# INTELLIGENT AGENTS

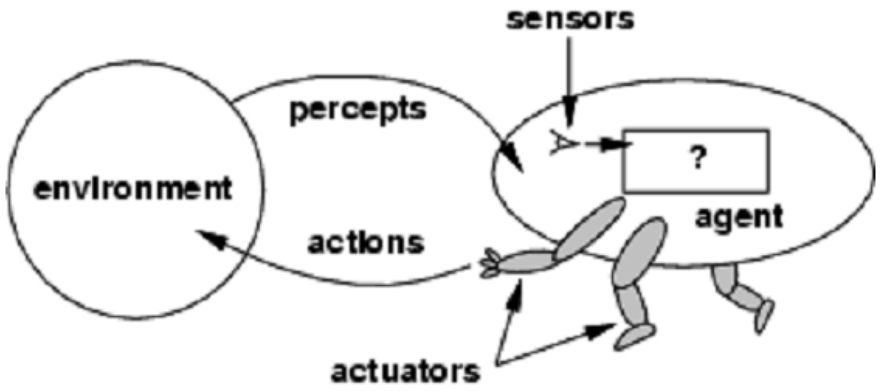
# After this lesson:

- Identify successful agents—systems that can reasonably be called **intelligent**.
- Describe a number of basic “skeleton” agent designs.

# AGENTS AND ENVIRONMENTS

- An **agent** is anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators.**
- A human agent has: eyes, ears, and other organs for sensors and hands, legs, vocal tract
- 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.

# AGENTS AND ENVIRONMENTS




- **Percept: agent's perceptual inputs at an instant**
- **The agent function maps from percept sequences to actions:**
  **[*f*: *P* * → *A*]**
- **The agent program runs on the physical architecture to produce *f***
- **agent = architecture + program**

