Reinforcement Learning

Part I

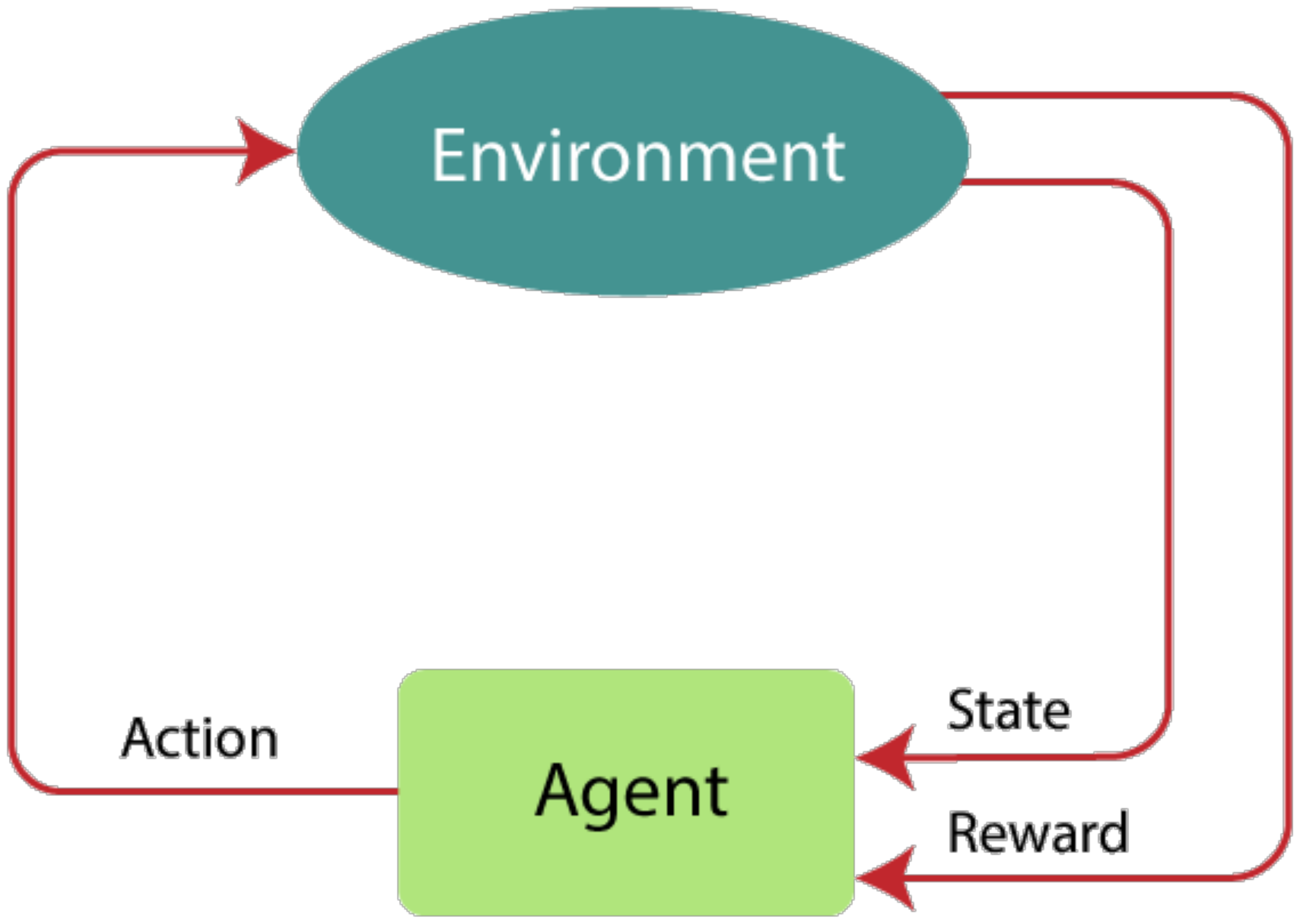
Prepared by: Joseph Bakarji

Example: OpenAI's Multi-Agent Hide and Seek

https://www.youtube.com/watch?v=kopoLzvh5jY&ab_channel=OpenAI

What are the features of an intelligent agent?

