Module 1

- **Artificial intelligence (AI) is a term to describe a branch of computer science that is dedicated to creating intelligent machines that would learn to work and react like humans.**

- **AI which stands for artificial intelligence refers to systems or machines that mimic human intelligence to perform tasks and can iteratively improve themselves based on the information they collect.**

1.1 WHAT IS AI?

- The field of artificial intelligence, or AI, it attempts not just to understand but also to build intelligent entities.
- AI currently encompasses a huge variety of subfields, ranging from the general (learning and perception) to the specific, such as playing chess, proving mathematical theorems, writing poetry, driving a car on a crowded street, and diagnosing diseases.
- AI is relevant to any intellectual task; it is truly a universal field.
- In Figure 1.1 we see eight definitions of AI, laid out along two dimensions.
- The definitions on top are concerned with thought processes and reasoning, whereas the ones on the bottom address behavior.
- The definitions on the left measure success in terms of fidelity to human performance, whereas RATIONALITY the ones on the right measure against an ideal performance measure, called rationality.
- A system is rational if it does the "right thing," given what it knows. Historically, all four approaches to AI have been followed, each by different people with different methods.
- A human-centered approach must be in part an empirical science, involving observations and hypotheses about human behavior.
- A rationalist approach involves a combination of mathematics and engineering.

Thinking Humanly	**Thinking Rationally**
"The exciting new effort to make computers think . . . *machines with minds*, in the full and literal sense." (Haugeland, 1985) "[The automation of] activities that we associate with human thinking, activities such as decision-making, problem solving, learning . . ." (Bellman, 1978)	"The study of mental faculties through the use of computational models." (Charniak and McDermott, 1985) "The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)
Acting Humanly	**Acting Rationally**
"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990) "The study of how to make computers do things at which, at the moment, people are better." (Rich and Knight, 1991)	"Computational Intelligence is the study of the design of intelligent agents." (Poole *et al.*, 1998) "AI . . . is concerned with intelligent behavior in artifacts." (Nilsson, 1998)

Figure 1.1 Some definitions of artificial intelligence, organized into four categories.

Four approaches to AI:

1. Acting humanly: The Turing Test approach
2. Thinking humanly: The cognitive modeling approach
3. Thinking rationally: The "laws of thought" approach
4. Acting rationally: The rational agent approach

1.1.1 Acting humanly: The Turing Test approach

- The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence.
- A computer passes the test if a human interrogator, after posing some written questions, cannot tell whether the written responses come from a person or from a computer.
- The computer would need to possess the following capabilities:
  - **natural language processing** to enable it to communicate successfully in English;
  - **knowledge representation** to store what it knows or hears;
  - **automated reasoning** to use the stored information to answer questions and to draw new conclusions;
  - **machine learning** to adapt to new circumstances and to detect and extrapolate patterns.

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

- To pass the total Turing Test, the computer will need
  - COMPUTER VISION: computer vision to perceive objects, and
  - ROBOTICS: robotics to manipulate objects and move about.

1.1.2 Thinking humanly: The cognitive modeling approach

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

experimental techniques from psychology to construct precise and testable theories of the human mind.

- The two fields continue to fertilize each other, most notably in computer vision, which incorporates neurophysiological evidence into computational models.

1.1.3 Thinking rationally: The "laws of thought" approach

- The Greek philosopher Aristotle was one of the first to attempt to codify "right thinking," that is, irrefutable reasoning processes.
- His syllogisms provided patterns for argument structures that always yielded correct conclusions when given correct premises—for example, "Socrates is a man; all men are mortal; therefore, Socrates is mortal." These laws of thought were supposed to govern the operation of the mind; their study initiated the field called logic.
- Logicians in the 19th century developed a precise notation for statements about all kinds of objects in the world and the relations among them. (Contrast this with ordinary arithmetic notation, which provides only for statements about numbers.)
- The so-called logicist tradition within artificial intelligence hopes to build on such programs to create intelligent systems.
- There are two main obstacles to this approach:
  → First, it is not easy to take informal knowledge and state it in the formal terms required by logical notation, particularly when the knowledge is less than 100% certain.
  →Second, there is a big difference between solving a problem "in principle" and solving it in practice.

1.1.4 Acting rationally: The rational agent approach

- **An agent** is just something that acts. All computer programs do something, but computer agents are expected to do more: operate autonomously, perceive their environment, persist over a prolonged time period, adapt to change, and create and pursue goals.
- **A rational agent** is one that acts so as to achieve the best outcome or, when there is uncertainty, the best expected outcome.
- Making correct inferences is sometimes part of being a rational agent, because one way to act rationally is to reason logically to the conclusion that a given action will achieve one's goals and then to act on that conclusion.
- On the other hand, correct inference is not all of rationality; in some situations, there is no provably correct thing to do, but something must still be done.
- All the skills needed for the Turing Test also allow an agent to act rationally.
- Knowledge representation and reasoning enable agents to reach good decisions.
- **The rational-agent approach has two advantages over the other approaches:**
  →First, it is more general than the "**laws of thought**" approach because correct inference is just one of several possible mechanisms for achieving rationality.
  →Second, it is more amenable to scientific development than are approaches based on human behavior or human thought.
- The standard of rationality is mathematically well defined and completely general, and can be "unpacked" to generate agent designs that provably achieve it.