# Simple Agent

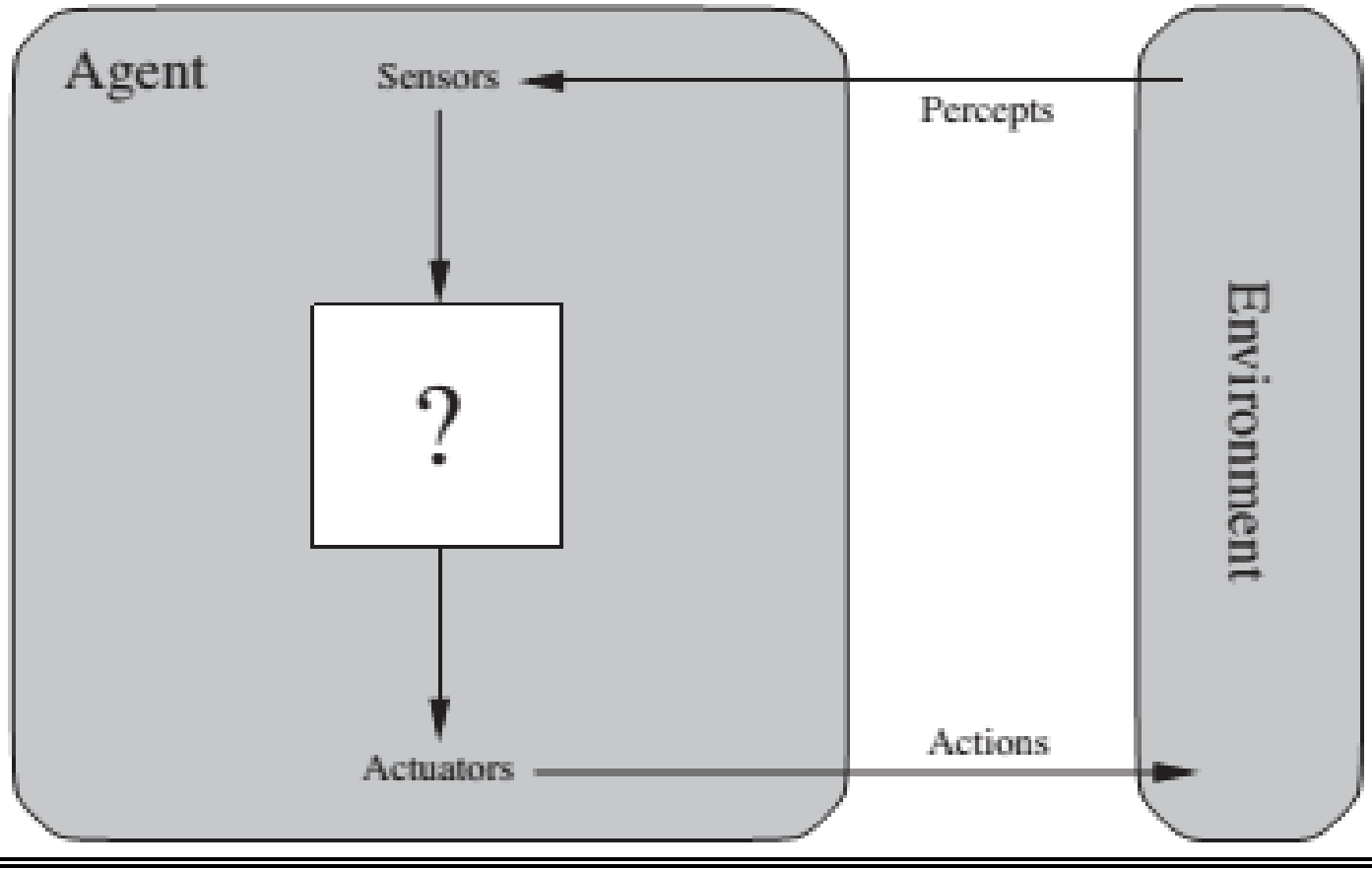



**Figure 2.1** Agents interact with environments through sensors and actuators.

# Percept and percept sequence

- **Percept** to refer to the agent's perceptual inputs at any given instant
- **Percept sequence** is the complete history of everything the agent has ever perceived
- 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

# AGENT FUNCTION & AGENT PROGRAM

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

# Simple Example: the vacuum-cleaner world

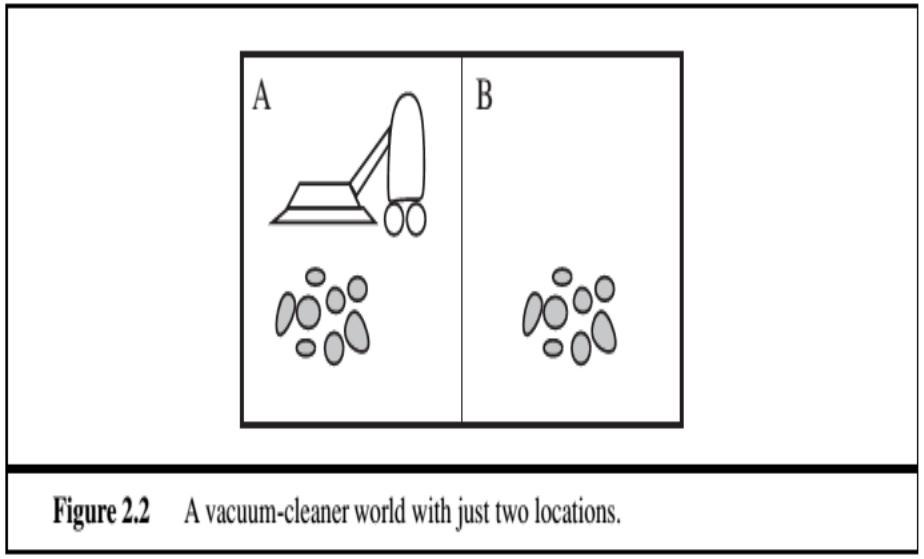



**Figure 2.2** A vacuum-cleaner world with just two locations.

- 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

# Simple Example: the vacuum-cleaner world

| 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.

