# Intelligent Agents

An **agent** is anything that can be viewed as **perceiving its environment** through **sensors** and acting upon that environment through **actuators**.

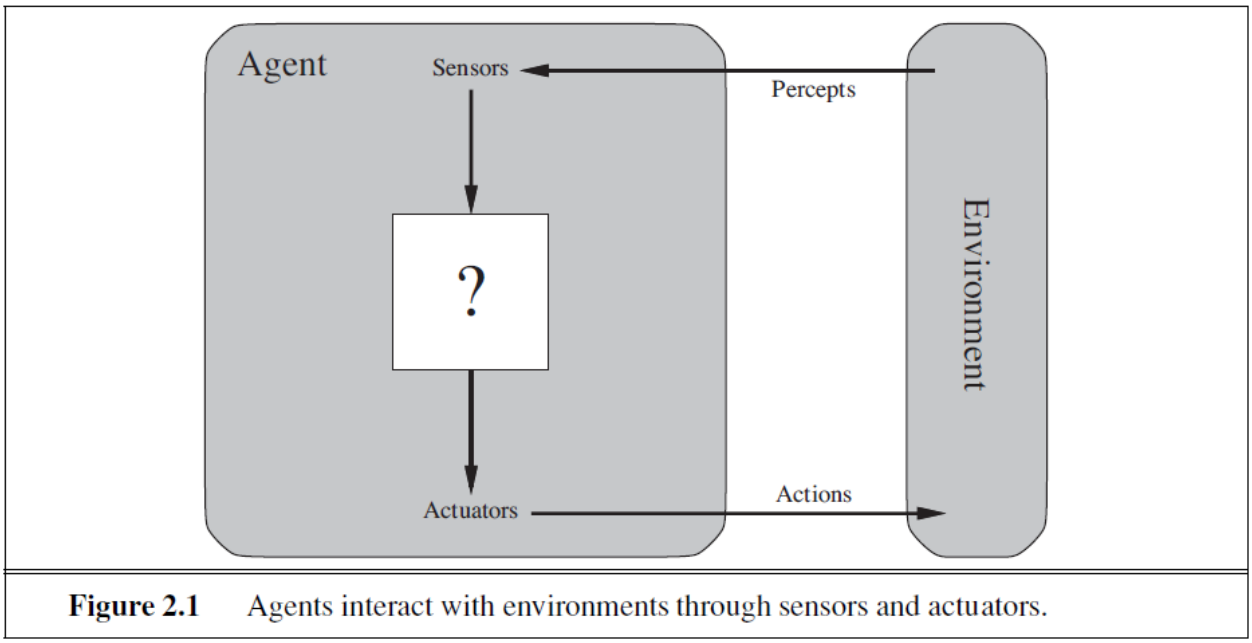



**Figure 2.1** Agents interact with environments through sensors and actuators.

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 agents' 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 anything it hasn't perceived. Mathematically speaking, we say that an agent's behavior is described by the **agent function** that maps any given percept sequence to an action. **Internally**, the agent function for an artificial agent will be implemented by an **agent program**. It is important

to keep these two ideas distinct. The **agent function** is an abstract mathematical description; the agent program is a concrete implementation, running within some physical system.

## Good Behavior: The concept of Rationality

A **rational agent** is one that does the right thing. Obviously, doing the right thing is better than doing the wrong thing, but what does it mean to do the right thing?

Let considering the consequences of the agent's behavior. When an agent working in its environment, it generates a sequence of actions according 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. Obviously, there is not one fixed performance measure for all tasks and agents, typically, a designer will devise one appropriate to the circumstances.

### 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 expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge that agent has.*

### Omniscience, learning, and autonomy?

We need to be careful to distinguish between rationality and **omniscience**. Our definition of rationality does not require omniscience, then, because the rational choice depends only on the percept sequence to date. We must also ensure that we haven't inadvertently allowed the agent to engage in decidedly under intelligent activities. For example, if an agent does not look both ways before crossing a busy road, then its percept sequence will not tell it that there is a large truck approaching at high speed. Does our definition of rationality say that it's now OK to cross the road? Far from it! First, it would not be rational to cross the road given this uninformative percept sequence: the risk of accident from crossing with out looking is too great. Second, a rational agent should choose the "looking" action before stepping into the street, because looking helps maximize the expected performance. Doing actions in order to modify future percepts, sometimes called **information gathering** which is an important part of rationality. A second example of information gathering is provided by the **exploration** that must be undertaken by an agent in an initially unknown environment.

