answer with pen and paper. The idea of symbolic reasoning was further developed by Descartes (1596–1650), Pascal (1623–1662), Spinoza (1632–1677), Leibniz (1646–1716), and others who were pioneers in the philosophy of mind.

The idea of symbolic operations became more concrete with the development of computers. The first general-purpose computer designed (but not built until 1991, at the Science Museum of London) was the **Analytical Engine** by Babbage (1792–1871). In the early part of the 20th century, there was much work done on understanding computation. Several models of computation were proposed, including the Turing machine by Alan Turing (1912–1954), a theoretical machine that writes symbols on an infinitely long tape, and the lambda calculus of Church (1903–1995), which is a mathematical formalism for rewriting formulas. It can be shown that these very different formalisms are equivalent in that any function computable by one is computable by the others. This leads to the **Church–Turing thesis**:

> Any effectively computable function can be carried out on a Turing machine (and so also in the lambda calculus or any of the other equivalent formalisms).

Here **effectively computable** means following well-defined operations; "computers" in Turing's day were people who followed well-defined steps and computers as we know them today did not exist. This thesis says that all computation can be carried out on a Turing machine or one of the other equivalent computational machines. The Church–Turing thesis cannot be proved but it is a hypothesis that has stood the test of time. No one has built a machine that has carried out computation that cannot be computed by a Turing machine. There is no evidence that people can compute functions that are not Turing computable. An agent's actions are a function of its abilities, its history, and its goals or preferences. This provides an argument that computation is more than just a metaphor for intelligence; reasoning *is* computation and computation can be carried out by a computer.

Once real computers were built, some of the first applications of computers were AI programs. For example, Samuel [1959] built a checkers program in 1952 and implemented a program that learns to play checkers in the late 1950s. Newell and Simon [1956] built a program, Logic Theorist, that discovers proofs in propositional logic.

In addition to that for high-level symbolic reasoning, there was also much work on low-level learning inspired by how neurons work. McCulloch and Pitts [1943] showed how a simple thresholding "formal neuron" could be the basis for a Turing-complete machine. The first learning for these neural networks was described by Minsky [1952]. One of the early significant works was the Perceptron of Rosenblatt [1958]. The work on neural networks went into decline for a number of years after the 1968 book by Minsky and Papert [1988],

Does Afghanistan border China?
What is the capital of Upper_Volta?
Which country's capital is London?
Which is the largest african country?
How large is the smallest american country?
What is the ocean that borders African countries and that borders Asian countries?
What are the capitals of the countries bordering the Baltic?
How many countries does the Danube flow through?
What is the total area of countries south of the Equator and not in Australasia?
What is the average area of the countries in each continent?
Is there more than one country in each continent?
What are the countries from which a river flows into the Black_Sea?
What are the continents no country in which contains more than two cities whose population exceeds 1 million?
Which country bordering the Mediterranean borders a country that is bordered by a country whose population exceeds the population of India?
Which countries with a population exceeding 10 million border the Atlantic?

Figure 1.2: Some questions CHAT-80 could answer

which argued that the representations learned were inadequate for intelligent action.

These early programs concentrated on learning and search as the foundations of the field. It became apparent early that one of the main problems was how to represent the knowledge needed to solve a problem. Before learning, an agent must have an appropriate target language for the learned knowledge. There have been many proposals for representations from simple feature-based representations to complex logical representations of McCarthy and Hayes [1969] and many in between such as the frames of Minsky [1975].

During the 1960s and 1970s, success was had in building natural language understanding systems in limited domains. For example, the STUDENT program of Daniel Bobrow [1967] could solve high school algebra problems expressed in natural language. Winograd's [1972] SHRDLU system could, using restricted natural language, discuss and carry out tasks in a simulated blocks world. CHAT-80 [Warren and Pereira, 1982] could answer geographical questions placed to it in natural language. Figure 1.2 shows some questions that CHAT-80 answered based on a database of facts about countries, rivers, and so on. All of these systems could only reason in very limited domains using restricted vocabulary and sentence structure.

During the 1970s and 1980s, there was a large body of work on **expert systems**, where the aim was to capture the knowledge of an expert in some domain so that a computer could carry out expert tasks. For example, **DENDRAL** [Buchanan and Feigenbaum, 1978], developed from 1965 to 1983 in the field of organic chemistry, proposed plausible structures for new organic compounds. **MYCIN** [Buchanan and Shortliffe, 1984], developed from 1972 to 1980, diagnosed infectious diseases of the blood, prescribed antimicrobial therapy, and explained its reasoning. The 1970s and 1980s were also a period when AI reasoning became widespread in languages such as **Prolog** [Colmerauer and Roussel, 1996; Kowalski, 1988].

During the 1990s and the 2000s there was great growth in the subdisciplines of AI such as perception, probabilistic and decision-theoretic reasoning, planning, embodied systems, machine learning, and many other fields. There has also been much progress on the foundations of the field; these form the foundations of this book.

1.2.1 Relationship to Other Disciplines

AI is a very young discipline. Other disciplines as diverse as philosophy, neurobiology, evolutionary biology, psychology, economics, political science, sociology, anthropology, control engineering, and many more have been studying intelligence much longer.

The science of AI could be described as "synthetic psychology," "experimental philosophy," or "computational epistemology"– **epistemology** is the study of knowledge. AI can be seen as a way to study the old problem of the nature of knowledge and intelligence, but with a more powerful experimental tool than was previously available. Instead of being able to observe only the external behavior of intelligent systems, as philosophy, psychology, economics, and sociology have traditionally been able to do, AI researchers experiment with executable models of intelligent behavior. Most important, such models are open to inspection, redesign, and experiment in a complete and rigorous way. Modern computers provide a way to construct the models about which philosophers have only been able to theorize. AI researchers can experiment with these models as opposed to just discussing their abstract properties. AI theories can be empirically grounded in implementation. Moreover, we are often surprised when simple agents exhibit complex behavior. We would not have known this without implementing the agents.

It is instructive to consider an analogy between the development of flying machines over the past few centuries and the development of thinking machines over the past few decades. There are several ways to understand flying. One is to dissect known flying animals and hypothesize their common structural features as necessary fundamental characteristics of any flying agent. With this method, an examination of birds, bats, and insects would suggest that flying involves the flapping of wings made of some structure covered with feathers or a membrane. Furthermore, the hypothesis could be tested by

strapping feathers to one's arms, flapping, and jumping into the air, as Icarus did. An alternate methodology is to try to understand the principles of flying without restricting oneself to the natural occurrences of flying. This typically involves the construction of artifacts that embody the hypothesized principles, even if they do not behave like flying animals in any way except flying. This second method has provided both useful tools – airplanes – and a better understanding of the principles underlying flying, namely *aerodynamics*.

AI takes an approach analogous to that of aerodynamics. AI researchers are interested in testing general hypotheses about the nature of intelligence by building machines that are intelligent and that do not necessarily mimic humans or organizations. This also offers an approach to the question, "Can computers really think?" by considering the analogous question, "Can airplanes really fly?"

AI is intimately linked with the discipline of computer science. Although there are many non-computer scientists who are doing AI research, much, if not most, AI research is done within computer science departments. This is appropriate because the study of computation is central to AI. It is essential to understand algorithms, data structures, and combinatorial complexity to build intelligent machines. It is also surprising how much of computer science started as a spinoff from AI, from timesharing to computer algebra systems.

Finally, AI can be seen as coming under the umbrella of *cognitive science*. Cognitive science links various disciplines that study cognition and reasoning, from psychology to linguistics to anthropology to neuroscience. AI distinguishes itself within cognitive science by providing tools to build intelligence rather than just studying the external behavior of intelligent agents or dissecting the inner workings of intelligent systems.

1.3 Agents Situated in Environments

AI is about practical reasoning: reasoning in order to do something. A coupling of perception, reasoning, and acting comprises an **agent**. An agent acts in an **environment**. An agent's environment may well include other agents. An agent together with its environment is called a **world**.

An agent could be, for example, a coupling of a computational engine with physical sensors and actuators, called a **robot**, where the environment is a physical setting. It could be the coupling of an advice-giving computer – an **expert system** – with a human who provides perceptual information and carries out the task. An agent could be a program that acts in a purely computational environment – a **software agent**.

Figure 1.3 shows the inputs and outputs of an agent. At any time, what an agent does depends on its

- **prior knowledge** about the agent and the environment;
- **history** of interaction with the environment, which is composed of

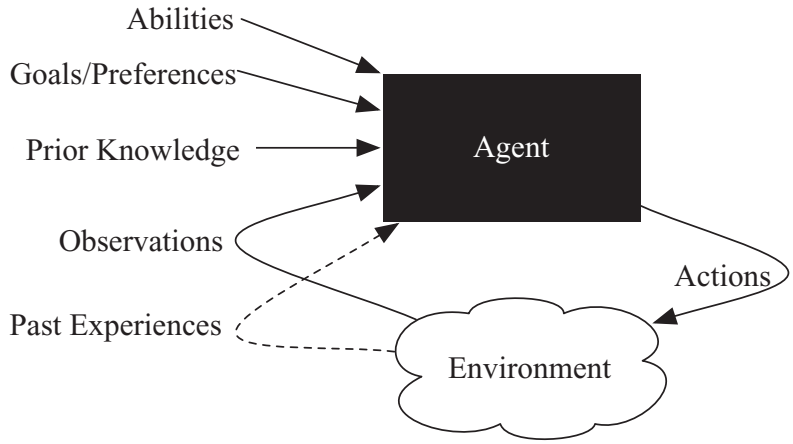

Figure 1.3: An agent interacting with an environment

  - **observations** of the current environment and
  - **past experiences** of previous actions and observations, or other data, from which it can learn;
- **goals** that it must try to achieve or preferences over states of the world; and
- **abilities**, which are the primitive actions it is capable of carrying out.