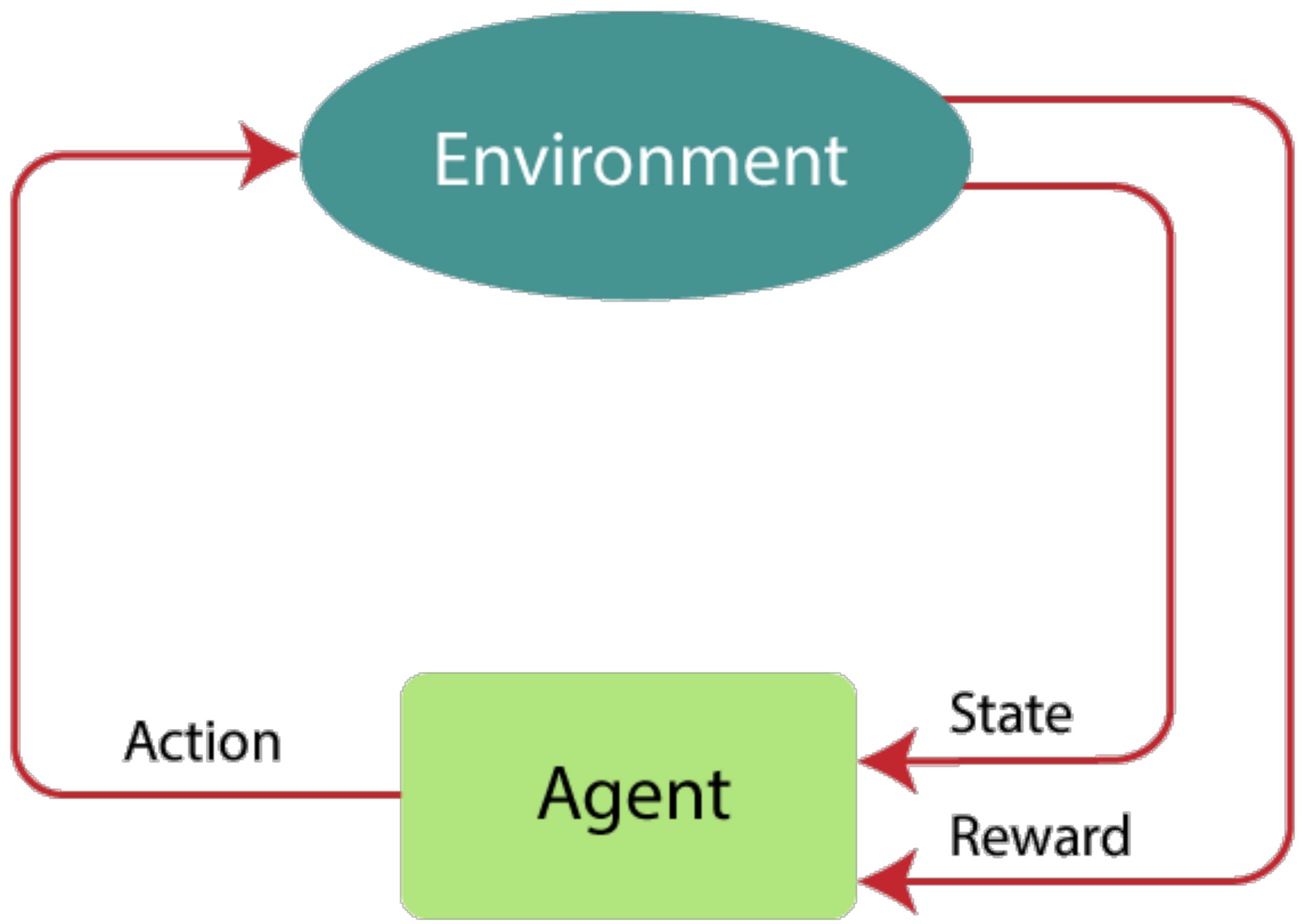
- **Actions**: make decisions and act from a set of possible
- **Perceptions**: observe the world and one's own state



https://www.javatpoint.com/reinforcement-learning

What are the features of an intelligent agent?