- Human behavior, on the other hand, is well adapted for one specific environment and is defined by, well, the sum total of all the things that humans do.

1.4 THE STATE OF THE ART

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

- **Robotic vehicles:**
  - A driverless robotic car named STANLEY sped through the rough terrain of the Mojave dessert at 22 mph, finishing the 132-mile course first to win the 2005 DARPAGrand Challenge.
  - STANLEY is a Volkswagen Touareg outfitted with cameras, radar, and laser rangefinders to sense the environment and onboard software to command the steer ing, braking, and acceleration (Thrun, 2006).
  - The following year CMU's BOSS won the Ur ban Challenge, safely driving in traffic through the streets of a closed Air Force base, obeying traffic rules and avoiding pedestrians and other vehicles.
- **Speech recognition:**
  - A traveler calling United Airlines to book a flight can have the en tire conversation guided by an automated speech recognition and dialog management system.
- **Autonomous planning and scheduling:**
  - A hundred million miles from Earth, NASA's Remote Agent program became the first on-board autonomous planning program to control the scheduling of operations for a spacecraft (Jonsson et al., 2000).
  - REMOTE AGENT gen erated plans from high-level goals specified from the ground and monitored the execution of those plans—detecting, diagnosing, and recovering from problems as they occurred.
  - Successor program MAPGEN (Al-Chang et al., 2004) plans the daily operations for NASA's Mars Exploration Rovers, and MEXAR2 (Cesta et al., 2007) did mission planning—both logistics and science planning—for the European Space Agency's Mars Express mission in 2008.
- **Game playing:**
  - IBM's DEEP BLUE became the first computer program to defeat the world champion in a chess match when it bested Garry Kasparov by a score of 3.5 to 2.5 in an exhibition match (Goodman and Keene, 1997).
  - Kasparov said that he felt a "new kind of intelligence" across the board from him. Newsweek magazine described the match as "The brain's last stand."
  - The value of IBM's stock increased by $18 billion. Human champions studied Kasparov's loss and were able to draw a few matches in subsequent years, but the most recent human-computer matches have been won convincingly by the computer.
- **Spam fighting:**
  - Each day, learning algorithms classify over a billion messages as spam, saving the recipient from having to waste time deleting what, for many users, could comprise 80%

or 90% of all messages, if not classified away by algorithms.

- Because the spammers are continually updating their tactics, it is difficult for a static programmed approach to keep up, and learning algorithms work best (Sahami et al., 1998; Goodman and Heckerman, 2004).

- **Logistics planning:**
  - During the Persian Gulf crisis of 1991, U.S. forces deployed a Dynamic Analysis and Replanning Tool, DART (Cross and Walker, 1994), to do automated logistics planning and scheduling for transportation.
  - This involved up to 50,000 vehicles, cargo, and people at a time, and had to account for starting points, destinations, routes, and conflict resolution among all parameters.
  - The AI planning techniques generated in hours a plan that would have taken weeks with older methods.
  - The Defense Advanced Research Project Agency (DARPA) stated that this single application more than paid back DARPA's 30-year investment in AI.
- **Robotics:**
  - The iRobot Corporation has sold over two million Roomba robotic vacuum cleaners for home use.
  - The company also deploys the more rugged PackBot to Iraq and Afghanistan, where it is used to handle hazardous materials, clear explosives, and identify the location of snipers.

- **Machine Translation:**
  - A computer program automatically translates from Arabic to English, allowing an English speaker to see the headline "Ardogan Confirms That Turkey Would Not Accept Any Pressure, Urging Them to Recognize Cyprus."
  - The program uses a statistical model built from examples of Arabic-to-English translations and from examples of English text totaling two trillion words (Brants et al., 2007).
  - None of the computer scientists on the team speak Arabic, but they do understand statistics and machine learning algorithms.

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

2. Intelligent Agents

2.1 AGENTS AND ENVIRONMENTS

- **An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators.**
- This simple idea is illustrated in Figure 2.1. A human agent has eyes, ears, and other organs for sensors and hands, legs, vocal tract, and so on for actuators.
- A robotic agent might have cameras and infrared range finders for sensors and various motors for actuators.
- A software agent receives keystrokes, file contents, and network packets as sensory inputs and acts on the environment by displaying on the screen, writing files, and sending network packets.
- We use the term percept to refer to the agent's perceptual inputs at any given instant.
- An agent's percept sequence is the complete history of everything the agent has ever perceived.
- In general, an agent's choice of action at any given instant can depend on the entire percept sequence observed to date, but not on anything it hasn't perceived.
- By specifying the agent's choice of action for every possible percept sequence, we have said more or less everything there is to say about the agent.
- Mathematically speaking, we say that an agent's behavior is described by the agent function that maps any given percept sequence to an action.