Two deterministic agents with the same prior knowledge, history, abilities, and goals should do the same thing. Changing any one of these can result in different actions.

Each agent has some internal state that can encode beliefs about its environment and itself. It may have goals to achieve, ways to act in the environment to achieve those goals, and various means to modify its beliefs by reasoning, perception, and learning. This is an all-encompassing view of intelligent agents varying in complexity from a simple thermostat, to a team of mobile robots, to a diagnostic advising system whose perceptions and actions are mediated by human beings, to society itself.

1.4 Knowledge Representation

Typically, a problem to solve or a task to carry out, as well as what constitutes a solution, is only given informally, such as "deliver parcels promptly when they arrive" or "fix whatever is wrong with the electrical system of the house."

The general framework for solving problems by computer is given in Figure 1.4 (on the next page). To solve a problem, the designer of a system must

- flesh out the task and determine what constitutes a solution;
- represent the problem in a language with which a computer can reason;

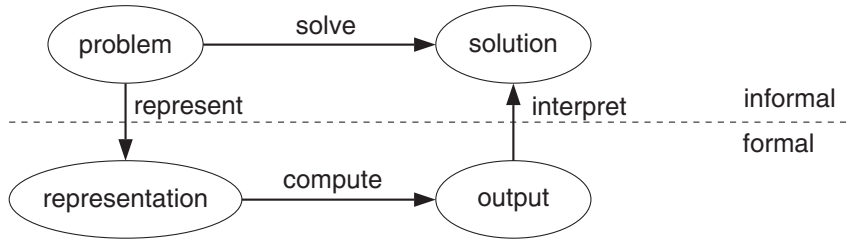

Figure 1.4: The role of representations in solving problems

- use the computer to compute an output, which is an answer presented to a user or a sequence of actions to be carried out in the environment; and
- interpret the output as a solution to the problem.

Knowledge is the information about a domain that can be used to solve problems in that domain. To solve many problems requires much knowledge, and this knowledge must be represented in the computer. As part of designing a program to solve problems, we must define how the knowledge will be represented. A **representation scheme** is the form of the knowledge that is used in an agent. A **representation** of some piece of knowledge is the internal representation of the knowledge. A representation scheme specifies the form of the knowledge. A **knowledge base** is the representation of all of the knowledge that is stored by an agent.

A good representation scheme is a compromise among many competing objectives. A representation should be

- rich enough to express the knowledge needed to solve the problem.
- as close to the problem as possible; it should be compact, natural, and maintainable. It should be easy to see the relationship between the representation and the domain being represented, so that it is easy to determine whether the knowledge represented is correct. A small change in the problem should result in a small change in the representation of the problem.
- amenable to efficient computation, which usually means that it is able to express features of the problem that can be exploited for computational gain and able to trade off accuracy and computation time.
- able to be acquired from people, data and past experiences.

Many different representation schemes have been designed. Many of these start with some of these objectives and are then expanded to include the other objectives. For example, some are designed for learning and then expanded to allow richer problem solving and inference abilities. Some representation schemes are designed with expressiveness in mind, and then inference and learning are added on. Some schemes start from tractable inference and then are made more natural, and more able to be acquired.

Some of the questions that must be considered when given a problem or a task are the following:

- What is a solution to the problem? How good must a solution be?
- How can the problem be represented? What distinctions in the world are needed to solve the problem? What specific knowledge about the world is required? How can an agent acquire the knowledge from experts or from experience? How can the knowledge be debugged, maintained, and improved?
- How can the agent compute an output that can be interpreted as a solution to the problem? Is worst-case performance or average-case performance the critical time to minimize? Is it important for a human to understand how the answer was derived?

These issues are discussed in the next sections and arise in many of the representation schemes presented later in the book.

1.4.1 Defining a Solution

Given an informal description of a problem, before even considering a computer, a knowledge base designer should determine what would constitute a solution. This question arises not only in AI but in any software design. Much of **software engineering** involves refining the specification of the problem.

Typically, problems are not well specified. Not only is there usually much left unspecified, but also the unspecified parts cannot be filled in arbitrarily. For example, if you ask a trading agent to find out all the information about resorts that may have health issues, you do not want the agent to return the information about all resorts, even though all of the information you requested is in the result. However, if the trading agent does not have complete knowledge about the resorts, returning all of the information may be the only way for it to guarantee that all of the requested information is there. Similarly, you do not want a delivery robot, when asked to take all of the trash to the garbage can, to take everything to the garbage can, even though this may be the only way to guarantee that all of the trash has been taken. Much work in AI is motivated by **commonsense reasoning**; we want the computer to be able to make commonsense conclusions about the unstated assumptions.

Given a well-defined problem, the next issue is whether it matters if the answer returned is incorrect or incomplete. For example, if the specification asks for all instances, does it matter if some are missing? Does it matter if there are some extra instances? Often a person does not want just any solution but the best solution according to some criteria. There are four common classes of solutions:

Optimal solution An **optimal solution** to a problem is one that is the best solution according to some measure of solution quality. This measure is typically specified as an **ordinal**, where only the ordermatters. However, in some

situations, such as when combining multiple criteria or when reasoning under uncertainty, you need a **cardinal** measure, where the relative magnitudes also matter. An example of an ordinal measure is for the robot to take out as much trash as possible; the more trash it can take out, the better. As an example of a cardinal measure, you may want the delivery robot to take as much of the trash as possible to the garbage can, minimizing the distance traveled, and explicitly specify a trade-off between the effort required and the proportion of the trash taken out. It may be better to miss some trash than to waste too much time. One general cardinal measure of desirability, known as **utility**, is used in decision theory (page 373).

Satisficing solution Often an agent does not need the best solution to a problem but just needs some solution. A **satisficing solution** is one that is good enough according to some description of which solutions are adequate. For example, a person may tell a robot that it must take all of trash out, or tell it to take out three items of trash.

Approximately optimal solution One of the advantages of a cardinal measure of success is that it allows for approximations. An **approximately optimal solution** is one whose measure of quality is close to the best that could theoretically be obtained. Typically agents do not need optimal solutions to problems; they only must get close enough. For example, the robot may not need to travel the optimal distance to take out the trash but may only need to be within, say, 10% of the optimal distance.

For some problems, it is much easier computationally to get an approximately optimal solution than to get an optimal solution. However, for other problems, it is (asymptotically) just as difficult to guarantee finding an approximately optimal solution as it is to guarantee finding an optimal solution. Some approximation algorithms guarantee that a solution is within some range of optimal, but for some algorithms no guarantees are available.

Probable solution A **probable solution** is one that, even though it may not actually be a solution to the problem, is likely to be a solution. This is one way to approximate, in a precise manner, a satisficing solution. For example, in the case where the delivery robot could drop the trash or fail to pick it up when it attempts to, you may need the robot to be 80% sure that it has picked up three items of trash. Often you want to distinguish the **false-positive error** rate (the proportion of the answers given by the computer that are not correct) from the **false-negative error** rate (which is the proportion of those answers not given by the computer that are indeed correct). Some applications are much more tolerant of one of these errors than of the other.

These categories are not exclusive. A form of learning known as probably approximately correct (PAC) learning considers probably learning an approximately correct concept (page 332).

1.4.2 Representations

Once you have some requirements on the nature of a solution, you must represent the problem so a computer can solve it.

Computers and human minds are examples of **physical symbol systems**. A **symbol** is a meaningful pattern that can be manipulated. Examples of symbols are written words, sentences, gestures, marks on paper, or sequences of bits. A **symbol system** creates, copies, modifies, and destroys symbols. Essentially, a symbol is one of the patterns manipulated as a unit by a symbol system.

The term physical is used, because symbols in a physical symbol system are physical objects that are part of the real world, even though they may be internal to computers and brains. They may also need to physically affect action or motor control.

Much of AI rests on the **physical symbol system hypothesis** of Newell and Simon [1976]:

> A physical symbol system has the necessary and sufficient means for general intelligent action.

This is a strong hypothesis. It means that any intelligent agent is necessarily a physical symbol system. It also means that a physical symbol system is all that is needed for intelligent action; there is no magic or an as-yet-to-be-discovered quantum phenomenon required. It does not imply that a physical symbol system does not need a body to sense and act in the world. The physical symbol system hypothesis is an empirical hypothesis that, like other scientific hypotheses, is to be judged by how well it fits the evidence, and what alternative hypotheses exist. Indeed, it could be false.

An intelligent agent can be seen as manipulating symbols to produce action. Many of these symbols are used to refer to things in the world. Other symbols may be useful concepts that may or may not have external meaning. Yet other symbols may refer to internal states of the agent.

An agent can use physical symbol systems to model the world. A **model** of a world is a representation of the specifics of what is true in the world or of the dynamic of the world. The world does not have to be modeled at the most detailed level to be useful. All models are **abstractions**; they represent only part of the world and leave out many of the details. An agent can have a very simplistic model of the world, or it can have a very detailed model of the world. The **level of abstraction** provides a partial ordering of abstraction. A lower-level abstraction includes more details than a higher-level abstraction. An agent can have multiple, even contradictory, models of the world. The models are judged not by whether they are correct, but by whether they are useful.

