Chapter 1

Basic Concepts

This chapter introduces the basic concepts of reinforcement learning (RL) that will be used throughout this book. We first introduce these concepts by using examples and then formalize them in the context of the Markov decision processes.

1.1 A grid-world example

Consider the example shown in Figure 1.1, where a robot moves in a grid world. The robot or called *agent* can move across adjacent cells in the grid. Each time it can only occupy a single cell. The white cells are *accessible* and the orange ones are *forbidden* to enter. There is a *target* cell that the robot would like to reach. We will use such kind of grid world examples throughout this book considering that they are intuitive to illustrate new concepts and algorithms.

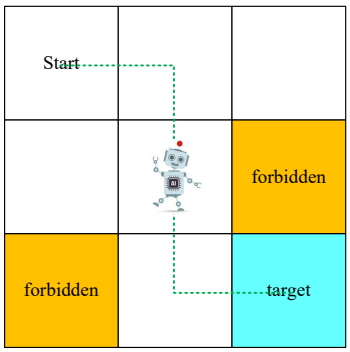


Figure 1.1: The grid-world example used throughout the book.

The ultimate goal of the agent is to find a "good" policy to reach the target cell starting from any initial cell. How to define the goodness of a policy? The intuition is that the agent should reach the target without entering the forbidden cells, taking unnecessary detours, or colliding with the boundary of the grid.

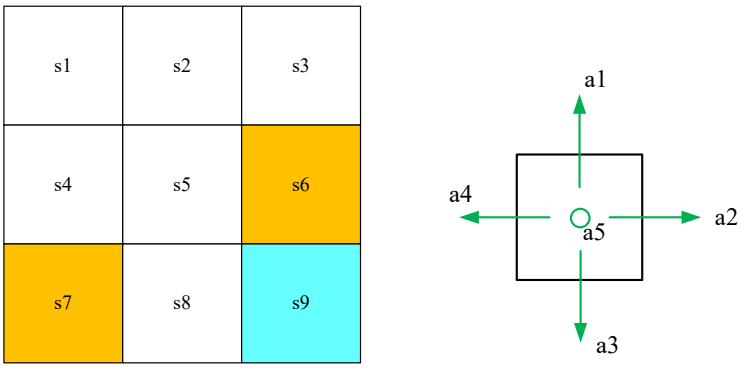
It would be trivial to plan a path to reach the target cell if the agent knows the map of the grid world. The task becomes nontrivial if the agent does not know any information

about the environment in advance. Then, the agent has to interact with the environment to find a good policy to reach the target cell by trial-and-error. To do that, we introduce some necessary concepts in the rest of the chapter.

1.2 State and action

The first concept to be introduced is called *state*, which describes the status of the agent with respect to the environment. In the grid-world example, the state corresponds to the location of the agent. Since there are nine possible locations/cells, there are nine states as well. They are indexed as $s_1, s_2, \ldots, s_9$. The set of all the states is called *state space*, denoted as $\mathcal{S} = \{s_1, \ldots, s_9\}$ (see Figure 1.2(a)).

For each state, the agent has five possible *actions* to take: move upwards, rightwards, downwards, leftwards, and stay unchanged. The five actions are denoted as $a_1, a_2, \ldots, a_5$, respectively (see Figure 1.2(b)). The set of all the actions is called *action space*. In particular, the action space of s_i is denoted as $\mathcal{A}(s_i)$. Here, we suppose that different states have the same action space: $\mathcal{A}(s_i) = \mathcal{A} = \{a_1, \ldots, a_5\}$.



(a) States (b) Actions

Figure 1.2: (a) In the grid-world example, there are nine states $\{s_1, \ldots, s_9\}$. (b) Each state has five possible actions $\{a_1, a_2, a_3, a_4, a_5\}$.

It should be noted that different states may have different action spaces in general. For instance, taking a_1 or a_4 at state s_1 would collide with the boundary of the grid. Then, we can remove these two actions from the state space of s_1 and hence $\mathcal{A}(s_1) = \{a_2, a_3, a_5\}$. Removing useless actions from the action space is favorable to simplify policy search processes in RL. However, such removal is based on the knowledge that some actions are not good. Such knowledge sometimes can be obtained easily and sometimes must be learned. In fact, the basic purpose of RL is to find out which actions are good and which are not. In this book, we consider the most general and challenging case that no actions are removed from the action space.