- Plan for long-term consequences: decisions today will affect the state tomorrow and later, etc.
- Decisions will determine how much information you collect from the environment
- No or little supervision: collect more data interactively



Example

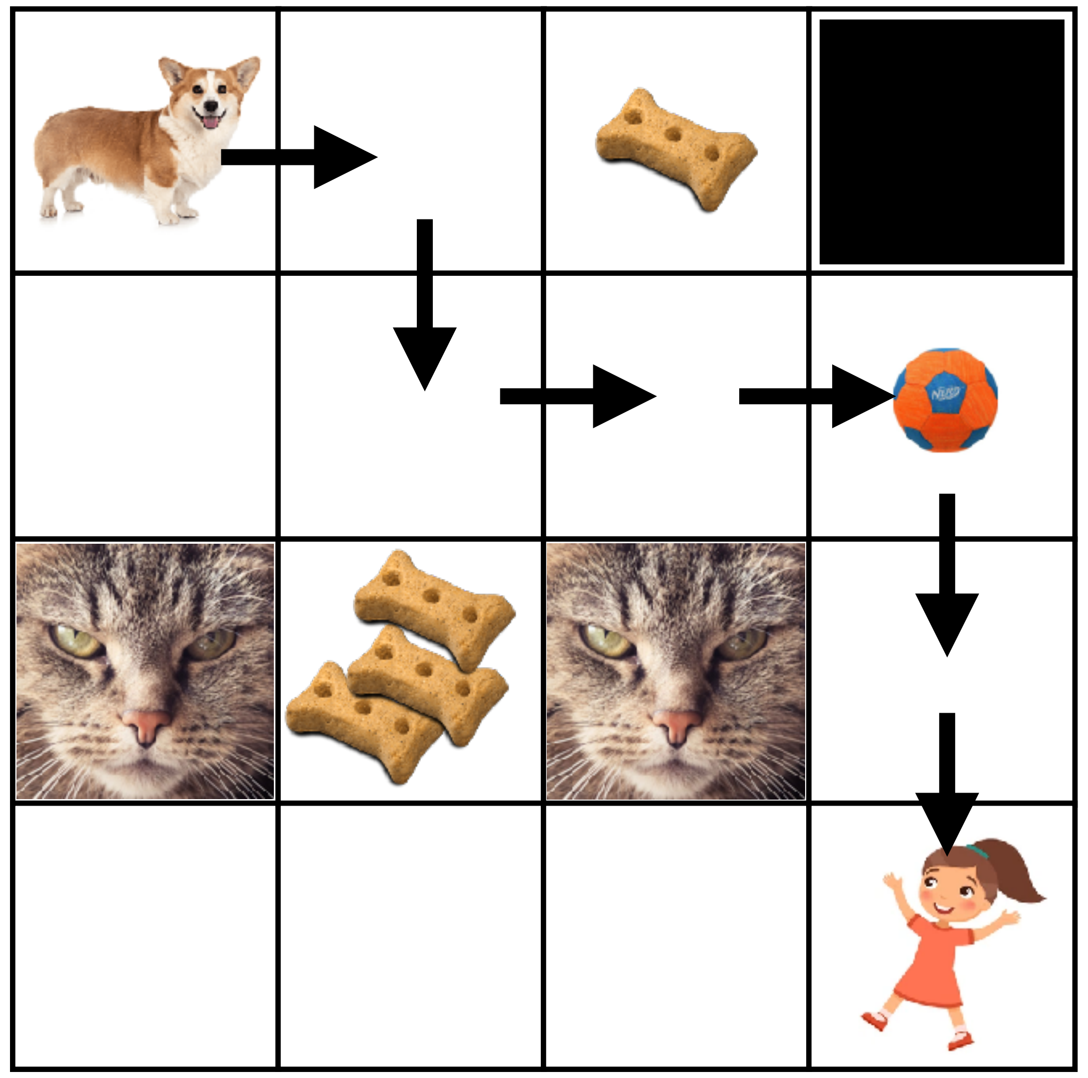
States: $s \in \{(1,1), (1,2), \ldots, (4,4)\}$