Example 1.2 A delivery robot can model the environment at a high level of abstraction in terms of rooms, corridors, doors, and obstacles, ignoring distances, its size, the steering angles needed, the slippage of the wheels, the weight of parcels, the details of obstacles, the political situation in Canada, and

virtually everything else. The robot could model the environment at lower levels of abstraction by taking some of these details into account. Some of these details may be irrelevant for the successful implementation of the robot, but some may be crucial for the robot to succeed. For example, in some situations the size of the robot and the steering angles may be crucial for not getting stuck around a particular corner. In other situations, if the robot stays close to the center of the corridor, it may not need to model its width or the steering angles.

Choosing an appropriate level of abstraction is difficult because

- a high-level description is easier for a human to specify and understand.
- a low-level description can be more accurate and more predictive. Often high-level descriptions abstract away details that may be important for actually solving the problem.
- the lower the level, the more difficult it is to reason with. This is because a solution at a lower level of detail involves more steps and many more possible courses of action exist from which to choose.
- you may not know the information needed for a low-level description. For example, the delivery robot may not know what obstacles it will encounter or how slippery the floor will be at the time that it must decide what to do.

It is often a good idea to model an environment at multiple levels of abstraction. This issue is further discussed in Section 2.3 (page 50).

Biological systems, and computers, can be described at multiple levels of abstraction. At successively lower levels are the neural level, the biochemical level (what chemicals and what electrical potentials are being transmitted), the chemical level (what chemical reactions are being carried out), and the level of physics (in terms of forces on atoms and quantum phenomena). What levels above the neuron level are needed to account for intelligence is still an open question. Note that these levels of description are echoed in the hierarchical structure of science itself, where scientists are divided into physicists, chemists, biologists, psychologists, anthropologists, and so on. Although no level of description is more important than any other, we conjecture that you do not have to emulate every level of a human to build an AI agent but rather you can emulate the higher levels and build them on the foundation of modern computers. This conjecture is part of what AI studies.

The following are two levels that seem to be common to both biological and computational entities:

- The **knowledge level** is a level of abstraction that considers what an agent knows and believes and what its goals are. The knowledge level considers what an agent knows, but not how it reasons. For example, the delivery agent's behavior can be described in terms of whether it knows that a parcel has arrived or not and whether it knows where a particular person is or not. Both human and robotic agents can be described at the knowledge level. At this level, you do not specify how the solution will be computed or even which of the many possible strategies available to the agent will be used.

- The **symbol level** is a level of description of an agent in terms of the reasoning it does. To implement the knowledge level, an agent manipulates symbols to produce answers. Many cognitive science experiments are designed to determine what symbol manipulation occurs during reasoning. Note that whereas the knowledge level is about what the agent believes about the external world and what its goals are in terms of the outside world, the symbol level is about what goes on inside an agent to reason about the external world.

1.4.3 Reasoning and Acting

The manipulation of symbols to produce action is called **reasoning**.

One way that AI representations differ from computer programs in traditional languages is that an AI representation typically specifies **what** needs to be computed, not **how** it is to be computed. We might specify that the agent should find the most likely disease a patient has, or specify that a robot should get coffee, but not give detailed instructions on how to do these things. Much AI reasoning involves searching through the space of possibilities to determine how to complete a task.

In deciding what an agent will do, there are three aspects of computation that must be distinguished: (1) the computation that goes into the design of the agent, (2) the computation that the agent can do before it observes the world and needs to act, and (3) the computation that is done by the agent as it is acting.

- **Design time reasoning** is the reasoning that is carried out to design the agent. It is carried out by the designer of the agent, not the agent itself.
- **Offline computation** is the computation done by the agent before it has to act. It can include compilation and learning. Offline, the agent takes background knowledge and data and compiles them into a usable form called a **knowledge base**. **Background knowledge** can be given either at design time or offline.
- **Online computation** is the computation done by the agent between observing the environment and acting in the environment. A piece of information obtained online is called an **observation**. An agent typically must use both its knowledge base and its observations to determine what to do.

It is important to distinguish between the knowledge in the mind of the designer and the knowledge in the mind of the agent. Consider the extreme cases:

- At one extreme is a highly specialized agent that works well in the environment for which it was designed, but it is helpless outside of this niche. The designer may have done considerable work in building the agent, but the agent may not need to do very much to operate well. An example is a thermostat. It may be difficult to design a thermostat so that it turns on and off at exactly the right temperatures, but the thermostat itself does not have to do much computation. Another example is a painting robot that always

paints the same parts in an automobile factory. There may be much design time or offline computation to get it to work perfectly, but the painting robot can paint parts with little online computation; it senses that there is a part in position, but then it carries out its predefined actions. These very specialized agents do not adapt well to different environments or to changing goals. The painting robot would not notice if a different sort of part were present and, even if it did, it would not know what to do with it. It would have to be redesigned or reprogrammed to paint different parts or to change into a sanding machine or a dog washing machine.

- At the other extreme is a very flexible agent that can survive in arbitrary environments and accept new tasks at run time. Simple biological agents such as insects can adapt to complex changing environments, but they cannot carry out arbitrary tasks. Designing an agent that can adapt to complex environments and changing goals is a major challenge. The agent will know much more about the particulars of a situation than the designer. Even biology has not produced many such agents. Humans may be the only extant example, but even humans need time to adapt to new environments.

Even if the flexible agent is our ultimate dream, researchers have to reach this goal via more mundane goals. Rather than building a universal agent, which can adapt to any environment and solve any task, they have built particular agents for particular environmental niches. The designer can exploit the structure of the particular niche and the agent does not have to reason about other possibilities.

Two broad strategies have been pursued in building agents:

- The first is to simplify environments and build complex reasoning systems for these simple environments. For example, factory robots can do sophisticated tasks in the engineered environment of a factory, but they may be hopeless in a natural environment. Much of the complexity of the problem can be reduced by simplifying the environment. This is also important for building practical systems because many environments can be engineered to make them simpler for agents.
- The second strategy is to build simple agents in natural environments. This is inspired by seeing how **insects** can survive in complex environments even though they have very limited reasoning abilities. Researchers then make the agents have more reasoning abilities as their tasks become more complicated.

One of the advantages of simplifying environments is that it may enable us to prove properties of agents or to optimize agents for particular situations. Proving properties or optimization typically requires a model of the agent and its environment. The agent may do a little or a lot of reasoning, but an observer or designer of the agent may be able to reason about the agent and the environment. For example, the designer may be able to prove whether the agent can achieve a goal, whether it can avoid getting into situations that may be bad for the agent (**safety goals**), whether it will get stuck somewhere (**liveness**),

or whether it will eventually get around to each of the things it should do (**fairness**). Of course, the proof is only as good as the model.

The advantage of building agents for complex environments is that these are the types of environments in which humans live and want our agents to live.

Fortunately, research along both lines is being carried out. In the first case, researchers start with simple environments and make the environments more complex. In the second case, researchers increase the complexity of the behaviors that the agents can carry out.

1.5 Dimensions of Complexity

Agents acting in environments range in complexity from thermostats to companies with multiple goals acting in competitive environments. A number of dimensions of complexity exist in the design of intelligent agents. These dimensions may be be considered separately but must be combined to build an intelligent agent. These dimensions define a **design space** of AI; different points in this space can be obtained by varying the values of the dimensions.

Here we present nine dimensions: modularity, representation scheme, planning horizon, sensing uncertainty, effect uncertainty, preference, number of agents, learning, and computational limits. These dimensions give a coarse division of the design space of intelligent agents. There are many other design choices that must be made to build an intelligent agent.

1.5.1 Modularity

The first dimension is the level of modularity.

Modularity is the extent to which a system can be decomposed into interacting modules that can be understood separately.

Modularity is important for reducing complexity. It is apparent in the structure of the brain, serves as a foundation of computer science, and is an important part of any large organization.

Modularity is typically expressed in terms of a hierarchical decomposition. For example, a human's visual cortex and eye constitute a module that takes in light and perhaps higher-level goals and outputs some simplified description of a scene. Modularity is hierarchical if the modules are organized into smaller modules, which, in turn, can be organized into even smaller modules, all the way down to primitive operations. This hierarchical organization is part of what biologists investigate. Large organizations have a hierarchical organization so that the top-level decision makers are not overwhelmed by details and do not have to micromanage all details of the organization. Procedural abstraction and object-oriented programming in computer science are designed to enable simplification of a system by exploiting modularity and abstraction.

In the **modularity dimension**, an agent's structure is one of the following:

- **flat**: there is no organizational structure;
- **modular**: the system is decomposed into interacting modules that can be understood on their own; or
- **hierarchical**: the system is modular, and the modules themselves are decomposed into interacting modules, each of which are hierarchical systems, and this recursion grounds out into simple components.

In a flat or modular structure the agent typically reasons at a single level of abstraction. In a hierarchical structure the agent reasons at multiple levels of abstraction. The lower levels of the hierarchy involve reasoning at a lower level of abstraction.

Example 1.3 In taking a trip from home to a holiday location overseas, an agent, such as yourself, must get from home to an airport, fly to an airport near the destination, then get from the airport to the destination. It also must make a sequence of specific leg or wheel movements to actually move. In a flat representation, the agent chooses one level of abstraction and reasons at that level. A modular representation would divide the task into a number of subtasks that can be solved separately (e.g., booking tickets, getting to the departure airport, getting to the destination airport, and getting to the holiday location). In a hierarchical representation, the agent will solve these subtasks in a hierarchical way, until the problem is reduced to simple problems such a sending an http request or taking a particular step.

A hierarchical decomposition is important for reducing the complexity of building an intelligent agent that acts in a complex environment. However, to explore the other dimensions, we initially ignore the hierarchical structure and assume a flat representation. Ignoring hierarchical decomposition is often fine for small or moderately sized problems, as it is for simple animals, small organizations, or small to moderately sized computer programs. When problems or systems become complex, some hierarchical organization is required.