Our definition requires a rational agent not only to gather information but also to **learn** as much as possible from what it perceives. The agent's initial configuration could reflect some prior knowledge of the environment, but as the agent gains experience this may be modified and augmented. There are extreme cases in which the environment is completely known a priori.

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

## The Nature of Environments?

**Task environments**, which is essential the "**problem**" to which rational agents are the **solutions**. We begin by showing how to specify a task environment, illustrating the process with a number of examples. We then show that task environments come in a variety of flavors. The flavor of the task environment directly affects the appropriate design for the agent program.

### 1. Specifying the task environment.

Task environment, acronymically minded, we call this the **PEAS** (**P**erformance, **E**nvironment, **A**ctuators, **S**ensors). In designing an agent, the first step must always be to specify the task environment as fully as possible. For example, the following figure describe the PEAS for the automated taxi 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 |

**Performance measure** to which we would like our automated taxi 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. Obviously, some of these goals conflict, so tradeoffs will be required.

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 such as weather conditions.

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.

The basic **sensor** for the taxi will include one or more controllable video cameras so that is 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 will need the usual array of engine, fuel and electrical system sensors. Like may 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 destination.

**2. Properties of task environments.**

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

1. **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 for the sensor data. For example, an automated taxi cannot see what other drivers are thinking.

2. **Single agent vs. multiagent:** the distinction between single-agent and multiagent environments may seem simple enough. For an 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. The key distinction is whether B's behavior is best described as maximizing a performance measure whose value depends on agent A's behavior. For example, in chess, the opponent entity B is trying to maximize its performance measure, which, by the rules of chess, minimizes agent A's performance measure. Thus, chess I a **competitive multiagent environment**. In the taxi-driving environment, on the other hand, avoiding collisions maximizes the performance measures of all agents, so it is a **partially cooperative multiagent environment**. It is also competitive because, for example, only one car can occupy a parking space.
3. **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 fully observable, deterministic environment. If the environment is partially observable, however, then it could appear to be stochastic. Most real situation are 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.
4. **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. For example, an agent that has to spot defective parts on an assembly line bases each decision on the current part, regardless of previous decisions, moreover, the current decision doesn't affect whether the next part is defective. In sequential environments, on the other hand, the current decision could affect all future decisions. Chess and taxi driving are sequential. Episodic environments are much simple than sequential environments because the agent does not need to think ahead.

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

| 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?

So far, we have talked about agents by describing behavior. The action that is performed after any given sequence of percepts. Now we must bite the bullet and talk about how the insides work. 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 agent program discussed in this lecture have the same skeleton, they take the current percept as input from the sensors and return an action to the actuators. We have to notice the difference between the agent program which takes the current percept as input and the agent function, which takes the entire percept history. The agent program takes just the current 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. The following example,