1.3 State transition

When taking an action, the agent may move from one state to another. Such a process is called *state transition*. For example, if the agent is at state s_1 and takes action a_2 (that is to move rightwards), then the agent would move to state s_2. Such a process can be expressed as

$$s_1 \xrightarrow{a_2} s_2.$$

What is the next state when the agent attempts to go out of the boundary, for example, taking action a_1 at state s_1? The answer is that the agent will get bounced back because it is impossible for the agent getting out of the state space. Hence, we have $s_1 \xrightarrow{a_1} s_1$.

What is the next state when the agent attempts to enter the forbidden cells, for example, taking action a_2 at state s_5? There are two different scenarios. The first is that, although s_6 is forbidden, it is still accessible but with a severe penalty. In this case, the next state is s_6 and hence the state transition process is $s_5 \xrightarrow{a_2} s_6$. The second scenario is that s_6 is physically not assessable. For example, it is surrounded by walls. The agent will get bounced back to s_5 if it attempts to move rightwards. In this case, the state transition is $s_5 \xrightarrow{a_2} s_5$. Which scenario should we consider? The answer depends on the physical environment. Since the grid-world example is based on numerical simulation, we can choose either. Here, we consider the first scenario that the forbidden cells are still accessible, though stepping into them may get punished. This scenario is more general and interesting.

Since we are considering a simulation task, we can define the state transition process whatever we like. If the system involves physical interaction, the state transition process would be determined by physical laws.

How to describe the state transition process? The state transition process is defined for each state and their associated actions. Therefore, it can be represented using a table as in Table 1.1. In this table, each row represents a state and each column represents an action.

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	s_1	s_2	s_4	s_1	s_1
s_2	s_2	s_3	s_5	s_1	s_2
s_3	s_3	s_3	s_6	s_2	s_3
s_4	s_1	s_5	s_7	s_4	s_4
s_5	s_2	s_6	s_8	s_4	s_5
s_6	s_3	s_6	s_9	s_5	s_6
s_7	s_4	s_8	s_7	s_7	s_7
s_8	s_5	s_9	s_8	s_7	s_8
s_9	s_6	s_9	s_9	s_8	s_9

Table 1.1: A tabular representation of state transition. Each cell indicates the next state to transit to after the agent taking an action at a state.

Though intuitive, the tabular representation is only able to describe deterministic state transition processes. It is necessary to use rigorous mathematics to describe state transition in a general way. For example, for s_1 and a_2, the conditional probability distribution is

$$
\begin{aligned}
p(s_1|s_1, a_2) &= 0, \\
p(s_2|s_1, a_2) &= 1, \\
p(s_3|s_1, a_2) &= 0, \\
p(s_4|s_1, a_2) &= 0, \\
p(s_5|s_1, a_2) &= 0,
\end{aligned}
$$

which indicates that, if taking a_2 at s_1, the probability for the agent to move to s_2 is one and the others is zero. As a result, taking action a_2 at s_1 will certainly take the agent to s_2. Preliminaries of conditional probability are given in the appendix. Readers are strongly advised to get familiar with probability theory first because it is necessary to study RL and will be used throughout this book.

It is noted that the state transition in the above example is *deterministic*. In general, state transition can be *stochastic*. For instance, if there are random wind gusts across the grid, when taking action a_2 at s_1, the agent may be blew to s_5 instead of s_2. In this case, we have $p(s_5|s_1, a_2) > 0$. However, we merely consider the deterministic case for the sake of simplicity.

1.4 Policy

A *policy* tells the agent which actions to take at each state.

Intuitively, policies can be depicted as arrows as shown in Figure 1.3. Following the policy starting from different initial states, the agent would generate some trajectories across different states as shown in Figure 1.3.