How to build hierarchically organized agents is discussed in Section 2.3 (page 50).

1.5.2 Representation Scheme

The **representation scheme dimension** concerns how the world is described.

The different ways the world could be to affect what an agent should do are called **states**. We can factor the state of the world into the agent's internal state (its belief state) and the environment state.

At the simplest level, an agent can reason explicitly in terms of individually identified states.

Example 1.4 A thermostat for a heater may have two belief states: *off* and *heating*. The environment may have three states: *cold*, *comfortable*, and *hot*. There are thus six states corresponding to the different combinations of belief and environment states. These states may not fully describe the world, but they are adequate to describe what a thermostat should do. The thermostat should move to, or stay in, *heating* if the environment is *cold* and move to, or stay in, *off* if the environment is *hot*. If the environment is *comfortable*, the thermostat should stay in its current state. The agent heats in the *heating* state and does not heat in the *off* state.

Instead of enumerating states, it is often easier to reason in terms of the state's features or propositions that are true or false of the state. A state may be described in terms of **features**, where a feature has a value in each state [see Section 4.1 (page 112)].

Example 1.5 An agent that has to look after a house may have to reason about whether light bulbs are broken. It may have features for the position of each switch, the status of each switch (whether it is working okay, whether it is shorted, or whether it is broken), and whether each light works. The feature *pos_s2* may be a feature that has value *up* when switch *s2* is up and has value *down* when the switch is down. The state of the house's lighting may be described in terms of values for each of these features.

A **proposition** is a Boolean feature, which means that its value is either *true* or *false*. Thirty propositions can encode $2^{30} = 1,073,741,824$ states. It may be easier to specify and reason with the thirty propositions than with more than a billion states. Moreover, having a compact representation of the states indicates understanding, because it means that an agent has captured some regularities in the domain.

Example 1.6 Consider an agent that has to recognize letters of the alphabet. Suppose the agent observes a binary image, a 30×30 grid of pixels, where each of the 900 grid points is either on or off (i.e., it is not using any color or gray scale information). The action is to determine which of the letters $\{a, \ldots, z\}$ is drawn in the image. There are 2^{900} different states of the image, and so $26^{2^{900}}$ different functions from the image state into the characters $\{a, \ldots, z\}$. We cannot even represent such functions in terms of the state space. Instead, we define features of the image, such as line segments, and define the function from images to characters in terms of these features.

When describing a complex world, the features can depend on relations and individuals. A relation on a single individual is a property. There is a feature for each possible relationship among the individuals.

Example 1.7 The agent that looks after a house in Example 1.5 could have the lights and switches as individuals, and relations *position* and *connected_to*. Instead of the feature $position\_s_1 = up$, it could use the relation $position(s_1, up)$.

This relation enables the agent to reason about all switches or for an agent to have knowledge about switches that can be used when the agent encounters a switch.

Example 1.8 If an agent is enrolling students in courses, there could be a feature that gives the grade of a student in a course, for every student–course pair where the student took the course. There would be a *passed* feature for every student–course pair, which depends on the *grade* feature for that pair. It may be easier to reason in terms of individual students, courses and grades, and the relations *grade* and *passed*. By defining how *passed* depends on *grade* once, the agent can apply the definition for each student and course. Moreover, this can be done before the agent knows of any of the individuals and so before it knows any of the features.

Thus, instead of dealing with features or propositions, it is often more convenient to have **relational descriptions** in terms of **individuals** and **relations** among them. For example, one binary relation and 100 individuals can represent $100^2 = 10,000$ propositions and 2^{10000} states. By reasoning in terms of relations and individuals, an agent can specify reason about whole classes of individuals without ever enumerating the features or propositions, let alone the states. An agent may have to reason about infinite sets of individuals, such as the set of all numbers or the set of all sentences. To reason about an unbounded or infinite number of individuals, an agent cannot reason in terms of states or features; it must reason at the relational level.

In the **representation scheme dimension**, the agent reasons in terms of

- states,
- features, or
- relational descriptions, in terms of individuals and relations.

Some of the frameworks will be developed in terms of states, some in terms of features and some relationally.

Reasoning in terms of states is introduced in Chapter 3. Reasoning in terms of features is introduced in Chapter 4. We consider relational reasoning starting in Chapter 12.

1.5.3 Planning Horizon

The next dimension is how far ahead in time the agent plans. For example, when a dog is called to come, it should turn around to start running in order to get a reward in the future. It does not act only to get an immediate reward. Plausibly, a dog does not act for goals arbitrarily far in the future (e.g., in a few months), whereas people do (e.g., working hard now to get a holiday next year).

How far the agent "looks into the future" when deciding what to do is called the **planning horizon**. That is, the planning horizon is how far ahead the

agent considers the consequences of its actions. For completeness, we include the non-planning case where the agent is not reasoning in time. The time points considered by an agent when planning are called **stages**.

In the **planning horizon dimension**, an agent is one of the following:

- A **non-planning** agent is an agent that does not consider the future when it decides what to do or when time is not involved.
- A **finite horizon** planner is an agent that looks for a fixed finite number of time steps ahead. For example, a doctor may have to treat a patient but may have time for some testing and so there may be two stages: a testing stage and a treatment stage to plan for. In the degenerate case where an agent only looks one time step ahead, it is said to be **greedy** or **myopic**.
- An **indefinite horizon** planner is an agent that looks ahead some finite, but not predetermined, number of steps ahead. For example, an agent that must get to some location may not know *a priori* how many steps it will take to get there.
- An **infinite horizon** planner is an agent that plans on going on forever. This is often called a **process**. For example, the stabilization module of a legged robot should go on forever; it cannot stop when it has achieved stability, because the robot has to keep from falling over.

1.5.4 Uncertainty

An agent could assume there is no uncertainty, or it could take uncertainty in the domain into consideration. Uncertainty is divided into two dimensions: one for uncertainty from sensing and one for uncertainty about the effect of actions.

Sensing Uncertainty

In some cases, an agent can observe the state of the world directly. For example, in some board games or on a factory floor, an agent may know exactly the state of the world. In many other cases, it may only have some noisy perception of the state and the best it can do is to have a probability distribution over the set of possible states based on what it perceives. For example, given a patient's symptoms, a medical doctor may not actually know which disease a patient may have and may have only a probability distribution over the diseases the patient may have.

The **sensing uncertainty dimension** concerns whether the agent can determine the state from the observations:

- **Fully observable** is when the agent knows the state of the world from the observations.
- **Partially observable** is when the agent does not directly observe the state of the world. This occurs when many possible states can result in the same observations or when observations are noisy.

Assuming the world is fully observable is often done as a simplifying assumption to keep reasoning tractable.

Effect Uncertainty

In some cases an agent knows the effect of an action. That is, given a state and an action, it can accurately predict the state resulting from carrying out that action in that state. For example, an agent interacting with a file system may be able to predict the effect of deleting a file given the state of the file system. In many cases, it is difficult to predict the effect of an action, and the best an agent can do is to have a probability distribution over the effects. For example, a person may not know the effect of calling his dog, even if he knew the state of the dog, but, based on experience, he has some idea of what it will do. The dog owner may even have some idea of what another dog, that he has never seen before, will do if he calls it.

The **effect uncertainty dimension** is that the dynamics can be

- **deterministic** – when the state resulting from an action is determined by an action and the prior state or
- **stochastic** – when there is only a probability distribution over the resulting states.

This dimension only makes sense when the world is fully observable. If the world is partially observable, a stochastic system can be modeled as a deterministic system where the effect of an action depends on some unobserved feature. It is a separate dimension because many of the frameworks developed are for the fully observable, stochastic action case.

Planning with deterministic actions is considered in Chapter 8. Planning with stochastic actions and with partially observable domains is considered in Chapter 9.

1.5.5 Preference

Agents act to have better outcomes for themselves. The only reason to choose one action over another is because the preferred action will lead to more desirable outcomes.

An agent may have a simple goal, which is a state to be reached or a proposition to be true such as getting its owner a cup of coffee (i.e., end up in a state where she has coffee). Other agents may have more complex preferences. For example, a medical doctor may be expected to take into account suffering, life expectancy, quality of life, monetary costs (for the patient, the doctor, and society), the ability to justify decisions in case of a lawsuit, and many other desiderata. The doctor must trade these considerations off when they conflict, as they invariably do.

The **preference dimension** is whether the agent has

- **goals**, either **achievement goals** to be achieved in some final state or **maintenance goals** that must be maintained in all visited states. For example, the goals for a robot may be to get two cups of coffee and a banana, and not to make a mess or hurt anyone.
- **complex preferences** involve trade-offs among the desirability of various outcomes, perhaps at different times. An **ordinal preference** is where only the ordering of the preferences is important. A **cardinal preference** is where the magnitude of the values matters. For example, an ordinal preference may be that Sam prefers cappuccino over black coffee and prefers black coffee over tea. A cardinal preference may give a trade-off between the wait time and the type of beverage, and a mess–taste trade-off, where Sam is prepared to put up with more mess in the preparation of the coffee if the taste of the coffee is exceptionally good.

Goals are considered in Chapter 8. Complex preferences are considered in Chapter 9.

1.5.6 Number of Agents

An agent reasoning about what it should do in an environment where it is the only agent is difficult enough. However, reasoning about what to do when there are other agents who are also reasoning is much more difficult. An agent in a multiagent setting should reason strategically about other agents; the other agents may act to trick or manipulate the agent or may be available to cooperate with the agent. With multiple agents, is often optimal to act randomly because other agents can exploit deterministic strategies. Even when the agents are cooperating and have a common goal, the problem of coordination and communication makes multiagent reasoning more challenging. However, many domains contain multiple agents and ignoring other agents' strategic reasoning is not always the best way for an agent to reason.