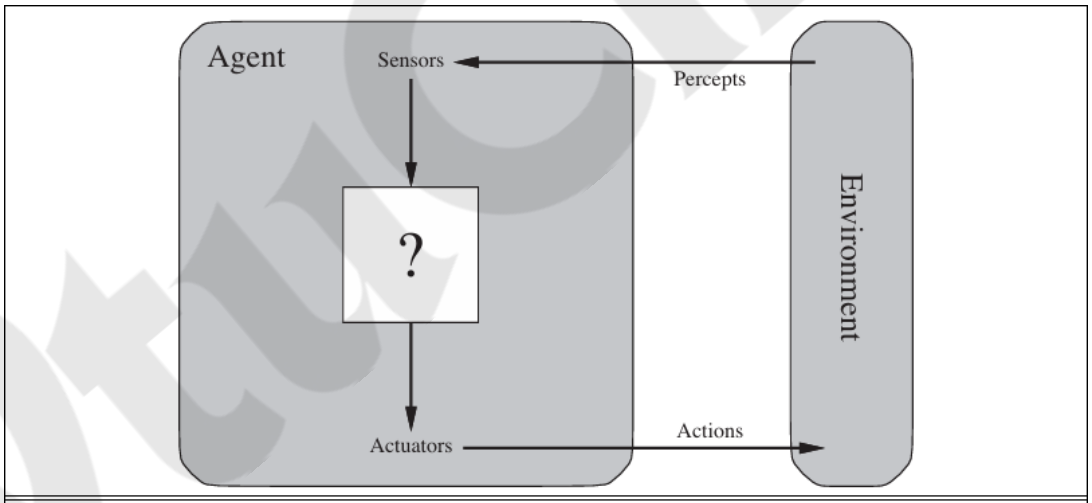

Figure 2.1 Agents interact with environments through sensors and actuators.

- **The agent function** is an abstract mathematical description**; the agent program** is a concrete implementation, running within some physical system.

The vacuum-cleaner world:

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

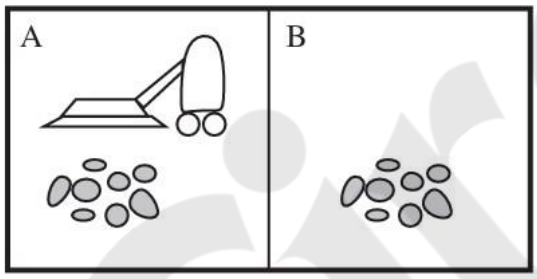

Figure 2.2 A vacuum-cleaner world with just two locations.

Percept sequence	Action
$[A, Clean]$	*Right*
$[A, Dirty]$	*Suck*
$[B, Clean]$	*Left*
$[B, Dirty]$	*Suck*
$[A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Dirty]$	*Suck*
⋮	⋮
$[A, Clean], [A, Clean], [A, Clean]$	*Right*
$[A, Clean], [A, Clean], [A, Dirty]$	*Suck*
⋮	⋮

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

GOOD BEHAVIOR: THE CONCEPT OF RATIONALITY

A rational agent is one that does the right thing—conceptually speaking, every entry in the table for the agent function is filled out correctly. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

- When an agent is plunked down in an environment, it generates a sequence of actions according to the percepts it receives.
- This sequence of actions causes the environment to go through a sequence of states.
- If the sequence is desirable, then the agent has performed well.
- This notion of desirability is captured by a performance measure that evaluates any given sequence of environment states.

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

Rationality

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

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

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

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

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

Let us assume the following:
• The performance measure awards one point for each clean square at each time step, over a "lifetime" of 1000-time steps.
• The "geography" of the environment is known apriori (Figure 2.2) but the dirt distri bution and the initial location of the agent are not. Clean squares stay clean and sucking cleans the current

square. The Left and Right actions move the agent left and right except when this would take the agent outside the environment, in which case the agent remains where it is.
• The only available actions are Left, Right, and Suck.
• The agent correctly perceives its location and whether that location contains dirt.

Omniscience, learning, and autonomy

- An omniscient agent knows the actual outcome of its actions and can act accordingly; but omniscience is impossible in reality.
- Rationality maximizes expected performance, while perfection maximizes actual performance.
- Doing actions in order to modify future percepts—sometimes called information gathering—is an important part of rationality.
- A second example of information gathering is provided by the exploration that must be undertaken by a vacuum-cleaning agent in an initially unknown environment.
- To the extent that an agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks autonomy. A rational agent should be autonomous—it should learn what it can to compensate for partial or incorrect prior knowledge. For example, a vacuum-cleaning agent that learns to foresee where and when additional dirt will appear will do better than one that does not.

Specifying the task environment

PEAS (Performance, Environment, Actuators, Sensors)

- The vacuum world was a simple example; let us consider a more complex problem: an automated taxi driver.
- We should point out, before the reader becomes alarmed, that a fully automated taxi is currently somewhat beyond the capabilities of existing technology.
- The full driving task is extremely open-ended.
- There is no limit to the novel combinations of circumstances that can arise—another reason we chose it as a focus for discussion.
- Figure 2.4 summarizes the PEAS description for the taxi's task environment. We discuss each element in more detail in the following paragraphs.