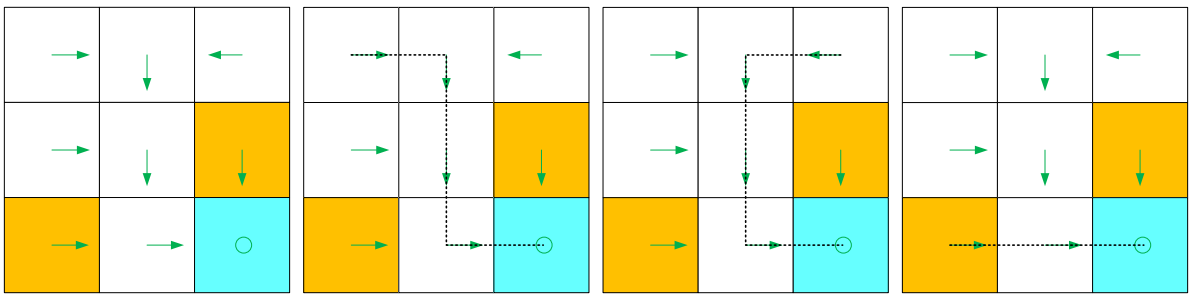


Figure 1.3: A policy represented by arrows and some trajectories obtained starting from different states.

Mathematically, policies can be described by conditional probabilities. Denote the policy in Figure 1.3 as $\pi(a|s)$, which is a conditional probability distribution function defined for *every* state-action pair. For example, the policy for s_1 as shown in Figure 1.3

is

$$\pi(a_1|s_1) = 0,$$
$$\pi(a_2|s_1) = 1,$$
$$\pi(a_3|s_1) = 0,$$
$$\pi(a_4|s_1) = 0,$$
$$\pi(a_5|s_1) = 0,$$

which indicates that the probability of taking action a_2 at state s_1 is one and the others zero. Therefore, the agent would definitely select action a_2 at s_1.

The above policy is *deterministic*. Policies may be *stochastic* in general. For example, the policy shown in Figure 1.4 is stochastic: at state s_1, the agent may take actions to go either rightwards or downwards. The probabilities of taking the two actions are the same as 0.5. In this case, the policy is

$$\pi(a_1|s_1) = 0,$$
$$\pi(a_2|s_1) = 0.5,$$
$$\pi(a_3|s_1) = 0.5,$$
$$\pi(a_4|s_1) = 0,$$
$$\pi(a_5|s_1) = 0.$$

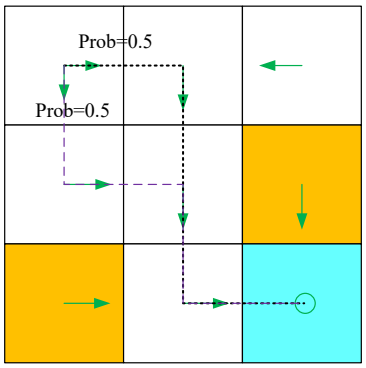


Figure 1.4: A stochastic policy. At state s_1, the agent may move rightwards or downwards with the equal probability of 0.5.

Policies represented by conditional probabilities can be intuitively described as tables. For example, Table 1.2 represents the stochastic policy depicted in Figure 1.4. The entry at the ith row and jth column is the probability to take the jth action at the ith state. Such kind of representation is called *tabular representation*, which will be used throughout this book. However, towards the end of this book, we will introduce another way to represent policies as parameterized functions instead of tables.

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	0	0.5	0.5	0	0
s_2	0	0	1	0	0
s_3	0	0	0	1	0
s_4	0	1	0	0	0
s_5	0	0	1	0	0
s_6	0	0	1	0	0
s_7	0	1	0	0	0
s_8	0	1	0	0	0
s_9	0	0	0	0	1

Table 1.2: A tabular representation of a policy. Each entry indicates the probability of taking an action at a state.

1.5 Reward

Reward is one of the most unique concepts in RL.

After taking an action, the agent would get a reward as feedback from the environment. The reward can be a positive, negative, or zero real number. Different rewards have different impacts on the policy that the agent would eventually learn. In particular, by a positive reward, we encourage the agent to take that action. By a negative reward, we discourage the agent to take that action. What if the agent receives a zero reward after taking an action? A zero reward can be interpreted as encouragement because the agent loses nothing if taking that action.

In the grid-world example, the rewards are designed as follows:

– If the agent attempts to get out of the boundary, let $r_{\text{boundary}} = -1$.

– If the agent attempts to enter a forbidden cell, let $r_{\text{forbidden}} = -1$.