Taking the point of view of a single agent, the **number of agents dimension** considers whether the agent does

- **single agent** reasoning, where the agent assumes that any other agents are just part of the environment. This is a reasonable assumption if there are no other agents or if the other agents are not going to change what they do based on the agent's action.
- **multiple agent** reasoning, where the agent takes the reasoning of other agents into account. This happens when there are other intelligent agents whose goals or preferences depend, in part, on what the agent does or if the agent must communicate with other agents.

Reasoning in the presence of other agents is much more difficult if the agents can act simultaneously or if the environment is only partially observable. Multiagent systems are considered in Chapter 10.

1.5.7 Learning

In some cases, a designer of an agent may have a good model of the agent and its environment. Often a designer does not have a good model, and an agent should use data from its past experiences and other sources to help it decide what to do.

The **learning dimension** determines whether

- **knowledge is given** or
- **knowledge is learned** (from data or past experience).

Learning typically means finding the best model that fits the data. Sometimes this is as simple as tuning a fixed set of parameters, but it can also mean choosing the best representation out of a class of representations. Learning is a huge field in itself but does not stand in isolation from the rest of AI. There are many issues beyond fitting data, including how to incorporate background knowledge, what data to collect, how to represent the data and the resulting representations, what learning biases are appropriate, and how the learned knowledge can be used to affect how the agent acts.

Learning is considered in Chapters 7, 11, and 14.

1.5.8 Computational Limits

Sometimes an agent can decide on its best action quickly enough for it to act. Often there are computational resource limits that prevent an agent from carrying out the best action. That is, the agent may not be able to find the best action quickly enough within its memory limitations to act while that action is still the best thing to do. For example, it may not be much use to take 10 minutes to derive what was the best thing to do 10 minutes ago, when the agent has to act *now*. Often, instead, an agent must trade off how long it takes to get a solution with how good the solution is; it may be better to find a reasonable solution quickly than to find a better solution later because the world will have changed during the computation.

The **computational limits dimension** determines whether an agent has

- **perfect rationality**, where an agent reasons about the best action without taking into account its limited computational resources; or
- **bounded rationality**, where an agent decides on the best action that it can find given its computational limitations.

Computational resource limits include computation time, memory, and numerical accuracy caused by computers not representing real numbers exactly.

An **anytime algorithm** is an algorithm whose solution quality improves with time. In particular, it is one that can produce its current best solution at any time, but given more time it could produce even better solutions. We can ensure that the quality doesn't decrease by allowing the agent to store the best solution found so far and return that when asked for a solution. However, waiting to act

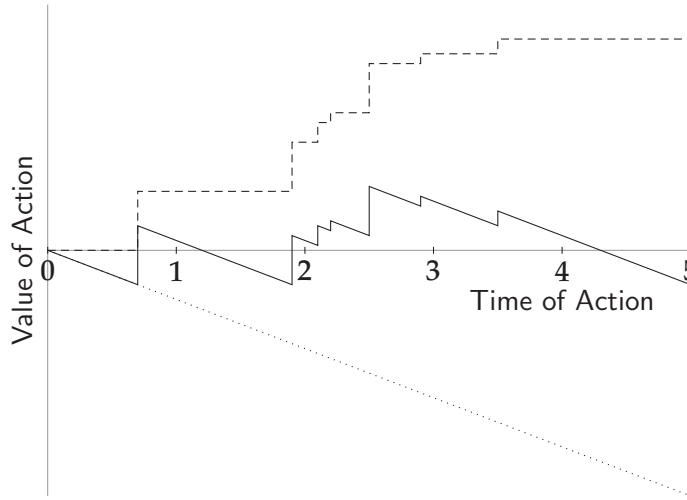

Figure 1.5: Solution quality as a function of time for an anytime algorithm. The agent has to choose an action. As time progresses, the agent can determine better actions. The value to the agent of the best action found so far, if it had been carried out initially, is given by the dashed line. The reduction in value to the agent by waiting to act is given by the dotted line. The net value to the agent, as a function of the time it acts, is given by the solid line.

has a cost; it may be better for an agent to act before it has found what would have been the best solution.

Example 1.9 Figure 1.5 shows how the computation time of an anytime algorithm can affect the solution quality. The agent has to carry out an action but can do some computation to decide what to do. The absolute solution quality, had the action been carried out at time zero, shown as the dashed line at the top, is improving as the agent takes time to reason. However, there is a penalty associated with taking time to act. In this figure, the penalty, shown as the dotted line at the bottom, is proportional to the time taken before the agent acts. These two values can be added to get the discounted quality, the time-dependent value of computation; this is the solid line in the middle of the graph. For the example of Figure 1.5 , an agent should compute for about 2.5 time units, and then act, at which point the discounted quality achieves its maximum value. If the computation lasts for longer than 4.3 time units, the resulting discounted solution quality is worse than if the algorithm just outputs the initial guess it can produce with virtually no computation. It is typical that the solution quality improves in jumps; when the current best solution changes, there is a jump in the quality. However, the penalty associated with waiting is often not as simple as a straight line.

To take into account bounded rationality, an agent must decide whether it should act or think more. This is challenging because an agent typically does not know how much better off it would be if it only spent a little bit more time

Dimension	Values
Modularity	flat, modular, hierarchical
Representation scheme	states, features, relations
Planning horizon	non-planning, finite stage, indefinite stage, infinite stage
Sensing uncertainty	fully observable, partially observable
Effect uncertainty	deterministic, stochastic
Preference	goals, complex preferences
Learning	knowledge is given, knowledge is learned
Number of agents	single agent, multiple agents
Computational limits	perfect rationality, bounded rationality

Figure 1.6: Dimensions of complexity

reasoning. Moreover, the time spent thinking about whether it should reason may detract from actually reasoning about the domain. However, bounded rationality can be the basis for approximate reasoning.

1.5.9 Interaction of the Dimensions

Figure 1.6 summarizes the dimensions of complexity. Unfortunately, we cannot study these dimensions independently because they interact in complex ways. Here we give some examples of the interactions.

The representation dimension interacts with the modularity dimension in that some modules in a hierarchy may be simple enough to reason in terms of a finite set of states, whereas other levels of abstraction may require reasoning about individuals and relations. For example, in a delivery robot, a module that maintains balance may only have a few states. A module that must prioritize the delivery of multiple parcels to multiple people may have to reason about multiple individuals (e.g., people, packages, and rooms) and the relations between them. At a higher level, a module that reasons about the activity over the day may only require a few states to cover the different phases of the day (e.g., there might be three states: busy time, available for requests, and recharge time).

The planning horizon interacts with the modularity dimension. For example, at a high level, a dog may be getting an immediate reward when it comes and gets a treat. At the level of deciding where to place its paws, there may be a long time until it gets the reward, and so at this level it may have to plan for an indefinite stage.

Sensing uncertainty probably has the greatest impact on the complexity of reasoning. It is much easier for an agent to reason when it knows the state of the world than when it doesn't. Although sensing uncertainty with states

is well understood, sensing uncertainty with individuals and relations is an active area of current research.

The effect uncertainty dimension interacts with the modularity dimension: at one level in a hierarchy, an action may be deterministic, whereas at another level, it may be stochastic. As an example, consider the result of flying to Paris with a companion you are trying to impress. At one level you may know where you are (in Paris); at a lower level, you may be quite lost and not know where you are on a map of the airport. At an even lower level responsible for maintaining balance, you may know where you are: you are standing on the ground. At the highest level, you may be very unsure whether you have impressed your companion.

Preference models interact with uncertainty because an agent must have a trade-off between satisfying a major goal with some probability or a less desirable goal with a higher probability. This issue is explored in Section 9.1 (page 373).

Multiple agents can also be used for modularity; one way to design a single agent is to build multiple interacting agents that share a common goal of making the higher-level agent act intelligently. Some researchers, such as Minsky [1986], argue that intelligence is an emergent feature from a "society" of unintelligent agents.

Learning is often cast in terms of learning with features – determining which feature values best predict the value of another feature. However, learning can also be carried out with individuals and relations. Much work has been done on learning hierarchies, learning in partially observable domains, and learning with multiple agents, although each of these is challenging in its own right without considering interactions with multiple dimensions.

Two of these dimensions, modularity and bounded rationality, promise to make reasoning more efficient. Although they make the formalism more complicated, breaking the system into smaller components, and making the approximations needed to act in a timely fashion and within memory limitations, should help build more complex systems.

1.6 Prototypical Applications

AI applications are widespread and diverse and include medical diagnosis, scheduling factory processes, robots for hazardous environments, game playing, autonomous vehicles in space, natural language translation systems, and tutoring systems. Rather than treating each application separately, we abstract the essential features of such applications to allow us to study the principles behind intelligent reasoning and action.

This section outlines four application domains that will be developed in examples throughout the book. Although the particular examples presented are simple – otherwise they would not fit into the book – the application domains

are representative of the range of domains in which AI techniques can be, and are being, used.