# RATIONAL AGENT

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

# RATIONAL AGENT: Important Parts

- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)
- An agent is autonomous if its behavior is determined by its own percepts & experience (with ability to learn and adapt) without depending solely on build-in knowledge

# Task Environment

- Before we design an intelligent agent, we must specify its “task environment”:
- **PEAS:**

**Performance measure**
**Environment**
**Actuators**
**Sensors**

# PEAS

- **Example: Agent = taxi driver**

**– Performance measure: Safe, fast, legal, comfortable trip, maximize profits**

**– Environment: Roads, other traffic, pedestrians, customers**

**– Actuators: Steering wheel, accelerator, brake, signal, horn**

**– Sensors: Cameras, sonar, speedometer, GPS, odometer, engine sensors, keyboard**

# PEAS

- **Example: Agent = Medical diagnosis system**

**Performance measure: Healthy patient, minimize costs, lawsuits**

**Environment: Patient, hospital, staff**

**Actuators: Screen display (questions, tests, diagnoses, treatments, referrals)**

**Sensors: Keyboard (entry of symptoms, findings, patient's answers)**

# PEAS

**Example: Agent = Part-picking robot**

- **Performance measure: Percentage of parts in correct bins**
- **Environment: Conveyor belt with parts, bins**
- **Actuators: Jointed arm and hand**
- **Sensors: Camera, joint angle sensors**

# Environment types

- **Fully observable (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.**
- **Deterministic (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent. (If the environment is deterministic except for the actions of other agents, then the environment is strategic)**
- **Episodic (vs. sequential): An agent's action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.**

# Environment types

- **Static** (vs. **dynamic**): **The environment is unchanged while an agent is deliberating. (The environment is semi-dynamic if the environment itself does not change with the passage of time but the agent's performance score does)**

- **Discrete** (vs. **continuous**): **A limited number of distinct, clearly defined percepts and actions**

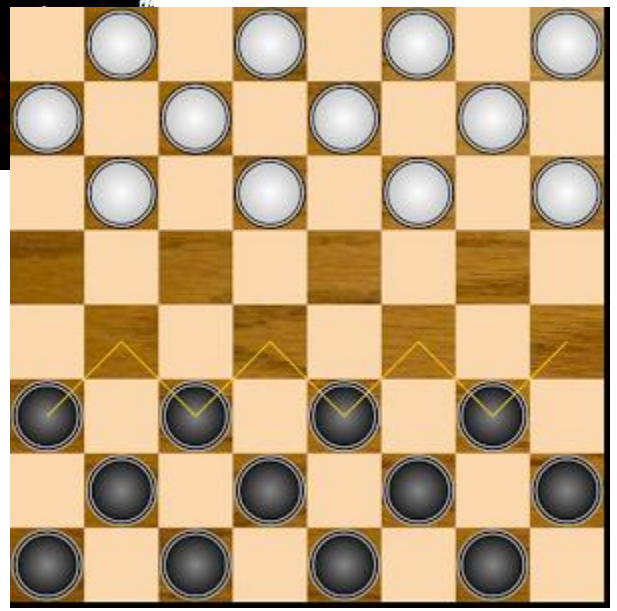
- **Single agent** (vs. **multi-agent**): **An agent operating by itself in an environment.**

# Environment types

- **Known ( vs. unknown): In a known environment, the outcomes for all actions are given. if the environment is unknown, the agent will have to learn how it works in order to make good decision**