– If the agent reaches the target cell, let $r_{\text{target}} = +1$.

– Otherwise, the agent gets a reward of $r = 0$.

Reward can be interpreted as a human-machine interface, with which we can guide the agent to behave as we expect. For example, with the above designed rewards, we can expect that the agent will try to avoid getting out of the boundary or stepping into the forbidden cells. The basic idea behind it is that optimal policies aim to collect rewards as great as possible. At this moment, it is still unclear how to define the optimality of policies. It will be discussed when we study the Bellman optimality equation.

Designing appropriate rewards is an important step of RL. This step is, however, nontrivial for complex tasks since it may require us to have a good understanding of the problem. Nevertheless, designing rewards and using RL to solve a complex problem may still be much easier than solving the problem using other approaches that require an even deeper understanding of the problem and professional background. This is one of the reasons why RL can attract a more broad range of users.

Reward transition is the process of getting a reward after taking an action. This process can be intuitively represented as a table as shown in Table 1.3. Each row of the table corresponds to a state and each column corresponds to an action. The value in each cell of the table indicates the reward that can be obtained by taking an action at a state.

One question that beginners may ask is, if given the table of rewards, whether we can find good policies by simply selecting the actions with the greatest rewards. The answer is no. That is because the above rewards are *immediate rewards* that can be obtained after taking an action. In order to determine a good policy, we need to consider the total reward obtained in a long run (see next section). It is often the case that an action that has the greatest immediate reward may not lead to the greatest total reward.

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	r_{boundary}	0	0	r_{boundary}	0
s_2	r_{boundary}	0	0	0	0
s_3	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$	0	0
s_4	0	0	$r_{\text{forbidden}}$	r_{boundary}	0
s_5	0	$r_{\text{forbidden}}$	0	0	0
s_6	0	r_{boundary}	r_{target}	0	$r_{\text{forbidden}}$
s_7	0	0	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$
s_8	0	r_{target}	r_{boundary}	$r_{\text{forbidden}}$	0
s_9	$r_{\text{forbidden}}$	r_{boundary}	r_{boundary}	0	r_{target}

Table 1.3: A tabular representation of reward transition. Here, the reward transition is deterministic and each cell indicates how much reward can be obtained after the agent taking an action at a given state.

Though intuitive, the tabular representation is only able to describe deterministic reward transition processes. A more powerful way is to use conditional probabilities $p(r|s,a)$ to describe general reward transition. For example, for state s_1, we have

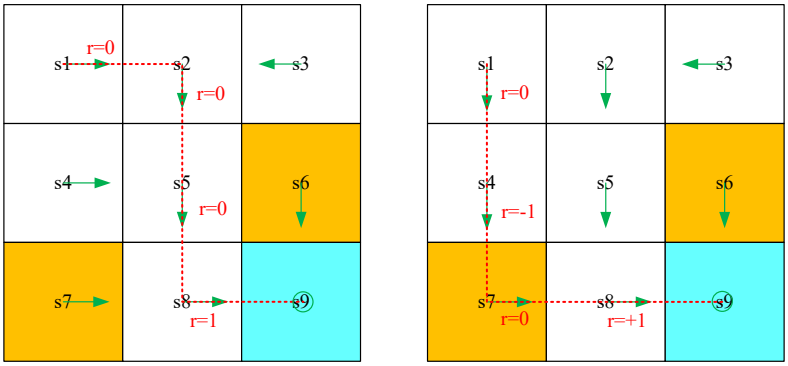
$$p(r=-1|s_1,a_2)=1, \quad p(r\neq -1|s_1,a_2)=0,$$

which indicates that, if taking a_2 at s_1, it is certain that the agent would get $r=-1$. Again, in this example, the reward transition process is deterministic. In general, it can be stochastic. For example, if a student studies hard, he/she would get a positive reward in general (for example, higher grades in exams), but how much is uncertain.