shows an 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.

```
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.

To build an agent program in this way, we as designers must construct a table that contains the appropriate action for every possible percept sequence. It is instructive to consider why the table-driven approach to agent construction is doomed to failure. Let 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 $\sum_{t=1}^{T} |P|^t$ 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, 640X480 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 would have at lest $10^{150}$ entries. The daunting size of these tables means that:

1. No physical agent in this universe will have the space to store the table.
2. The designer would not have time to create the table.
3. No agent could ever learn all the right table entries from its experience.
4. No guidance about how to fill in the table entries.

The key challenge for AI is to find out how to write program that, to the extent possible, produce rational behavior from a smallish program rather than from a vast table. We have many examples showing that this can be done successfully in other areas. For example,

the huge tables of square roots used by engineers and schoolchildren prior to the 1970s have not been replaced by a five-line program for newton's method running on electronic calculators. In the reminder of this lecture, we outline four basic kinds of agent programs that embody the principles underlying almost all intelligent systems.

- Simple reflex agents
- Model-base reflex agents
- Goal- based agents
- Utility-based agents.

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. A simple reflex behavior occurs in many environments. Imagine the automated taxi, if the car in front brakes and its brake lights come on, then the automated taxi program 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-breaking

   Figure below, 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.

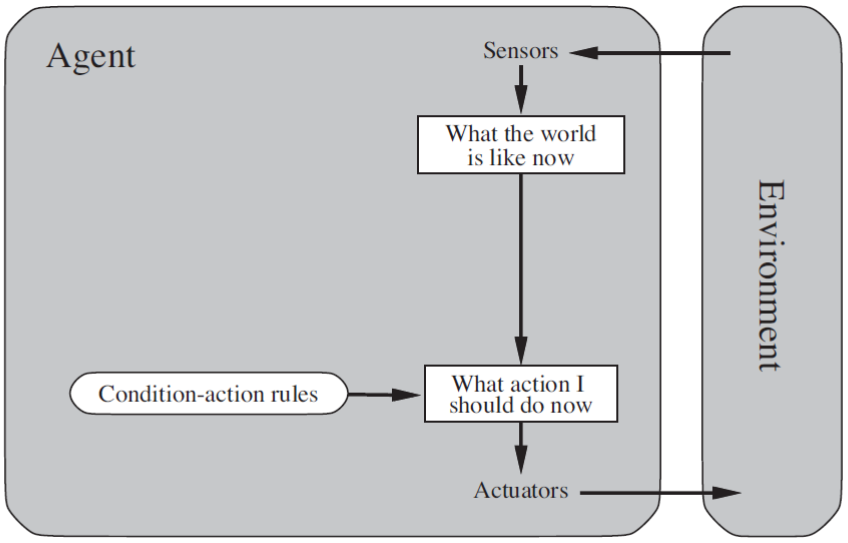