# Do you know?

- Poker
- Checker
- Chess
- Backgammon



# Properties of task 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**

# Four basic kinds of agent programs

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

# 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

```
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.

# Simple reflex agents

- condition–action rule (situation–action rules, productions, if–then rules)
  - If car-in-front-is-braking then initiate-braking.

  Ex: blinking when something approaches the eye
- 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.

# Simple reflex agents

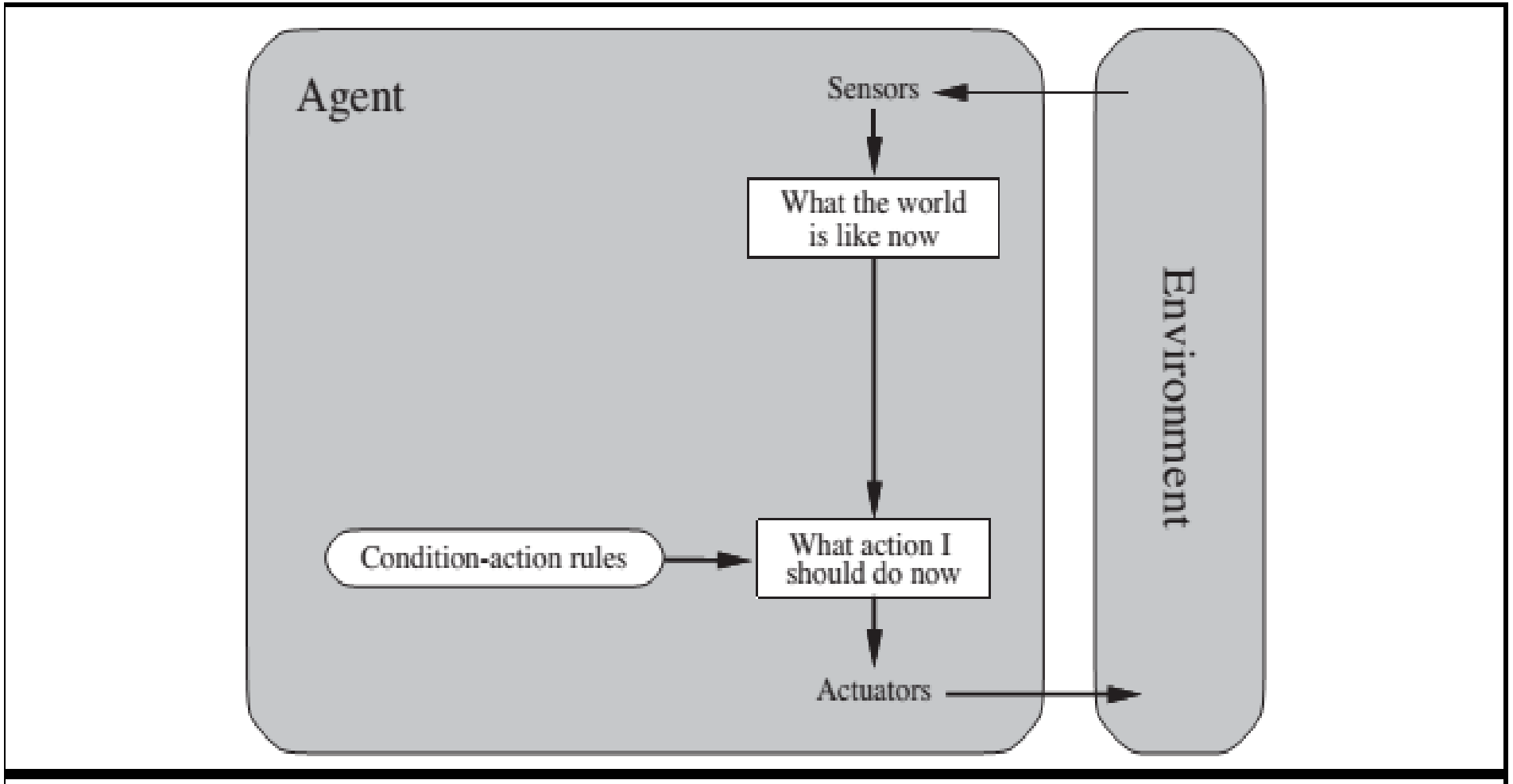



**Figure 2.9** Schematic diagram of a simple reflex agent.

# Simple reflex agents

```
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.

# Simple reflex agents

- The INTERPRET-INPUT function generates an abstracted description of the current state from the percept
- The RULE-MATCH function returns the first rule in the set of rules that matches the given state description

# Four basic kinds of agent programs

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

# Model-based reflex agents

- Keep track of the part of the world it can't see now
- 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
  1. we need some information about how the world evolves independently of the agent
  2. we need some information about how the agent's own actions affect the world

# Model-based reflex agents

- Updating this internal state information as time goes by requires two kinds of knowledge to be encoded in the agent program
- This knowledge about **“how the world works”**
- whether implemented in simple Boolean circuits or in complete scientific theories—is called **a model of the world**
- An agent that uses such a model is called a **model-based agent**.