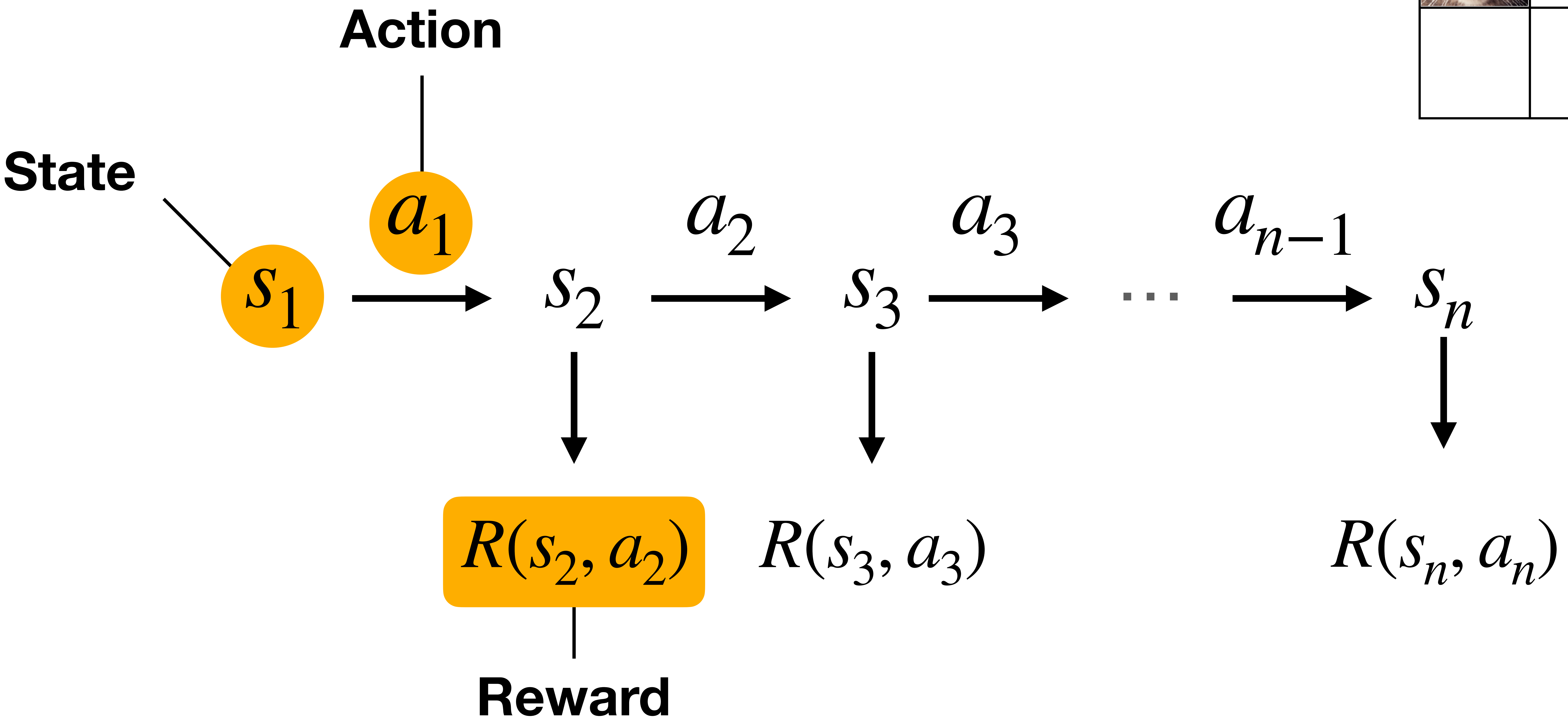
Actions: $a \in \{ \rightarrow, \leftarrow, \uparrow, \downarrow \}$