One problem that may confuse beginners is whether a reward should depend on the action taken or the next state reached. A mathematical rephrase of this problem is whether we should use $p(r|s,a)$ or $p(r|s,s')$ to represent reward transition. Here, s' is the next state reached after leaving the state s. In the grid-world examples, since the state transition is deterministic, the two ways happen to be equivalent. For example, $p(r|s_1,a_2)=p(r|s_1,s_2)$ since the agent taking a_2 at s_1 will certainly lead to s_2. However, when the state transition is stochastic, the two ways are not equivalent anymore. More importantly, we should always use $p(r|s,a)$ instead of $p(r|s,s')$ because we want to punish

or encourage an action instead of the next state. For example, suppose the current state is s_1. Although taking actions a_1 and a_5 will both lead to the next state as s_1, taking a_1 is worse than a_5 because a_1 attempts to collide to the boundary and should be given negative rewards.

1.6 Trajectory, return, and episode



(a) (b)

Figure 1.5: Trajectories obtained following two policies. The trajectories are indicated by red dashed lines.

A *trajectory* is a state-action-reward chain.

For example, given the policy shown in Figure 1.5(a), starting from s_1, the agent follows a trajectory as

$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9.$$

The *return* of this trajectory is the sum of all the rewards collected along the trajectory:

$$\text{return} = 0 + 0 + 0 + 1 = 1. \tag{1.1}$$

Return is also sometimes called *total rewards* or *cumulative rewards*.

Return can be used to evaluate the "goodness" of policies. For example, we can compare the two policies in Figure 1.5 by comparing their returns. In particular, starting from s_1, the return obtained by the left policy is 1 as calculated above. For the right policy, starting from s_1 gives the trajectory as

$$s_1 \xrightarrow[r=0]{a_3} s_4 \xrightarrow[r=-1]{a_3} s_7 \xrightarrow[r=0]{a_2} s_8 \xrightarrow[r=+1]{a_2} s_9.$$

The corresponding return is:

$$\text{return} = 0 - 1 + 0 + 1 = 0.$$

The values of the returns for the two policies indicate that the left policy is better than the right one since its return is greater. This mathematical conclusion is also consistent with the intuition that the right policy is not good since it passes through a forbidden cell.

While reward only reflects the encouragement or discouragement for taking a single action, return can be used to evaluate a sequence of actions in a long run. A return consists of the *immediate reward* and the *delayed reward*. Here, the immediate reward is the reward obtained after taking an action at the initial state; the delayed reward is the sum of the rewards obtained after leaving the initial state. Although the immediate reward may be negative, the delayed reward may be positive. Thus, which actions to take should be determined by the return (i.e., total reward) rather than the immediate reward to avoid short-sighted decisions. As we will see in the next chapter, return plays an important role to evaluate different policies.

The return in (1.1) is defined for a finite-length trajectory. Return can also be defined for infinitely long trajectories. For example, the trajectory in Figure 1.5 stops after reaching s_9. Such a stop relies on a stop criterion: that is, the agent stops moving after reaching the target state. Since the policy is also well defined for the target state s_9, the agent does not have to stop after reaching s_9. Then, we obtain the following infinitely long trajectory:

$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9 \xrightarrow[r=1]{a_5} s_9 \xrightarrow[r=1]{a_5} s_9 \dots$$

The direct summation of the rewards obtained along this trajectory is

$$\text{return} = 0 + 0 + 0 + 1 + 1 + 1 + \cdots = \infty,$$

which unfortunately diverges. Therefore, we need to introduce the concept of *discounted return* for infinitely long trajectories. In particular, the discounted return is the sum of the discounted rewards:

$$\text{discounted return} = 0 + \gamma 0 + \gamma^2 0 + \gamma^3 1 + \gamma^4 1 + \gamma^5 1 + \dots, \tag{1.2}$$

where $\gamma \in (0, 1)$ is called the *discount rate*. When $\gamma \in (0, 1)$, (1.2) is finite and can be calculated as

$$\text{discounted return} = \gamma^3(1 + \gamma + \gamma^2 + \dots) = \gamma^3 \frac{1}{1-\gamma}.$$

The reason that we consider discounted return is twofold. First, it removes the stop criterion and the mathematics is more elegant. Second, the discount rate can be used to adjust the emphasis on near or far future rewards. In particular, if γ is close to 0, then the user put more emphasis on the reward obtained in the near future. The resulting

policy would be short-sighted. If γ is close to 1, then the agent put more emphasis on the far future rewards. In this case, the resulting policy would dare to take risks of getting negative rewards in the near future. These points can be well demonstrated later by the examples in Section 3.5 in Chapter 3.