# Model-based reflex agents

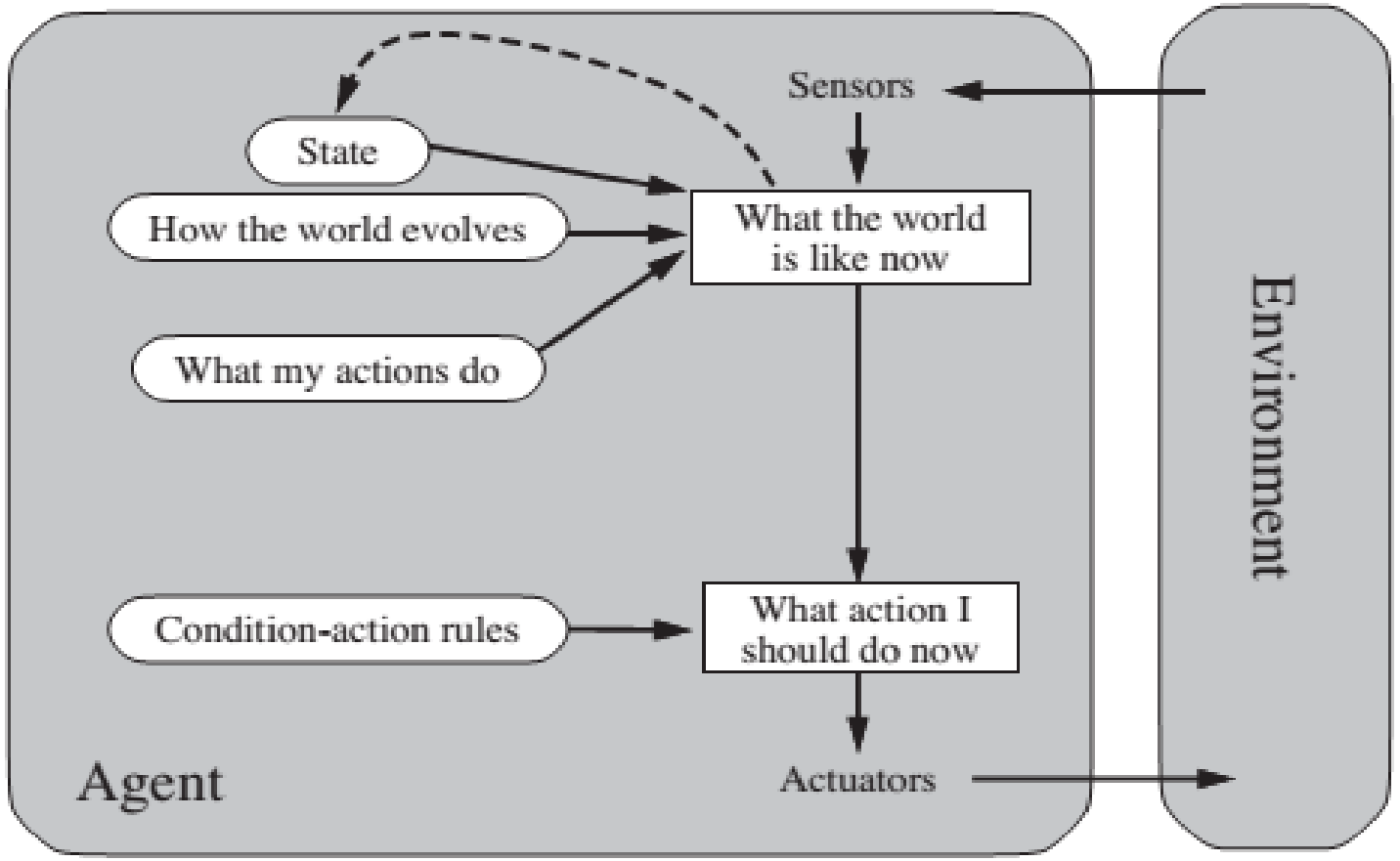



**Figure 2.11** A model-based reflex agent.

# Model-based reflex agents

```
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.

# Four basic kinds of agent programs

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

# 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

# Goal-based agents

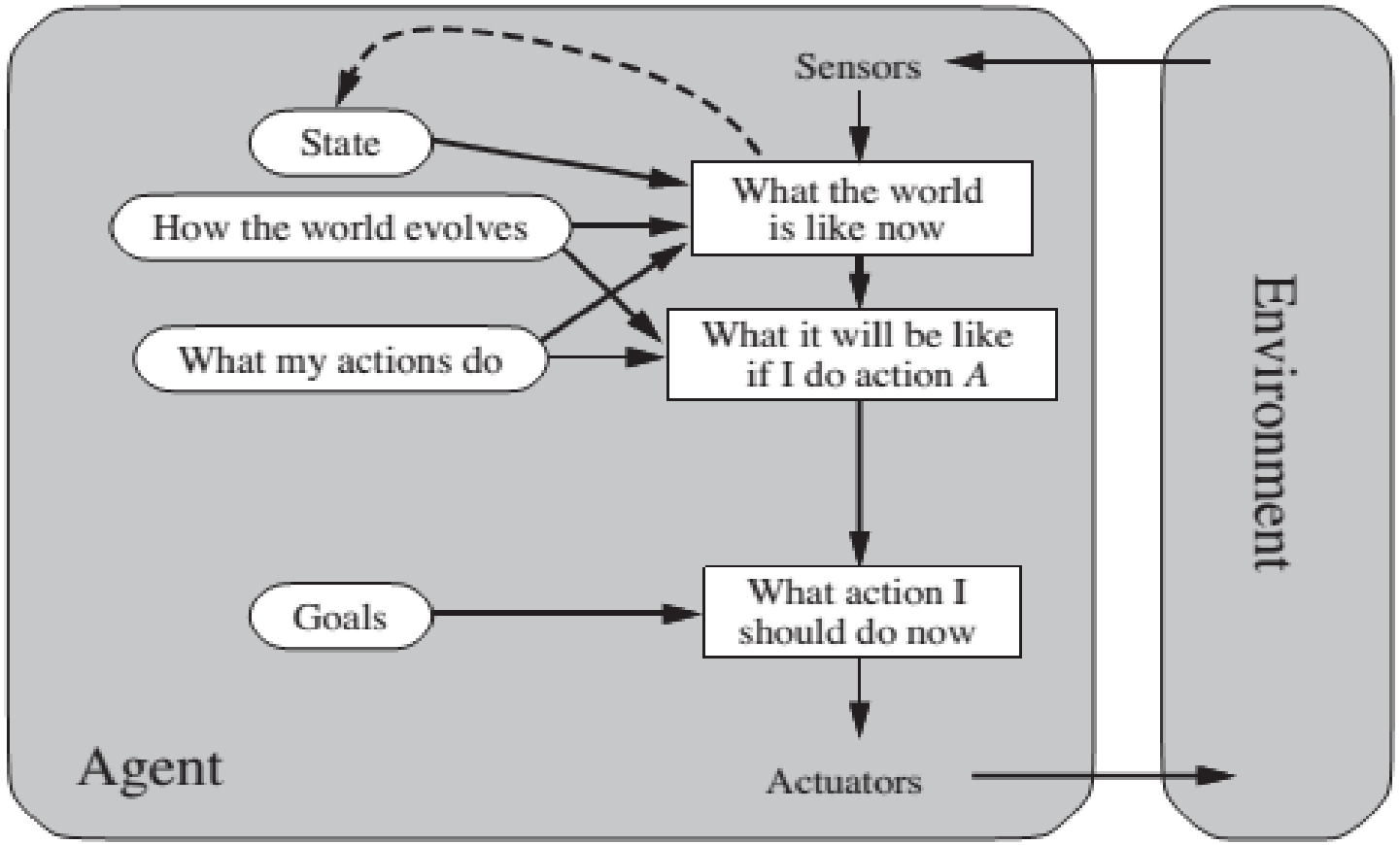



**Figure 2.13** A model-based, goal-based agent. It keeps track of the world state as well as a set of goals it is trying to achieve, and chooses an action that will (eventually) lead to the achievement of its goals.

# Goal-based agents

- Sometimes goal-based action selection is straightforward
  - for example, when goal satisfaction 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.