Rewards: $R(s_i) \in [-5, 20]$

Transition probability: $P(s' \mid s, a) \in [0, 1]$

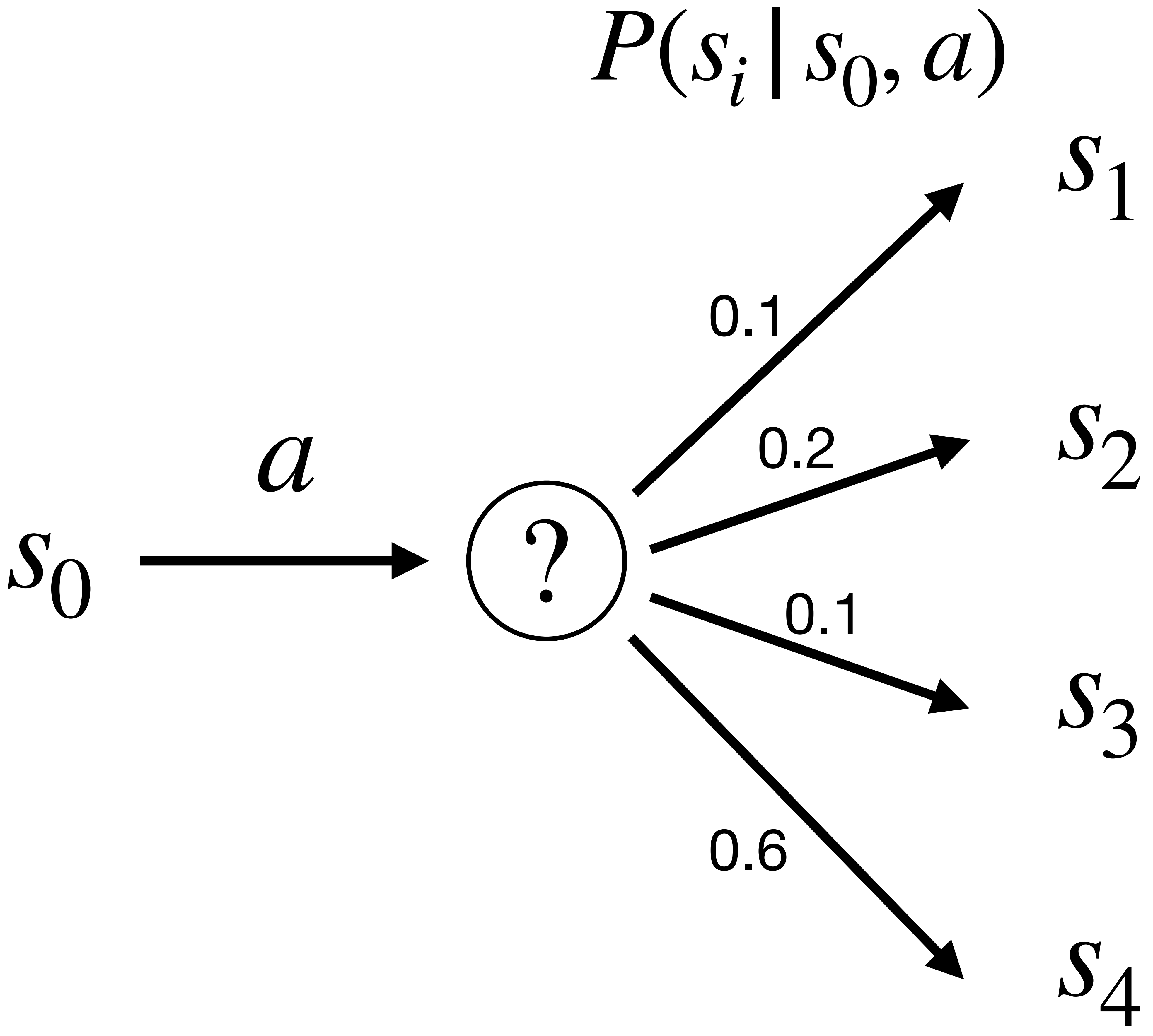