One important notion that was not explicitly mentioned in the above discussion is *episode*. When interacting with the environment following a policy, the agent may stop at some *terminal states*. The resulting trajectory is called an *episode* or a *trial*. If the environment or policy is stochastic, we would obtain different episodes starting from the same state. However, if everything is deterministic, we may always obtain the same episode starting from the same state.

An episode is usually assumed to be a finite trajectory. Tasks with episodes are called *episodic tasks*. However, some tasks may have no terminal states, meaning the interaction with the environment will never end. Such tasks are called *continuing tasks*. In fact, we can treat episodic and continuing tasks in a unified mathematical way by converting episodic tasks to continuing tasks. The key is to well define the process after reaching the target/terminal state. Specifically, after reaching the target or terminal state in an episodic task, the agent can continue taking actions. We can treat the target/terminal state in two ways.

First, if we treat it as a special state, we can specially design its action space or state transition such that the agent stays at this state forever. Such states are called *absorbing states*, meaning that the agent would never leave the state once it is reached. For example, for the target state s_9, we can specify $\mathcal{A}(s_9) = \{a_5\}$ or set $\mathcal{A}(s_9) = \{a_1, \ldots, a_5\}$ but $p(s_9|s_9, a_i) = 1$ for all $i = 1, \ldots, 5$. We can also set the reward obtained after reaching s_9 as always zero.

Second, if we treat the target state as a normal one, we can simply set its action space the same as others and the agent may leave the state. Since a positive reward of $r = 1$ can be obtained every time s_9 is reached, the agent will eventually learn to stay at s_9 forever to collect more rewards. Of course, when the episode is infinitely long and the reward of staying at s_9 is positive, a discount rate must be used to calculate the discounted return to avoid divergence. In this book, we consider the second scenario where the target state is treated as a normal state whose action space is $\mathcal{A}(s_9) = \{a_1, \ldots, a_5\}$.

1.7 Markov decision process

The previous sections of this chapter have illustrated some fundamental concepts in RL by examples. This section presents these concepts in a more formal way under the framework of the Markov decision processes (MDP).

MDP is a general framework to describe stochastic dynamical systems. The key ingredients of an MDP are listed below.

– Sets:

- State set: the set of all states, denoted as $\mathcal{S}$.
- Action set: a set of actions, denoted as $\mathcal{A}(s)$, is associated for each state $s \in \mathcal{S}$.
- Reward set: a set of rewards, denoted as $\mathcal{R}(s,a)$, is associated for each state action pair (s,a).

– Model:

- State transition probability: at state s, taking action a, the probability to transit to state s' is $p(s'|s,a)$.
- Reward transition probability: at state s, taking action a, the probability to get reward r is $p(r|s,a)$.

– Policy: at state s, the probability to choose action a is $\pi(a|s)$.

– Markov property: One key property of MDPs is the *Markov property*, which refers to the memoryless property of a stochastic process. Mathematically, it means

$$\begin{aligned} p(s_{t+1}|s_t,a_t,s_{t-1},a_{t-1},\ldots,s_0,a_0) &= p(s_{t+1}|s_t,a_t), \\ p(r_{t+1}|s_t,a_t,s_{t-1},a_{t-1},\ldots,s_0,a_0) &= p(r_{t+1}|s_t,a_t). \end{aligned} \tag{1.3}$$

That is, the next state or reward merely depends on the present state and action rather than the previous states or actions. Equation (1.3) indicates the property of *conditional independence* of random variables. Preliminaries to the probability theory can be found in the appendix. Markov property is important for deriving the fundamental Bellman equation of MDPs as shown in the next chapter.

The state transition, reward transition, and policy can be all stochastic, although they are sometimes deterministic in our illustrative examples. Here, $p(s'|s,a)$ and $p(r|s,a)$ for all s,a are called the *model* or *dynamics* of an MDP. We will show later in the book that there are *model-based* and *model-free* RL algorithms. Moreover, the model can be either *stationary* or *nonstatinary*, or in other words, time-variant or time-invariant. In stationary environments, the models do not change over time; in nonstationary environments, the models may vary over time. For instance, in the example of grid world, if some forbidden areas may pop up or disappear in the grid, such an environment is nonstationary.