# Goal-based agents

- 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

# Four basic kinds of agent programs

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

# Utility-based agents

- 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
- Goals just provide a crude binary distinction between “happy” and “unhappy” states.
- 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.

# Utility-based agents

- In two kinds of cases, goals are inadequate but a utility-based agent can still make rational decisions
- First, when there are conflicting goals, only some of which can be achieved (for example, speed and safety), the utility function specifies the appropriate tradeoff
- Second, when there are several goals that the agent can aim for, none of which can be achieved with certainty, utility provides a way in which the likelihood of success can be weighed against the importance of the goals.

# Utility-based agents

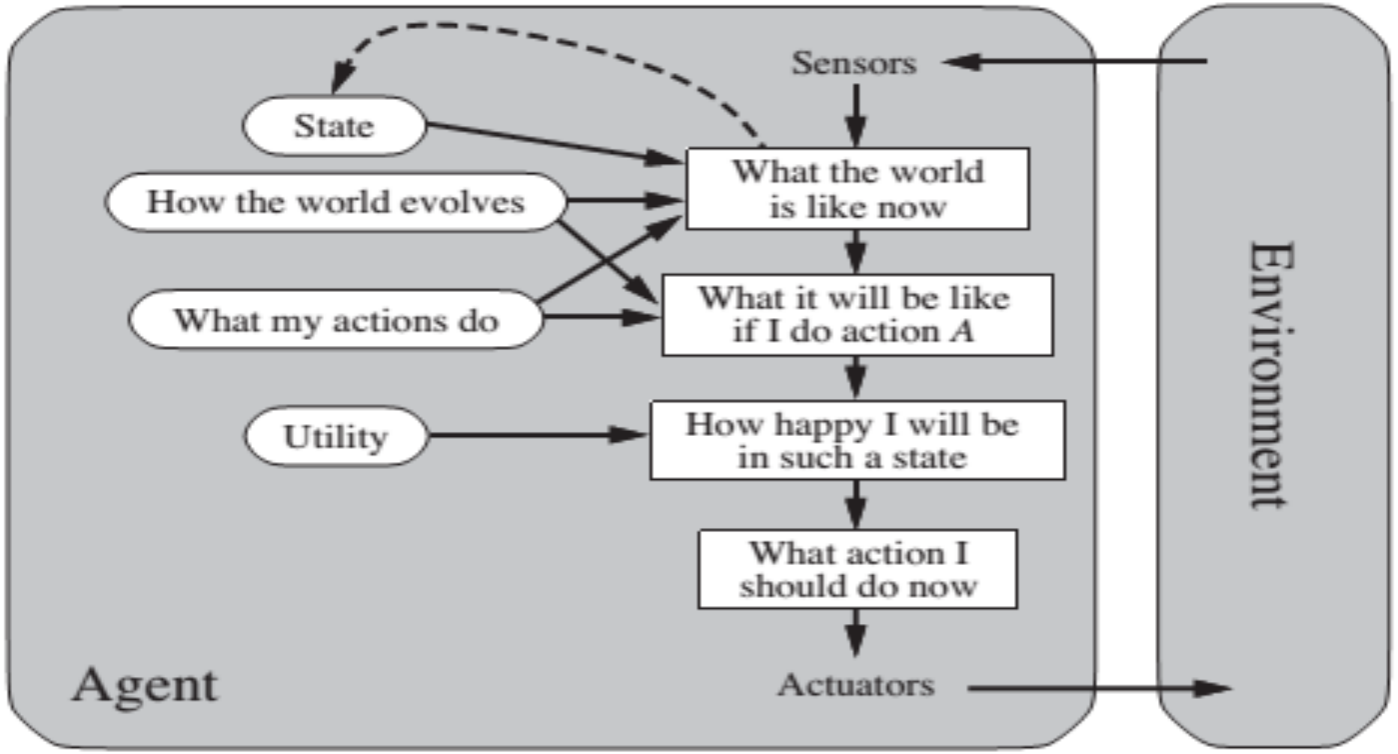



**Figure 2.14** A model-based, utility-based agent. It uses a model of the world, along with a utility function that measures its preferences among states of the world. Then it chooses the action that leads to the best expected utility, where expected utility is computed by averaging over all possible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

# Utility-based agents

- A rational utility-based agent chooses the action that maximizes the expected utility of the action outcomes
- A utility-based agent has to model and keep track of its environment, tasks that have involved a great deal of research on perception, representation, reasoning, and learning.