Definitions

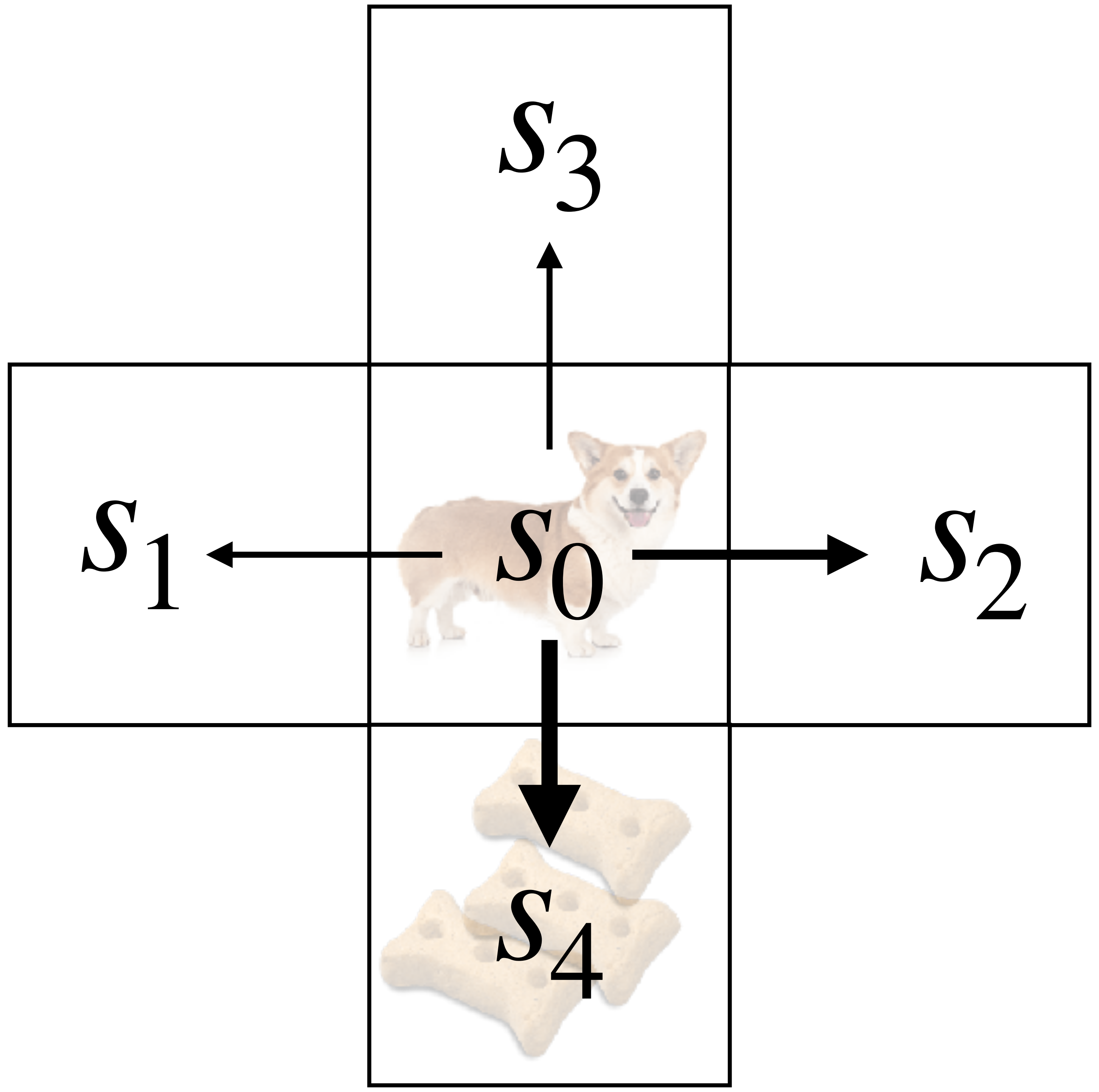




Markov Decision Process (MDP)