The four application domains are as follows:

- An **autonomous delivery robot** roams around a building delivering packages and coffee to people in the building. This delivery agent should be able to find paths, allocate resources, receive requests from people, make decisions about priorities, and deliver packages without injuring people or itself.
- A **diagnostic assistant** helps a human troubleshoot problems and suggests repairs or treatments to rectify the problems. One example is an electrician's assistant that suggests what may be wrong in a house, such as a fuse blown, a light switch broken, or a light burned out, given some symptoms of electrical problems. Another example is a medical diagnostician that finds potential diseases, useful tests, and appropriate treatments based on knowledge of a particular medical domain and a patient's symptoms and history. This assistant should be able to explain its reasoning to the person who is carrying out the tests and repairs and who is ultimately responsible for their actions.
- A **tutoring system** interacts with a student, presenting information about some domain and giving tests of the student's knowledge or performance. This entails more than presenting information to students. Doing what a good teacher does, namely tailoring the information presented to each student based on his or her knowledge, learning preferences, and misunderstandings, is more challenging. The system must understand both the subject matter and how students learn.
- A **trading agent** knows what a person wants and can buy goods and services on her behalf. It should know her requirements and preferences and how to trade off competing objectives. For example, for a family holiday a travel agent must book hotels, airline flights, rental cars, and entertainment, all of which must fit together. It should determine a customer's trade-offs. If the most suitable hotel cannot accommodate the family for all of the days, it should determine whether they would prefer to stay in the better hotel for part of the stay or if they prefer not to move hotels. It may even be able to shop around for specials or to wait until good deals come up.

These four domains will be used for the motivation for the examples in the book. In the next sections, we discuss each application domain in detail.

1.6.1 An Autonomous Delivery Robot

Imagine a robot that has wheels and can pick up objects and put them down. It has sensing capabilities so that it can recognize the objects that it must manipulate and can avoid obstacles. It can be given orders in natural language and obey them, making reasonable choices about what to do when its goals conflict. Such a robot could be used in an office environment to deliver packages, mail, and/or coffee, or it could be embedded in a wheelchair to help disabled people. It should be useful as well as safe.

In terms of the black box characterization of an agent in Figure 1.3 (page 11), the autonomous delivery robot has the following as inputs:

- prior knowledge, provided by the agent designer, about its own capabilities, what objects it may encounter and have to differentiate, what requests mean, and perhaps about its environment, such as a map;
- past experience obtained while acting, for instance, about the effect of its actions, what objects are common in the world, and what requests to expect at different times of the day;
- goals in terms of what it should deliver and when, as well as preferences that specify trade-offs, such as when it must forgo one goal to pursue another, or the trade-off between acting quickly and acting safely; and
- observations about its environment from such input devices as cameras, sonar, touch, sound, laser range finders, or keyboards.

The robot's outputs are motor controls that specify how its wheels should turn, where its limbs should move, and what it should do with its grippers. Other outputs may include speech and a video display.

In terms of the dimensions of complexity, the simplest case for the robot is a flat system, represented in terms of states, with no uncertainty, with achievement goals, with no other agents, with given knowledge, and with perfect rationality. In this case, with an indefinite stage planning horizon, the problem of deciding what to do is reduced to the problem of finding a path in a graph of states. This is explored in Chapter 3.

Each dimension can add conceptual complexity to the task of reasoning:

- A hierarchical decomposition can allow the complexity of the overall system to be increased while allowing each module to be simple and able to be understood by itself. This is explored in Chapter 2.
- Modeling in terms of features allows for a much more comprehensible system than modeling explicit states. For example, there may be features for the robot's location, the amount of fuel it has, what it is carrying, and so forth. Reasoning in terms of the states, where a state is an assignment of a value to each feature, loses the structure that is provided by the features. Reasoning in terms of the feature representation can be exploited for computational gain. Planning in terms of features is discussed in Chapter 8. When dealing with multiple individuals (e.g., multiple people or objects to deliver), it may be easier to reason in terms of individuals and relations. Planning in terms of individuals and relations is explored in Section 14.1 (page 598).
- The planning horizon can be finite if the agent only looks ahead a few steps. The planning horizon can be indefinite if there is a fixed set of goals to achieve. It can be infinite if the agent has to survive for the long term, with ongoing requests and actions, such as delivering mail whenever it arrives and recharging its battery when its battery is low.
- There could be goals, such as "deliver coffee to Chris and make sure you always have power." A more complex goal may be to "clean up the lab, and put everything where it belongs." There can be complex preferences, such as

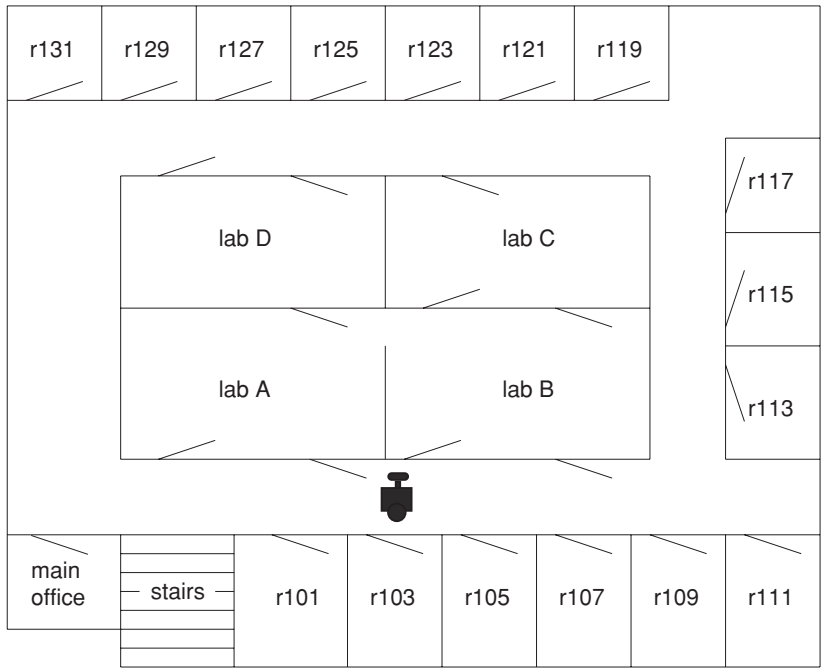

Figure 1.7: An environment for the delivery robot, which shows a typical laboratory environment. This also shows the locations of the doors and which way they open.

"deliver mail when it arrives and service coffee requests as soon as possible, but it is more important to deliver messages marked as important, and Chris really needs her coffee quickly when she asks for it."

- There can be sensing uncertainty because the robot does not know what is in the world based on its limited sensors.
- There can be uncertainty about the effects of an action, both at the low level, such as due to slippage of the wheels, or at the high level in that the agent might not know if putting the coffee on Chris's desk succeeded in delivering coffee to her.
- There can be multiple robots, which can coordinate to deliver coffee and parcels and compete for power outlets. There may also be children out to trick the robot.
- A robot has lots to learn, such as how slippery floors are as a function of their shininess, where Chris hangs out at different parts of the day and when she will ask for coffee, and which actions result in the highest rewards.

Figure 1.7 depicts a typical laboratory environment for a delivery robot. This environment consists of four laboratories and many offices. The robot can only push doors, and the directions of the doors in the diagram reflect the directions in which the robot can travel. Rooms require keys, and those keys can be obtained from various sources. The robot must deliver parcels,

beverages, and dishes from room to room. The environment also contains a stairway that is potentially hazardous to the robot.

1.6.2 A Diagnostic Assistant

A **diagnostic assistant** is intended to advise a human about some particular system such as a medical patient, the electrical system in a house, or an automobile. The diagnostic assistant should advise about potential underlying faults or diseases, what tests to carry out, and what treatment to prescribe. To give such advice, the assistant requires a model of the system, including knowledge of potential causes, available tests, and available treatments, and observations of the system (which are often called symptoms).

To be useful, the diagnostic assistant must provide added value, be easy for a human to use, and not be more trouble than it is worth. A diagnostic assistant connected to the Internet can draw on expertise from throughout the world, and its actions can be based on the most up-to-date research. However, it must be able to justify why the suggested diagnoses or actions are appropriate. Humans are, and should be, suspicious of computer systems that are opaque and impenetrable. When humans are responsible for what they do, even if it is based on a computer system's advice, they should have reasonable justifications for the suggested actions.

In terms of the black box definition of an agent in Figure 1.3 (page 11), the diagnostic assistant has the following as inputs:

- prior knowledge, such as how switches and lights normally work, how diseases or malfunctions manifest themselves, what information tests provide, and the effects of repairs or treatments.
- past experience, in terms of data of previous cases that include the effects of repairs or treatments, the prevalence of faults or diseases, the prevalence of symptoms for these faults or diseases, and the accuracy of tests. These data are usually about similar artifacts or patients, rather than the actual one being diagnosed.
- goals of fixing the device and trade-offs, such as between fixing or replacing different components, or whether patients prefer to live longer if it means they will be in pain or be less coherent.
- observations of symptoms of a device or patient.

The output of the diagnostic assistant is in terms of recommendations of treatments and tests, along with a rationale for its recommendations.