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

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

- First, what is the performance measure to which we would like our automated driver to aspire?
    - Desirable qualities include getting to the correct destination;
    - Minimizing fuel consumption and wear and tear;
    - Minimizing the trip time or cost;
    - Minimizing violations of traffic laws and disturbances to other drivers; maximizing safety and passenger comfort;
    - Maximizing profits.

- Next, what is the driving environment that the taxi will face?
    - ✓ Any taxi driver must deal with a variety of roads, ranging from rural lanes and urban alleys to 12-lane freeways.
    - ✓ The roads contain other traffic, pedestrians, stray animals, road works, police cars, puddles and potholes.
    - ✓ The taxi must also interact with potential and actual passengers. There are also some optional choices.
    - ✓ The taxi might need to operate in Southern California, where snow is seldom a problem, or in Alaska, where it seldom is not.
    - ✓ It could always be driving on the right, or we might want it to be flexible enough to drive on the left when in Britain or Japan.
    - ✓ Obviously, the more restricted the environment, the easier the design problem.

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

- The basic **sensors** for the taxi will include one or more controllable video cameras so that it can see the road; it might augment these with infrared or sonar sensors to detect distances to other cars and obstacles.
    - ✓ To avoid speeding tickets, the taxi should have a speedometer, and to control the vehicle properly, especially on curves, it should have an accelerometer.
    - ✓ To determine the mechanical state of the vehicle, it will need the usual array of engine, fuel, and electrical system sensors.
    - ✓ Like many human drivers, it might want a global positioning system (GPS) so that it doesn't get lost.
    - ✓ Finally, it will need a keyboard or microphone for the passenger to request a destination.

In Figure 2.5, we have sketched the basic PEAS elements for a number of additional agent types.

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

Figure 2.5 Examples of agent types and their PEAS descriptions.

Fully observable vs. partially observable:

- ✓ If an agent's sensors give it access to the complete state of the environment at each point in time, then we say that the task environment is fully observable.
- ✓ A task environment is effectively fully observable if the sensors detect all aspects that are relevant to the choice of action; relevance, in turn, depends on the performance measure.
- ✓ Fully observable environments are convenient because the agent need not maintain any internal state to keep track of the world.
- ✓ An environment might be partially observable because of noisy and inaccurate sensors or because parts of the state are simply missing from the sensor data—for example, a vacuum agent with only a local dirt sensor cannot tell whether there is dirt in other squares, and an automated taxi cannot see what other drivers are thinking.
- ✓ If the agent has no sensors at all then the environment is unobservable.

Single agent vs. multiagent:

- ✓ The distinction between single-agent and multiagent en vironments may seem simple enough.
- ✓ For example, an agent solving a crossword puzzle by itself is clearly in a single-agent environment, whereas an agent playing chess is in a two agent environment.

- ✓ There are, however, some subtle issues. First, we have described how an entity may be viewed as an agent, but we have not explained which entities must be viewed as agents. Does an agent A (the taxi driver for example) have to treat an object B (another vehicle) as an agent, or can it be treated merely as an object behaving according to the laws of physics, analogous to waves at the beach or leaves blowing in the wind?
- ✓ The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior.
- ✓ For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's per formance measure. Thus, chess is a competitive multiagent environment.
- ✓ In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measure of all agents, so it is a partially cooperative multiagent environment.
- ✓ It is also partially competitive because, for example, only one car can occupy a parking space.
- ✓ The agent-design problems in multiagent environments are often quite different from those in single-agent environments; for example, communication often emerges as a rational behavior in multiagent environments; in some competitive environments, randomized behavior is rational because it avoids the pitfalls of predictability.

Deterministic vs. stochastic:

- ✓ If the next state of the environment is completely determined by the current state and the action executed by the agent, then we say the environment is deterministic; otherwise, it is stochastic.
- ✓ In principle, an agent need not worry about uncertainty in a fully observable, deterministic environment.
- ✓ If the environment is partially observable, however, then it could appear to be stochastic. Most real situations are so complex that it is impossible to keep track of all the unobserved aspects; for practical purposes, they must be treated as stochastic.
- ✓ Taxi driving is clearly stochastic in this sense, because one can never predict the behavior of traffic exactly; moreover, one's tires blow out and one's engine seizes up without warning.
- ✓ The vacuum world as we described it is deterministic, but variations can include stochastic elements
- ✓ One final note: our use of the word "stochastic" generally implies that uncertainty about outcomes is quantified in terms of probabilities; a nondeterministic environment is one in which actions are characterized by their possible outcomes, but no probabilities are attached to them. Nondeterministic environment descriptions are usually associated with performance measures that require the agent to succeed for all possible outcomes of its actions.

- **Episodic vs. sequential:**
  - ✓ In an episodic task environment, the agent's experience is divided into atomic episodes.
  - ✓ In each episode the agent receives a percept and then performs a single action.
  - ✓ Crucially, the next episode does not depend on the actions taken in previous episodes.

- ✓ Many classification tasks are episodic.