**Figure 2.9** Schematic diagram of a simple reflex agent.

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. For example, the braking problem the internal state is not too extensive just the previous frame from the camera, allowing the agent to detect when two red lights at the edge of the vehicle go on or off simultaneously. 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 e volves independently of the agent. Second, we need some information about how the agent's own actions affect the world. This knowledge about "how the world works" is called a model of the world. An agent that uses such a model is called a model-based agent.

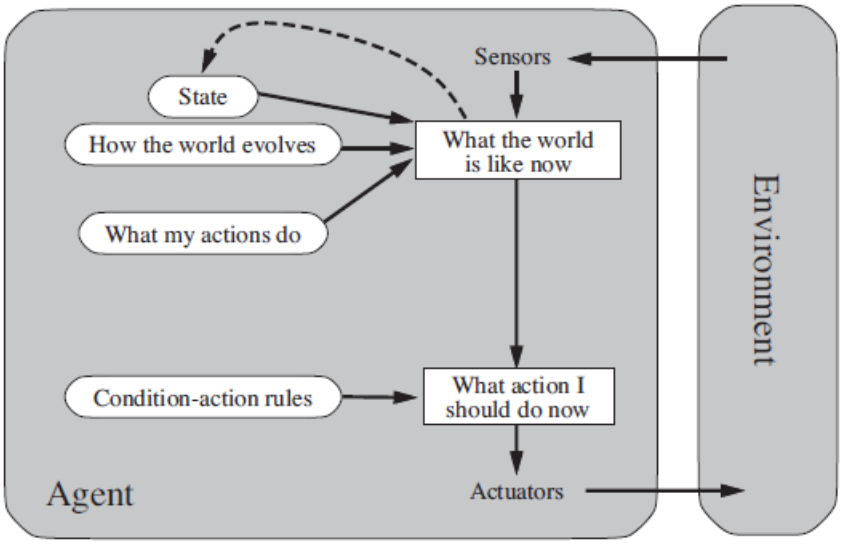
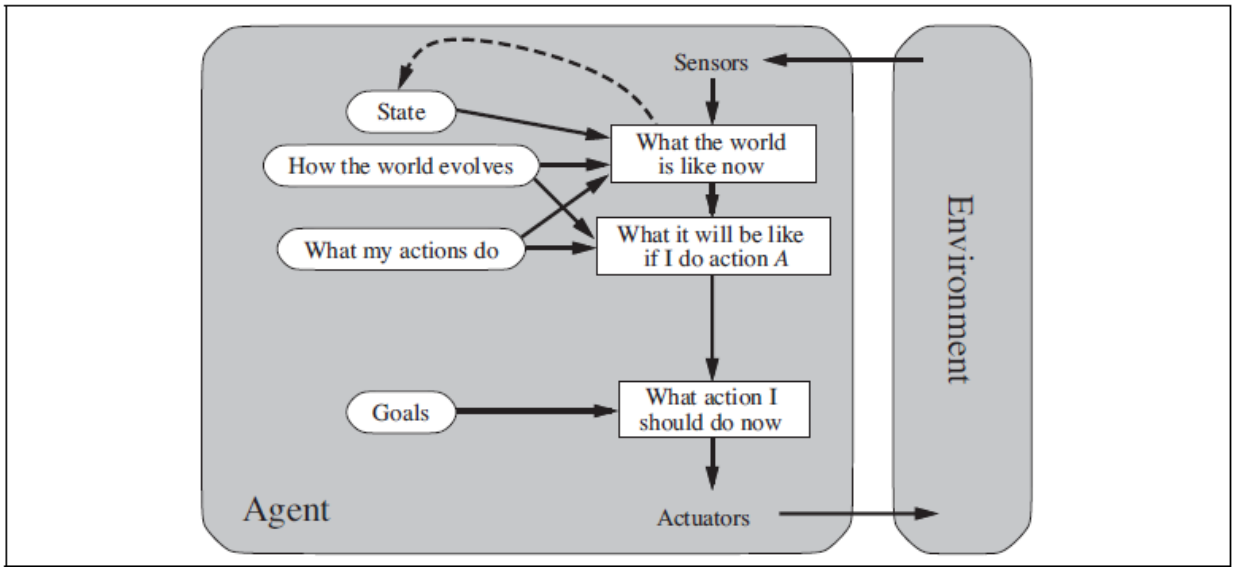
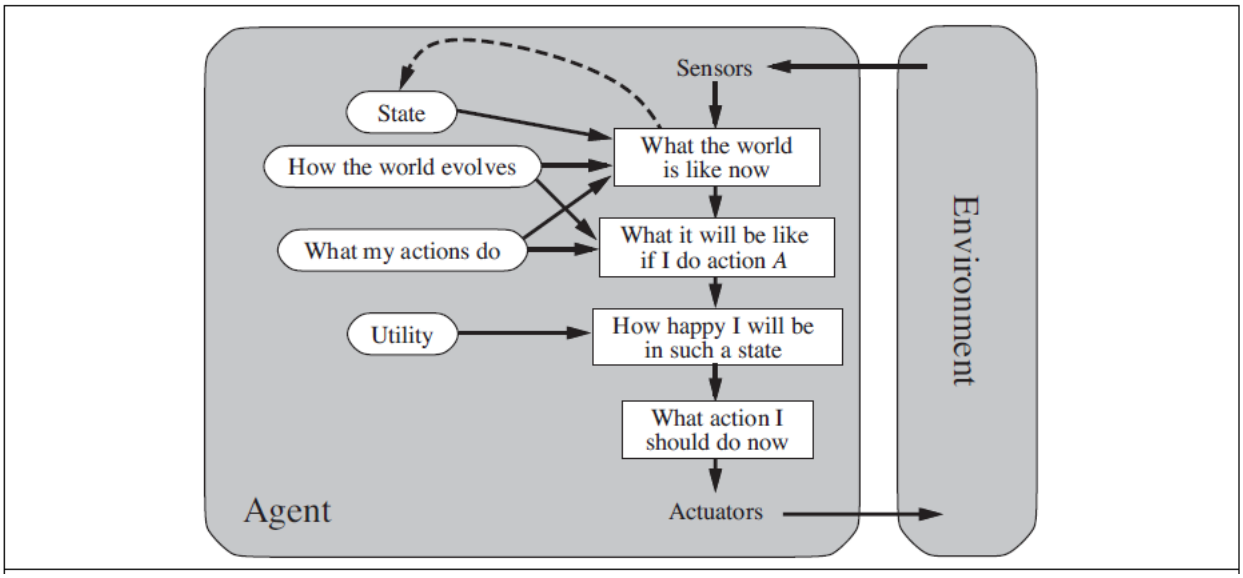



**Figure 2.11** A model-based reflex agent.

3. **Goal-based agents:** knowing something about the current state of the environment is not always enough to decided 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, the passenger destination. The agent program can combine this with the model to choose action that achieve the goal. Figure below shows the goal-based agents structure.

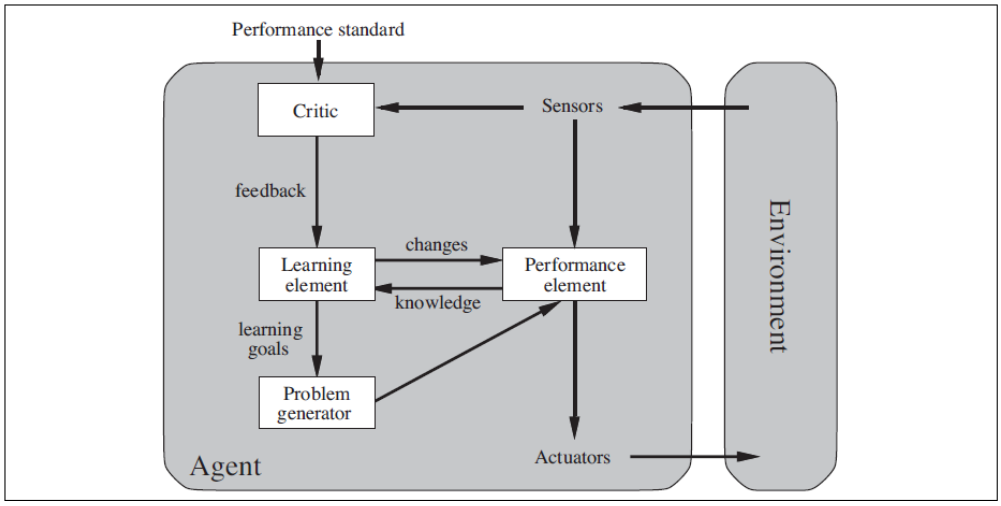



4. **Utility-based agents:** goals alone are not enough to generate high-quality behavior in environments. For example, many action sequences will get the taxi to its destination (thereby achieving the goal) but some are quicker, safer, more reliable, or cheaper than others. We have discussed that a performance measure assigns a score to any given sequence of environment states. So, it can easily distinguish between more and less desirable ways of getting to the taxi's destination. An agent's **utility function** is essentially an internalization of the performance measure.



## Learning agents?

A learning agent can be divided into four components, as shown in figure below.




1. Learning element, which is responsible for making improvements.
2. Performance element, which is responsible for selecting external actions. It takes in percepts and decides on actions (we consider to be the entire agent).
3. Critic: the learning element used 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.
4. Problem generator: it is responsible for suggesting actions that will lead to new informative experiences. The problem generator's job is to suggest these exploratory actions which may lead to discover much better actions for the long run.