Example 1.10 Figure 1.8 (on the next page) shows a depiction of an electrical distribution system in a house. In this house, power comes into the house through circuit breakers and then it goes to power outlets or to lights through light switches. For example, light l_1 is on if there is power coming into the house, if circuit breaker cb_1 is *on*, and if switches s_1 and s_2 are either both up or both down. This is the sort of model that normal householders may have of the electrical power in the house, and which they could use to determine what is

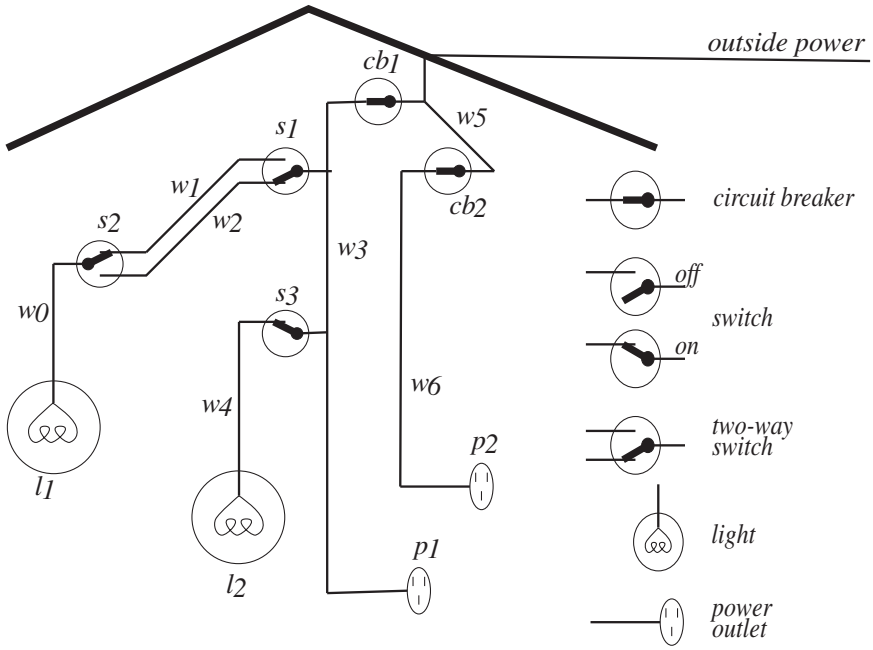

Figure 1.8: An electrical environment for the diagnostic assistant

wrong given evidence about the position of the switches and which lights are on and which are off. The diagnostic assistant is there to help a householder or an electrician troubleshoot electrical problems.

Each dimension is relevant to the diagnostic assistant:

- Hierarchical decomposition allows for very-high-level goals to be maintained while treating the lower-level causes and allows for detailed monitoring of the system. For example, in a medical domain, one module could take the output of a heart monitor and give higher-level observations such as notifying when there has been a change in the heart rate. Another module could take in this observation and other high-level observations and notice what other symptoms happen at the same time as a change in heart rate. In the electrical domain, Figure 1.8 is at one level of abstraction; a lower level could specify the voltages, how wires are spliced together, and the internals of switches.
- Most systems are too complicated to reason about in terms of the states, and so they are usually described in terms of the features or individual components and relations among them. For example, a human body may be described in terms of the values for features of its various components. Designers may want to model the dynamics without knowing the actual individuals. For example, designers of the electrical diagnosis system would model how lights and switches work before knowing which lights and switches exist in an actual house and, thus, before they know the features.

This can be achieved by modeling in terms of relations and their interaction and by adding the individual components when they become known.

- It is possible to reason about a static system, such as reasoning about what could be wrong when a light is off given the position of switches. It is also possible to reason about a sequence of tests and treatments, where the agents keep testing and treating until the problem is fixed, or where the agent carries out ongoing monitoring of a system, continuously fixing whatever gets broken.
- Sensing uncertainty is the fundamental problem that faces diagnosis. Diagnosis is required if an agent cannot directly observe the internals of the system.
- Effect uncertainty also exists in that an agent may not know the outcome of a treatment and, often, treatments have unanticipated outcomes.
- The goal may be as simple as "fix what is wrong," but often there are complex trade-offs involving costs, pain, life expectancy, the probability that the diagnosis is correct, and the uncertainty as to efficacy and side effects of the treatment.
- Although it is often a single-agent problem, diagnosis becomes more complicated when multiple experts are involved who perhaps have competing experience and models. There may be other patients with whom an agent must compete for resources (e.g., doctor's time, surgery rooms).
- Learning is fundamental to diagnosis. It is through learning that we understand the progression of diseases and how well treatments work or do not work. Diagnosis is a challenging domain for learning, because all patients are different, and each individual doctor's experience is only with a few patients with any particular set of symptoms. Doctors also see a biased sample of the population; those who come to see them usually have unusual or painful symptoms.
- Diagnosis often requires a quick response, which may not allow for the time to carry out exhaustive reasoning or perfect rationality.

1.6.3 An Intelligent Tutoring System

An **intelligent tutoring system** is a computer system that tutors students in some domain of study.

For example, in a tutoring system to teach elementary physics, such as mechanics, the system may present the theory and worked-out examples. The system can ask the student questions and it must be able to understand the student's answers, as well as determine the student's knowledge based on what answers were given. This should then affect what is presented and what other questions are asked of the student. The student can ask questions of the system, and so the system should be able to solve problems in the physics domain.

In terms of the black box definition of an agent in Figure 1.3 (page 11), an intelligent tutoring system has the following as inputs:

- prior knowledge, provided by the agent designer, about the subject matter being taught, teaching strategies, possible errors, and misconceptions of the students.
- past experience, which the tutoring system has acquired by interacting with students, about what errors students make, how many examples it takes to learn something, and what students forget. This can be information about students in general or about a particular student.
- preferences about the importance of each topic, the level of achievement of the student that is desired, and costs associated with usability. There are often complex trade-offs among these.
- observations of a student's test results and observations of the student's interaction (or non-interaction) with the system. Students can also ask questions or provide new examples with which they want help.

The output of the tutoring system is the information presented to the student, tests the students should take, answers to questions, and reports to parents and teachers.

Each dimension is relevant to the tutoring system:

- There should be both a hierarchical decomposition of the agent and a decomposition of the task of teaching. Students should be taught the basic skills before they can be taught higher-level concepts. The tutoring system has high-level teaching strategies, but, at a much lower level, it must design the details of concrete examples and specific questions for a test.
- A tutoring system may be able to reason in terms of the state of the student. However, it is more realistic to have multiple features for the student and the subject domain. A physics tutor may be able to reason in terms of features that are known at design time if the examples are fixed and it is only reasoning about one student. For more complicated cases, the tutoring system should refer to individuals and relations. If the tutoring system or the student can create examples with multiple individuals, the system may not know the features at design time and will have to reason in terms of individuals and relations.
- In terms of planning horizon, for the duration of a test, it may be reasonable to assume that the domain is static and that the student does not learn while taking a test. For some subtasks, a finite horizon may be appropriate. For example, there may be a teach, test, reteach sequence. For other cases, there may be an indefinite horizon where the system may not know at design time how many steps it will take until the student has mastered some concept. It may also be possible to model teaching as an ongoing process of learning and testing with appropriate breaks, with no expectation of the system finishing.
- Uncertainty will have to play a large role. The system cannot directly observe the knowledge of the student. All it has is some sensing input, based

on questions the student asks or does not ask, and test results. The system will not know for certain the effect of a particular teaching episode.

- Although it may be possible to have a simple goal such as to teach some particular concept, it is more likely that complex preferences must be taken into account. One reason is that, with uncertainty, there may be no way to guarantee that the student knows the concept being taught; any method that tries to maximize the probability that the student knows a concept will be very annoying, because it will continue to repeatedly teach and test if there is a slight chance that the student's errors are due to misunderstanding as opposed to fatigue or boredom. More complex preferences would enable a trade-off among fully teaching a concept, boring the student, the time taken, and the amount of retesting. The user may also have a preference for a teaching style that should be taken into account.
- It may be appropriate to treat this as a single-agent problem. However, the teacher, the student, and the parent may all have different preferences that must be taken into account. Each of these agents may act strategically by not telling the truth.
- We would expect the system to be able to learn about what teaching strategies work, how well some questions work at testing concepts, and what common mistakes students make. It could learn general knowledge, or knowledge particular to a topic (e.g., learning about what strategies work for teaching mechanics) or knowledge about a particular student, such as learning what works for Sam.
- One could imagine that choosing the most appropriate material to present would take a lot of computation time. However, the student must be responded to in a timely fashion. Bounded rationality would play a part in ensuring that the system does not compute for a long time while the student is waiting.

1.6.4 A Trading Agent

A **trading agent** is like a robot, but instead of interacting with a physical environment, it interacts with an information environment. Its task is to procure goods and services for a user. It must be able to be told the needs of a user, and it must interact with sellers (e.g., on the Web). The simplest trading agent involves proxy bidding for a user on an auction site, where the system will keep bidding until the user's price limit is reached. A more complicated trading agent will buy multiple complementary items, like booking a flight, a hotel, and a rental car that fit together, in addition to trading off competing preferences. Another example of a trading agent is one that monitors how much food and groceries are in a household, monitors the prices, and orders goods before they are needed, keeping costs to a minimum.

In terms of the black box definition of an agent in Figure 1.3 (page 11), the trading agent has the following as inputs:

- prior knowledge about types of goods and services, selling practices, and how auctions work;

- past experience about where is the best place to look for specials, how prices vary in time in an auction, and when specials tend to turn up;
- preferences in terms of what the user wants and how to trade off competing goals; and
- observations about what items are available, their price, and, perhaps, how long they are available.

The output of the trading agent is either a proposal to the user that they can accept or reject or an actual purchase.