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

- **Static vs. dynamic:**
  - ✓ If the environment can change while an agent is deliberating, then we say the environment is dynamic for that agent; otherwise, it is static.
  - ✓ Static environments are easy to deal with because the agent need not keep looking at the world while it is deciding on an action, nor need it worry about the passage of time.
  - ✓ Dynamic environments, on the other hand, are continuously asking the agent what it wants to do; if it hasn't decided yet, that counts as deciding to do nothing.
  - ✓ If the environment itself does not change with the passage of time but the agent's performance score does, then we say the environment is semidynamic.

  Taxi driving is clearly dynamic: the other cars and the taxi itself keep moving while the driving algorithm dithers about what to do next. Chess, when played with a clock, is semi dynamic. Crossword puzzles are static.

- **Discrete vs. continuous:**
  - ✓ The discrete/continuous distinction applies to the state of the environment, to the way time is handled, and to the percepts and actions of the agent.
  - ✓ For example, the chess environment has a finite number of distinct states.
  - ✓ Chess also has a discrete set of percepts and actions.

  Taxi driving is a continuous-state and continuous-time problem:
  - ✓ the speed and location of the taxi and of the other vehicles sweep through a range of continuous values and do so smoothly over time.
  - ✓ Taxi-driving actions are also continuous (steering angles, etc.). Input from digital cameras is discrete, strictly speaking, but is typically treated as representing continuously varying intensities and locations.

- **Known vs. unknown:**
  - ✓ In a known environment, the outcomes (or outcome probabilities if the environment is stochastic) for all actions are given. Obviously, if the environment is unknown, the agent will have to learn how it works in order to make good decisions. Note that the distinction between known and unknown environments is not the same as the one between fully and partially observable environments.
  - ✓ It is quite possible for a known environment to be partially observable—for example, in solitaire card games, I know the rules but am still unable to see the cards that have not yet been turned over. Conversely, an unknown environment can

be fully observable—in a new video game, the screen may show the entire game state but I still don't know what the buttons do until I try them.

Figure 2.6 lists the properties of a number of familiar environments.

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

Figure 2.6 Examples of task environments and their characteristics.

THE STRUCTURE OF AGENTS

- The job of AI is to design an agent program that implements the agent function— the mapping from percepts to actions.
- We assume this program will run on some sort of computing device with physical sensors and actuators—we call this the architecture:

$$agent = architecture + program\ .$$

Agent programs

- The difference between the agent program, which takes the current percept as input, and the agent function, which takes the entire percept history.
- The agent program takes just the current percept as input because nothing more is available from the environment; if the agent's actions need to depend on the entire percept sequence, the agent will have to remember the percepts.
- For example, Figure 2.7 shows a rather trivial agent program that keeps track of the percept sequence and then uses it to index into a table of actions to decide what to do.
- To build a rational agent in this way, we as designers must construct a table that contains the appropriate action for every possible percept sequence.

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

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Figure 2.7 The TABLE-DRIVEN-AGENT program is invoked for each new percept and returns an action each time. It retains the complete percept sequence in memory.

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

Despite all this, TABLE-DRIVEN-AGENT does do what we want: it implements the desired agent function.

- Four basic kinds of agent programs that embody the principles underlying almost all intelligent systems:
  - ✓ Simple reflex agents;
  - ✓ Model-based reflex agents;
  - ✓ Goal-based agents; and
  - ✓ Utility-based agents.

```
function REFLEX-VACUUM-AGENT([location,status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```

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

1. Simple reflex agents

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

✓ Simple reflex behaviors occur even in more complex environments.

✓ Imagine yourself as the driver of the automated taxi. If the car in front brakes and its brake lights come on, then you should notice this and initiate braking.

✓ In other words, some processing is done on the visual input to establish the condition we call "The car in front is braking."

✓ Then, this triggers some established connection in the agent program to the action "initiate braking." We call such a connection a condition–action rule, written as

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

✓ The program in Figure 2.8 is specific to one particular vacuum environment.

✓ A more general and flexible approach is first to build a general-purpose interpreter for condition action rules and then to create rule sets for specific task environments.

✓ Figure 2.9 gives the structure of this general program in schematic form, showing how the condition–action rules allow the agent to make the connection from percept to action.

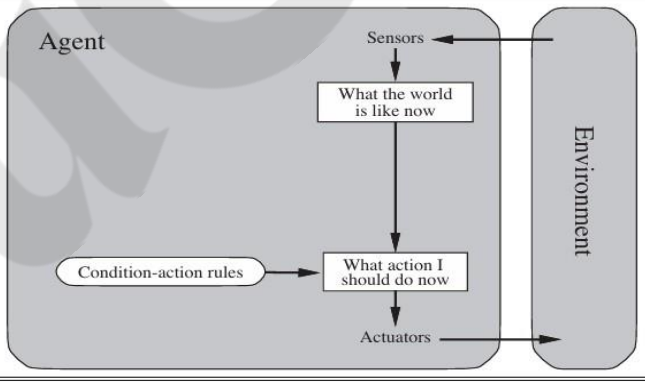

Figure 2.9 Schematic diagram of a simple reflex agent.