To make the overall design more concrete, let us return to the automated taxi example. The performance element consists of whatever collection of knowledge and procedures the taxi has for select its driving actions. The taxi goes out on the road and drives, using this performance element. The critic observes the world and passes information along to the learning element. For example, after the taxi makes quick left turn across three lanes of traffic, critic observes the shocking language used by other

drivers. From this experience, the learning element is able to formulate a rule saying this was a bad action, and the performance element is modified by installation of the new rule. The problem generator might identify certain areas of behavior in need of improvement and suggest experiments, such as trying out the brakes on different road surfaces under different conditions.

## Summary?

- An agent is something that perceives and acts in an environment. The agent function for an agent specifies the action taken by the agent in response to any percept sequence.
- The performance measure evaluates the behavior of the agent in an environment. A rational agent acts so as to maximize the expected value of the performance measure, given the percept sequence it has been so far.
- A task environment specification includes the performance measure, the external environment, the actuators and the sensors. In designing an agent, the first step must always be to specify the task environment as fully as possible.
- Task environments vary along several significant dimensions. They can be fully or partially observable, single-agent or multiagent, deterministic or stochastics, episodic or sequential, static or dynamic, discrete or continuous, and known or unknown.
- The agent program implements the agent function. There exists a variety of basic agent-program designs reflecting the kind of information made explicit and used in the decision process. The designs vary in efficiency. Compactness, and flexibility. The

appropriate design of the agent program depends on the nature of the environment.

- Simple reflex agents respond directly to percepts, whereas model-based reflex agents maintain internal state to track aspects of the world that are not evident in the current percept. Goal-based agents act to achieve their goals, and utility-based agents try to maximize their own expected “happiness (utility)”.
- All agents can improve their performance through learning.