# Learning agents

- How the agent programs come into being
- Turing (1950)
  - The method he proposes is to build learning machines and then to teach them
- A learning agent can be divided into four conceptual components
  - Learning element
  - Performance element
  - Critic
  - Problem generator

# Learning agents

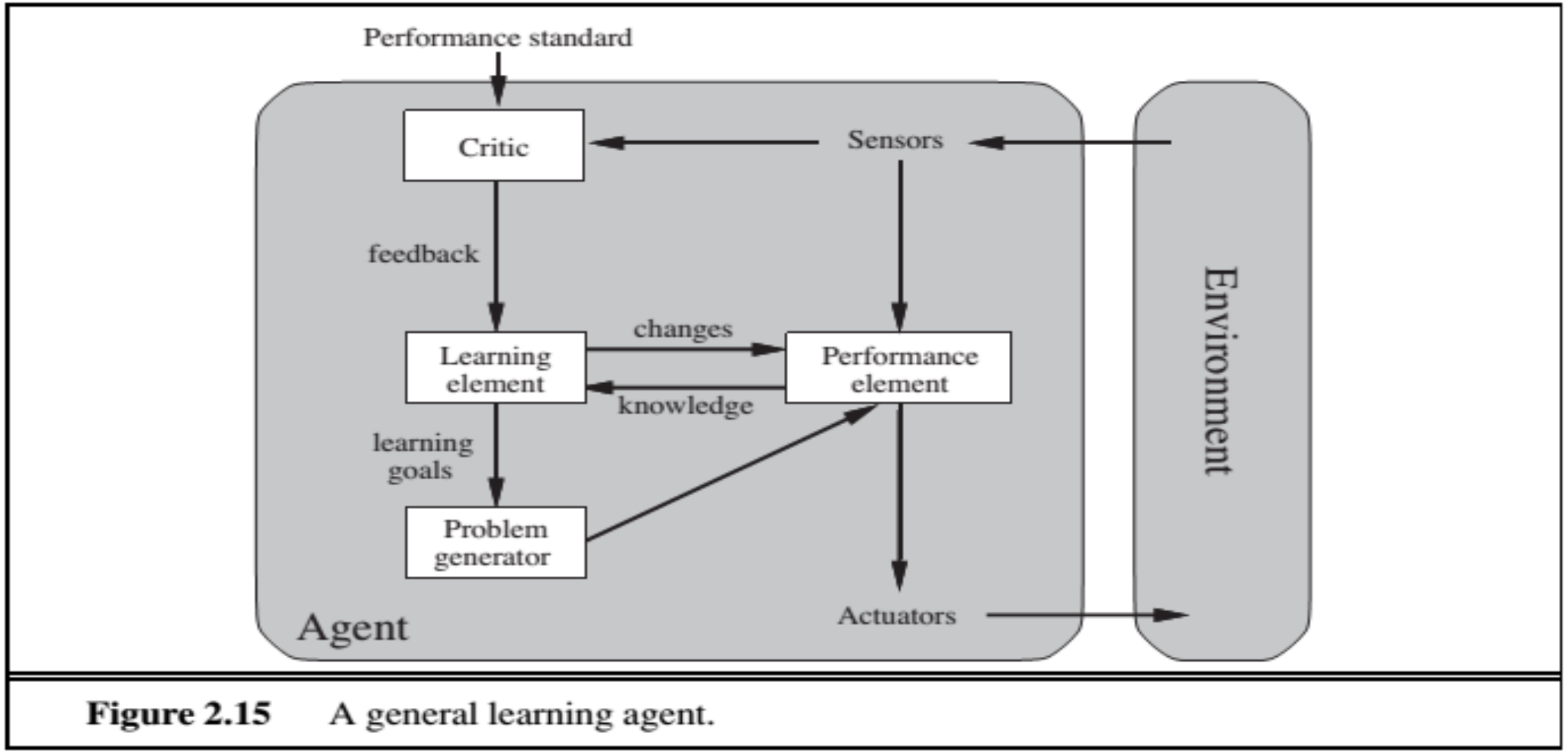



**Figure 2.15** A general learning agent.

# Learning agents

- **Learning element**, which is responsible for making improvements
- **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 last component of the learning agent is the **problem generator**. It is responsible for suggesting actions that will lead to new and informative experiences

# Summary