```
function SIMPLE-REFLEX-AGENT(percept) returns an action
  persistent: rules, a set of condition–action rules

  state ← INTERPRET-INPUT(percept)
  rule ← RULE-MATCH(state, rules)
  action ← rule.ACTION
  return action
```

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

- ✓ The agent program, which is also very simple, is shown in Figure 2.10.
- ✓ The INTERPRET-INPUT function generates an abstracted description of the current state from the percept, and the RULE-MATCH function returns the first rule in the set of rules that matches the given state description.
- ✓ Simple reflex agents have the admirable property of being simple, but they turn out to be of limited intelligence.
- ✓ The agent in Figure 2.10 will work only if the correct decision can be made on the basis of only the current percept—that is, only if the environment is fully observable.
- ✓ Even a little bit of unobservability can cause serious trouble.
- ✓ Suppose that a simple reflex vacuum agent is deprived of its location sensor and has only a dirt sensor. Such an agent has just two possible percepts: [Dirty] and [Clean].
- ✓ It can Suck in response to [Dirty]; what should it do in response to [Clean]? MovingLeft fails (forever) if it happens to start in square A, and moving Right fails (forever) if it happens to start in square B.
- ✓ Infinite loops are often unavoidable for simple reflex agents operating in partially observable environments.
- ✓ Escape from infinite loops is possible if the agent can randomize its actions. For example, if the vacuum agent perceives [Clean], it might flip a coin to choose between Left and Right. It is easy to show that the agent will reach the other square in an average of two steps. Then, if that square is dirty, the agent will clean it and the task will be complete. Hence, a randomized simple reflex agent might outperform a deterministic simple reflex agent.

2. Model-based reflex agents

- The most effective way to handle partial observability is for the agent to keep track of the part of the world it can't see now. That is, the agent should maintain some sort of internal state that depends on the percept history and thereby reflects at least some of the unobserved aspects of the current state.
- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program.
  - ✓ First, we need some information about how the world evolves independently of the agent—for example, that an overtaking car generally will be closer behind than it was a moment ago.
  - ✓ Second, we need some information about how the agent's own actions affect the world—for example, that when the agent turns the steering wheel clockwise, the car turns to the right, or that after driving for five minutes northbound on the freeway, one is usually about five miles north of where one was five minutes ago.
  - ✓ **This knowledge about "how the world works"—whether implemented in simple Boolean circuits or in complete scientific theories—is called a model of the world. An agent that uses such a model is called a model-based agent.**

✓ Figure 2.11 gives the structure of the model-based reflex agent with internal state, showing how the current percept is combined with the old internal state to generate the updated description of the current state, based on the agent's model of how the

world works.

✓ The agent program is shown in Figure 2.12. The interesting part is the function UPDATE-STATE, which is responsible for creating the new internal state description.

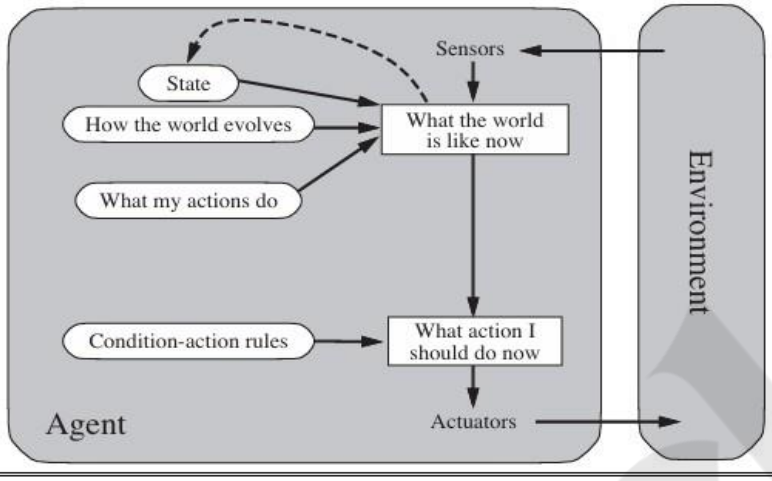

Figure 2.11 A model-based reflex agent.

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action
persistent: *state*, the agent's current conception of the world state
model, a description of how the next state depends on current state and action
rules, a set of condition–action rules
action, the most recent action, initially none

state ← UPDATE-STATE(*state*, *action*, *percept*, *model*)