The reader may have also heard about the Markov process (MP). What is the difference between MDP and MP? The answer is that, once the policy in an MDP is fixed, the MDP degenerates to an MP. In the literature on stochastic processes, a Markov process is also called a Markov chain if it is discrete-time and the number of states is finite or countable [1]. In our book, the Markov process and Markov chain are used interchangeably when the context is clear. In the first half of this book, we consider *finite* MDPs where the numbers of states, actions, and reward values are all finite. This is the simplest case that should be well understood first of all. Towards the end of this book, we will

consider the case of *infinite* states or actions.

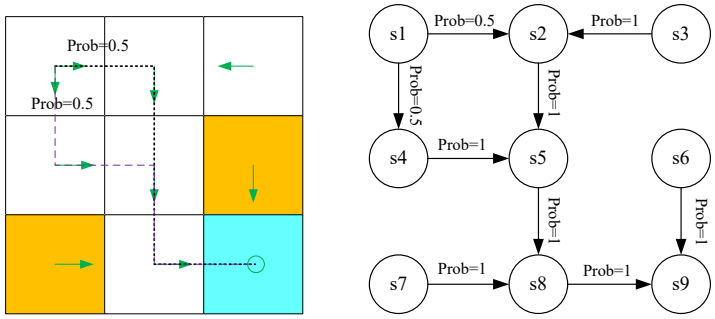
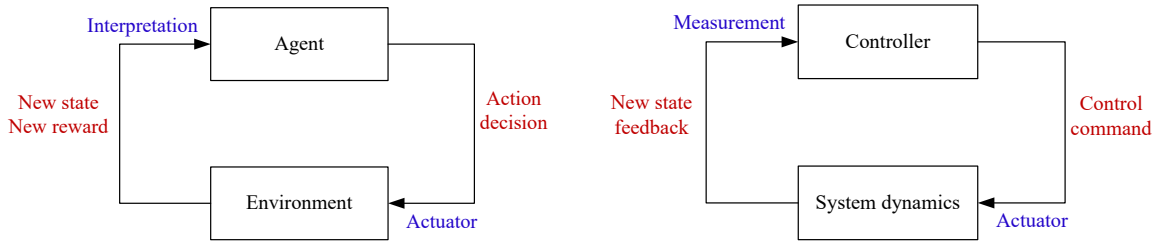


Figure 1.6: Abstraction of the grid-world example as a Markov process.

The grid-world example in Figure 1.6 can be abstracted as a *Markov decision process* (MDP), where the circles represent states and the links with arrows represent the state transition.

Finally, RL can be described as an agent-environment interaction process (see Figure 1.7). The *agent* is a decision-maker that can sense its own state, maintain policies, and execute actions. Everything outside of the agent is regarded as the *environment*. In the grid-world examples, the agent and environment refer to the robot and grid world, respectively. As depicted by Figure 1.7, after the agent decides to take an action, the actuator would execute such a decision. Then, the state of the agent would be changed and a reward can be obtained. By using interpreters, the agent can interpret the new state and reward to make the next decision. Thus, a closed-loop is formed.

If the reader is familiar with control theory, the agent-environment paradigm of RL is very similar to that of control systems. See Figure 1.7. In addition to different names of the ingredients, the key difference between RL and automatic control is the reward, which is a favorable human-machine interface in RL.



(a) Diagram of RL (b) Diagram of control systems

Figure 1.7: Diagrams of RL and control systems.

1.8 Summary

This chapter introduced some basic concepts in RL. We used intuitive grid-world examples to demonstrate these concepts and then formalized them in the framework of MDP. In this

chapter, we know that RL can be described as an agent-environment interaction process. The agent is a composition of a sensor, decision-maker, and actuator. That is, the agent is able to sense its own state, maintain a policy, and execute actions. The dynamics of the environment are described by state and reward probability transition probabilities. Through these probabilities, taking an action would change the state of the agent and, in the meantime, generate a reward signal. Such a reward is the feedback for the action taken and can be used by the agent to adjust its policy. For more information about the Markov decision processes, readers are advised to refer to [1, 2].