The trading agent should take all of the dimensions into account:

- Hierarchical decomposition is essential because of the complexity of domains. Consider the problem of making all of the arrangements and purchases for a custom holiday for a traveler. It is simpler to have a module that can purchase a ticket and optimize connections and timing, rather than to do this at the same time as determining what doors to go through to get to the taxi stand.
- The state space of the trading agent is too large to reason in terms of individual states. There are also too many individuals to reason in terms of features. The trading agent will have to reason in terms of individuals such as customers, days, hotels, flights, and so on.
- A trading agent typically does not make just one purchase, but must make a sequence of purchases, often a large number of sequential decisions (e.g., purchasing one hotel room may require booking ground transportation, which may in turn require baggage storage), and often plans for ongoing purchasing, such as for an agent that makes sure a household has enough food on hand at all times.
- There is often sensing uncertainty in that a trading agent does not know all of the available options and their availability, but must find out information that can become old quickly (e.g., if some hotel becomes booked up). A travel agent does not know if a flight will be canceled or delayed or whether the passenger's luggage will be lost. This uncertainty means that the agent must plan for the unanticipated.
- There is also effect uncertainty in that the agent does not know if an attempted purchase will succeed.
- Complex preferences are at the core of the trading agent. The main problem is in allowing users to specify what they want. The preferences of users are typically in terms of functionality, not components. For example, typical computer buyers have no idea of what hardware to buy, but they know what functionality they want and they also want the flexibility to be able to use new features that might not yet exist. Similarly, in a travel domain, what activities a user may want may depend on the location. Users also may want the ability to participate in a local custom at their destination, even though they may not know what these customs are.
- A trading agent has to reason about other agents. In commerce, prices are governed by supply and demand; this means that it is important to reason about the other competing agents. This happens particularly in a world

where many items are sold by auction. Such reasoning becomes particularly difficult when there are items that must complement each other, such as flights and hotel booking, and items that can substitute for each other, such as bus transport or taxis.

- A trading agent should learn about what items sell quickly, which of the suppliers are reliable, where to find good deals, and what unanticipated events may occur.
- A trading agent faces severe communication limitations. In the time between finding that some item is available and coordinating the item with other items, the item may have sold out. This can sometimes be alleviated by sellers agreeing to hold some items (not to sell them to someone else in the meantime), but sellers will not be prepared to hold an item indefinitely if others want to buy it.

Because of the personalized nature of the trading agent, it should be able to do better than a generic purchaser that, for example, only offers packaged tours.

1.7 Overview of the Book

The rest of the book explores the design space defined by the dimensions of complexity. It considers each dimension separately, where this can be done sensibly.

Chapter 2 analyzes what is inside the black box of Figure 1.3 (page 11) and discusses the modular and hierarchical decomposition of intelligent agents.

Chapter 3 considers the simplest case of determining what to do in the case of a single agent that reasons with explicit states, no uncertainty, and has goals to be achieved, but with an indefinite horizon. In this case, the problem of solving the goal can be abstracted to searching for a path in a graph. It is shown how extra knowledge of the domain can help the search.

Chapters 4 and 5 show how to exploit features. In particular, Chapter 4 considers how to find possible states given constraints on the assignments of values to features represented as variables. Chapter 5 shows how to determine whether some proposition must be true in all states that satisfy a given set of constraints.

Chapter 6 shows how to reason with uncertainty.

Chapter 7 shows how an agent can learn from past experiences and data. It covers the most common case of learning, namely supervised learning with features, where a set of observed target features are being learned.

Chapter 8 considers the problem of planning, in particular representing and reasoning with feature-based representations of states and actions. Chapter 9 considers the problem of planning with uncertainty, and Chapter 10 expands the case to multiple agents.

Chapter 11 introduces learning under uncertainty and reinforcement learning.

Chapter 12 shows how to reason in terms of individuals and relations. Chapter 13 discusses ontologies and how to build knowledge-based systems. Chapter 14 shows how reasoning about individuals and relations can be combined with planning, learning, and probabilistic reasoning.

Chapter 15 reviews the design space of AI and shows how the material presented can fit into that design space. It also presents ethical considerations involved in building intelligent agents.

1.8 Review

The following are the main points you should have learned from this chapter:

- Artificial intelligence is the study of computational agents that act intelligently.
- An agent acts in an environment and only has access to its prior knowledge, its history of observations, and its goals and preferences.
- An intelligent agent is a physical symbol system that manipulates symbols to determine what to do.
- A designer of an intelligent agent should be concerned about modularity, how to describe the world, how far ahead to plan, uncertainty in both perception and the effects of actions, the structure of goals or preferences, other agents, how to learn from experience, and the fact that all real agents have limited computational resources.
- To solve a problem by computer, the computer must have an effective representation with which to reason.
- To know when you have solved a problem, an agent must have a definition of what constitutes an adequate solution, such as whether it has to be optimal, approximately optimal, or almost always optimal, or whether a satisficing solution is adequate.
- In choosing a representation, you should find a representation that is as close as possible to the problem, so that it is easy to determine what it is representing and so it can be checked for correctness and be able to be maintained. Often, users want an explanation of why they should believe the answer.

1.9 References and Further Reading

The ideas in this chapter have been derived from many sources. Here, we will try to acknowledge those that are explicitly attributable to particular authors. Most of the other ideas are part of AI folklore; trying to attribute them to anyone would be impossible.

Haugeland [1997] contains a good collection of articles on the philosophy behind artificial intelligence, including that classic paper of Turing [1950] that proposes the Turing test. Cohen [2005] gives a recent discussion of the Turing test.

Nilsson [2009] gives a detailed description of the history of AI. Chrisley and Begeer [2000] present many classic papers on AI.

The physical symbol system hypothesis was posited by Newell and Simon [1976]. See also Simon [1996], who discusses the role of symbol systems in a multidisciplinary context. The distinctions between real, synthetic, and artificial intelligence are discussed by Haugeland [1985], who also provides useful introductory material on interpreted, automatic formal symbol systems and the Church–Turing thesis. For a critique of the symbol-system hypothesis see Brooks [1990] and Winograd [1990]. Nilsson [2007] evaluates the hypothesis in terms of recent criticisms.

The use of anytime algorithms is due to Horvitz [1989] and Boddy and Dean [1994]. See Dean and Wellman [1991, Chapter 8], Zilberstein [1996], and Russell [1997] for introductions to bounded rationality.

For discussions on the foundations of AI and the breadth of research in AI see Kirsh [1991a], Bobrow [1993], and the papers in the corresponding volumes, as well as Schank [1990] and Simon [1995]. The importance of knowledge in AI is discussed in Lenat and Feigenbaum [1991] and Smith [1991].

For overviews of cognitive science and the role that AI and other disciplines play in that field, see Gardner [1985], Posner [1989], and Stillings, Feinstein, Garfield, Rissland, Rosenbaum, Weisler, and Baker-Ward [1987].

Purchasing agents can become very complex. Sandholm [2007] describes how AI can be used for procurement of multiple goods with complex preferences.

A number of AI texts are valuable as reference books complementary to this book, providing a different perspective on AI. In particular, Russell and Norvig [2010] give a more encyclopedic overview of AI and provide a complementary source for many of the topics covered in this book. They provide an outstanding review of the scientific literature, which we do not try to duplicate.

The *Encyclopedia of Artificial Intelligence* [Shapiro, 1992] is an encyclopedic reference on AI written by leaders in the field and still provides background on some of the classic topics. There are also a number of collections of classic research papers. The general collections of most interest to readers of this book include Webber and Nilsson [1981] and Brachman and Levesque [1985]. More specific collections are given in the appropriate chapters.

The Association for the Advancement of Artificial Intelligence (AAAI) provides introductory material and news at their *AI Topics* web site (http://www.aaai.org/AITopics/html/welcome.html). *AI Magazine*, published by AAAI, often has excellent overview articles and descriptions of particular applications. *IEEE Intelligent Systems* also provides accessible articles on AI research.

There are many journals that provide in-depth research contributions and conferences where the most up-to-date research is found. These include the journals *Artificial Intelligence*, the *Journal of Artificial Intelligence Research*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and *Computational Intelligence*, as well as more specialized journals such as *Neural Computation*, *Computational Linguistics*, *Machine Learning*, the *Journal of Automated Reasoning*, the

Journal of Approximate Reasoning, *IEEE Transactions on Robotics and Automation*, and the *Theory and Practice of Logic Programming*. Most of the cutting-edge research is published first in conferences. Those of most interest to a general audience are the biennial International Joint Conference on Artificial Intelligence (IJCAI), the AAAI Annual Conference, the European Conference on AI (ECAI), the Pacific Rim International Conference on AI (PRICAI), various national conferences, and many specialized conferences and workshops.

1.10 Exercises

Exercise 1.1 For each of the following, give five reasons why:

(a) A dog is more intelligent than a worm.

(b) A human is more intelligent than a dog.

(c) An organization is more intelligent than an individual human.

Based on these, give a definition of what "more intelligent" may mean.

Exercise 1.2 Give as many disciplines as you can whose aim is to study intelligent behavior of some sort. For each discipline, find out what aspect of behavior is investigated and what tools are used to study it. Be as liberal as you can regarding what defines intelligent behavior.

Exercise 1.3 Find out about two applications of AI (not classes of applications, but specific programs). For each application, write, at most, one typed page describing it. You should try to cover the following questions:

(a) What does the application actually do (e.g., control a spacecraft, diagnose a photocopier, provide intelligent help for computer users)?

(b) What AI technologies does it use (e.g., model-based diagnosis, belief networks, semantic networks, heuristic search, constraint satisfaction)?

(c) How well does it perform? (According to the authors or to an independent review? How does it compare to humans? How do the authors know how well it works?)

(d) Is it an experimental system or a fielded system? (How many users does it have? What expertise do these users require?)

(e) Why is it intelligent? What aspects of it makes it an intelligent system?

(f) [optional] What programming language and environment was it written in? What sort of user interface does it have?

(g) References: Where did you get the information about the application? To what books, articles, or web pages should others who want to know about the application refer?

Exercise 1.4 Choose four pairs of dimensions that were not covered in the book. For each pair, give one commonsense example of where the dimensions interact.