rule ← RULE-MATCH(*state*, *rules*)
action ← *rule*.ACTION
return *action*

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

3. Goal-based agents

- Knowing something about the current state of the environment is not always enough to decide what to do.
- For example, at a road junction, the taxi can turn left, turn right, or go straight on. The correct decision depends on where the taxi is trying to get to.
- In other words, as well as a current state description, the agent needs some sort of goal information that describes situations that are desirable—for example, being at the passenger's destination.
- The agent program can combine this with the model (the same information as was used in the model based reflex agent) to choose actions that achieve the goal. Figure 2.13 shows the goal-based agent's structure.
- Sometimes goal-based action selection is straightforward—for example, when goal sat isfaction results immediately from a single action. Sometimes it will be more tricky—for example, when the agent has to consider long sequences of twists and turns in order to find a way to achieve the goal.

- **Search and planning** are the subfields of AI devoted to finding action sequences that achieve the agent's goals.

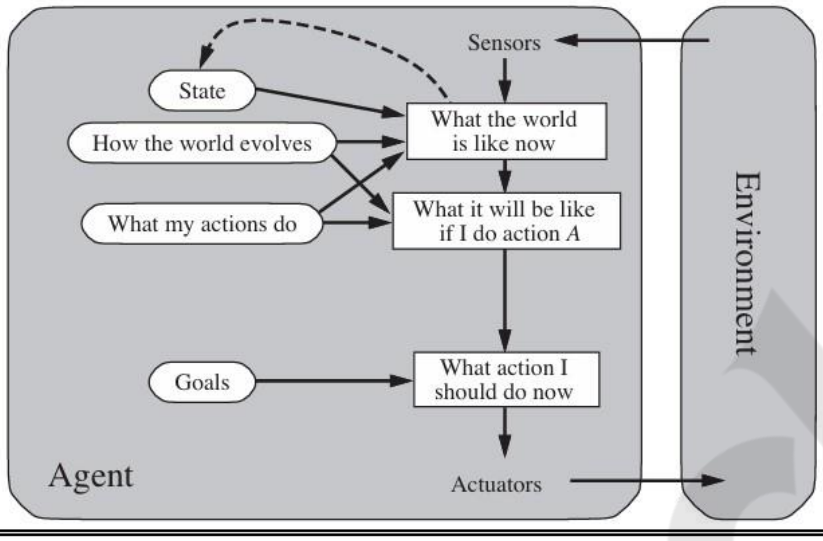

Figure 2.13 A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

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

4. Utility-based agents

- Goals alone are not enough to generate high-quality behavior in most environments.
- For example, many action sequences will get the taxi to its destination but some are quicker, safer, more reliable, or cheaper than others.
- Goals just provide a crude binary distinction between "happy" and "unhappy" states.
- A more general performance measure should allow a comparison of different world states according to exactly how happy they would make the agent.
- Because "happy" does not sound very scientific, economists and computer scientists use the term utility instead.
- **An agent's utility function** is essentially an internalization of the performance measure. If the internal utility function and the external performance measure are in agreement, then an

agent that chooses actions to maximize its utility will be rational according to the external performance measure.

- **A rational utility-based agent** chooses the action that maximizes the expected utility of the action outcomes—that is, the utility the agent expects to derive, on average, given the probabilities and utilities of each outcome.
- The utility-based agent structure appears in Figure 2.14.
- Utility-based agent programs appear in Part IV, where we design decision-making agents that must handle the uncertainty inherent in stochastic or partially observable environments.

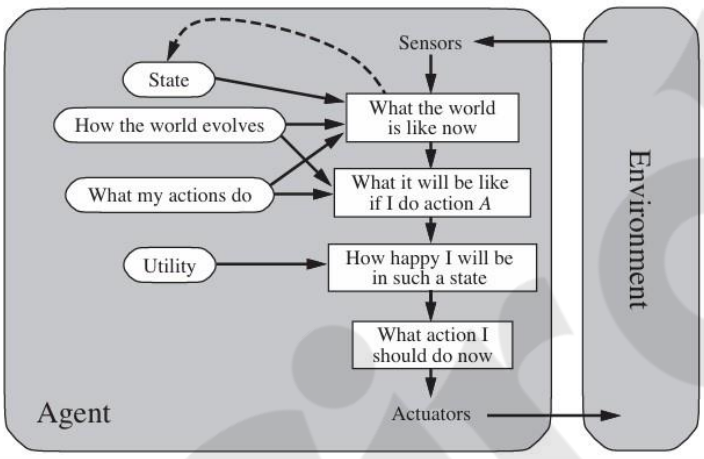

Figure 2.14 A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible outcome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

Learning agents

- A learning agent can be divided into four conceptual components, as shown in Figure 2.15.
- The most important distinction is between the learning element, which is re sponsible for making improvements, and the performance element, which is responsible for selecting external actions.
- The performance element is what we have previously considered to be the entire agent: it takes in percepts and decides on actions.
- The learning element uses feedback from the critic on how the agent is doing and determines how the performance element should be modified to do better in the future.
- The design of the learning element depends very much on the design of the performance element.
- When trying to design an agent that learns a certain capability, the first question is not “How am I going to get it to learn this?” but “What kind of performance element will my agent need to do this once it has learned how?” Given an agent design, learning mechanisms can be constructed to improve every part of the agent.

- The critic tells the learning element how well the agent is doing with respect to a fixed performance standard. The critic is necessary because the percepts themselves provide no indication of the agent's success.
- The last component of the **learning agent is the problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences.
- The point is that if the performance element had its way, it would keep doing the actions that are best, given what it knows.

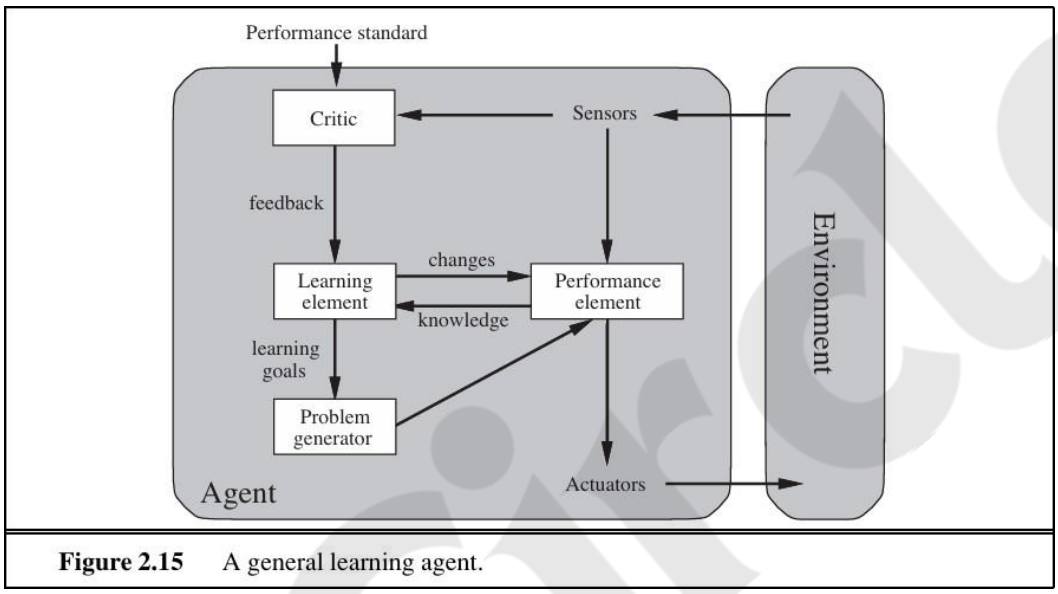

Figure 2.15 A general learning agent.

How the components of agent programs work?

- The various ways that the components can represent the environment that the agent inhabits.
- We can place the representations along an axis of increasing complexity and expressive power—**atomic, factored, and structured.**
- To illustrate these ideas, it helps to consider a particular agent component, such as the one that deals with "What my actions do." This component describes the changes that might occur in the environment as the result of taking an action, and Figure 2.16 provides schematic depictions of how those transitions might be represented.

1. Atomic representation:

- In an atomic representation each state of the world is indivisible—it has no internal structure.
- Consider the problem of finding a driving route from one end of a country to the other via some sequence of cities.
- For the purposes of solving this problem, it may suffice to reduce the state of world to just the name of the city we are in—a single atom of knowledge; a "black box" whose only discernible property is that of being identical to or different from another black box.

- The algorithms underlying search and game-playing , Hidden Markov models, and Markov decision processes all work with atomic representations—or, at least, they treat representations as if they were atomic.

2. Factored representation

- A factored representation splits up each state into a fixed set of variables or attributes, each of which can have a value.
- While two different atomic states have nothing in common—they are just different black boxes—two different factored states can share some attributes (such as being at some particular GPS location) and not others (such as having lots of gas or having no gas); this makes it much easier to work out how to turn one state into another.
- With factored representations, we can also represent uncertainty—for example, ignorance about the amount of gas in the tank can be represented by leaving that attribute blank.
- Many important areas of AI are based on factored representations, including constraint satisfaction algorithms, propositional logic, planning, Bayesian networks, and the machine learning algorithms.

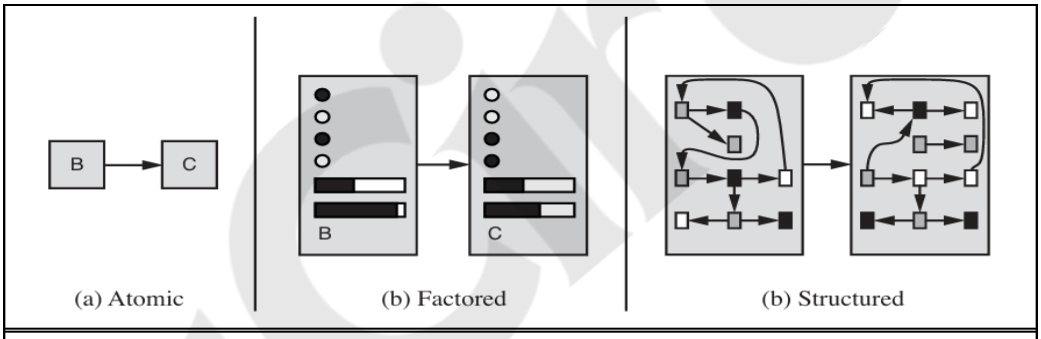

Figure 2.16 Three ways to represent states and the transitions between them. (a) Atomic representation: a state (such as B or C) is a black box with no internal structure; (b) Factored representation: a state consists of a vector of attribute values; values can be Boolean, real-valued, or one of a fixed set of symbols. (c) Structured representation: a state includes objects, each of which may have attributes of its own as well as relationships to other objects.

- A factored representation is unlikely to be pre-equipped with the attribute *TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow* with value true or false.

3. Structured representation

- A structured representation, in which objects such as cows and trucks and their various and varying relationships can be described explicitly. (See Figure 2.16(c).)
- Structured representations underlie relational databases and first-order logic, first-order probability models, knowledge-based learning and much of natural language understanding.
- In fact, almost everything that humans express in natural language concerns objects and their relationships.

****