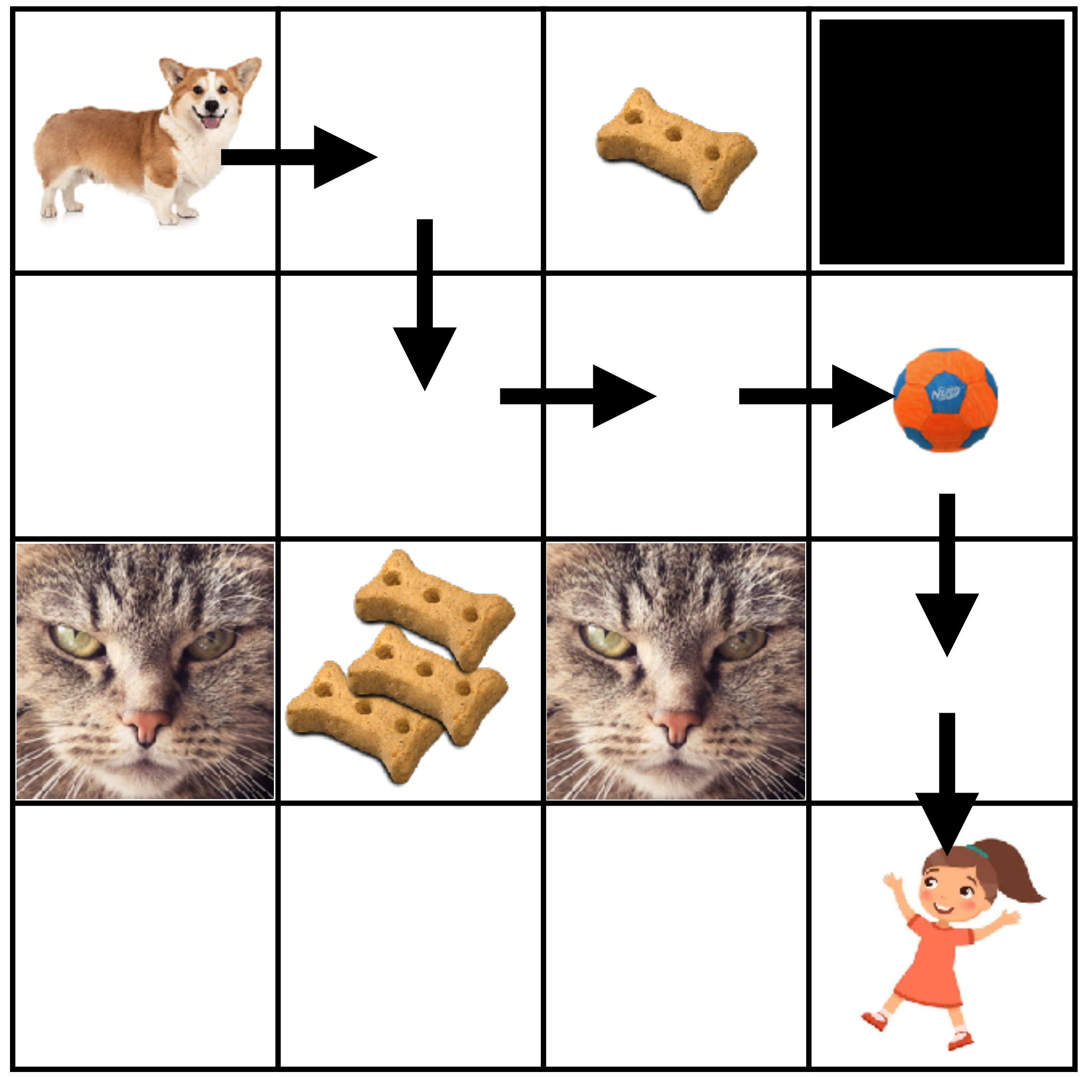
Assumes a probabilistic (uncertain) world

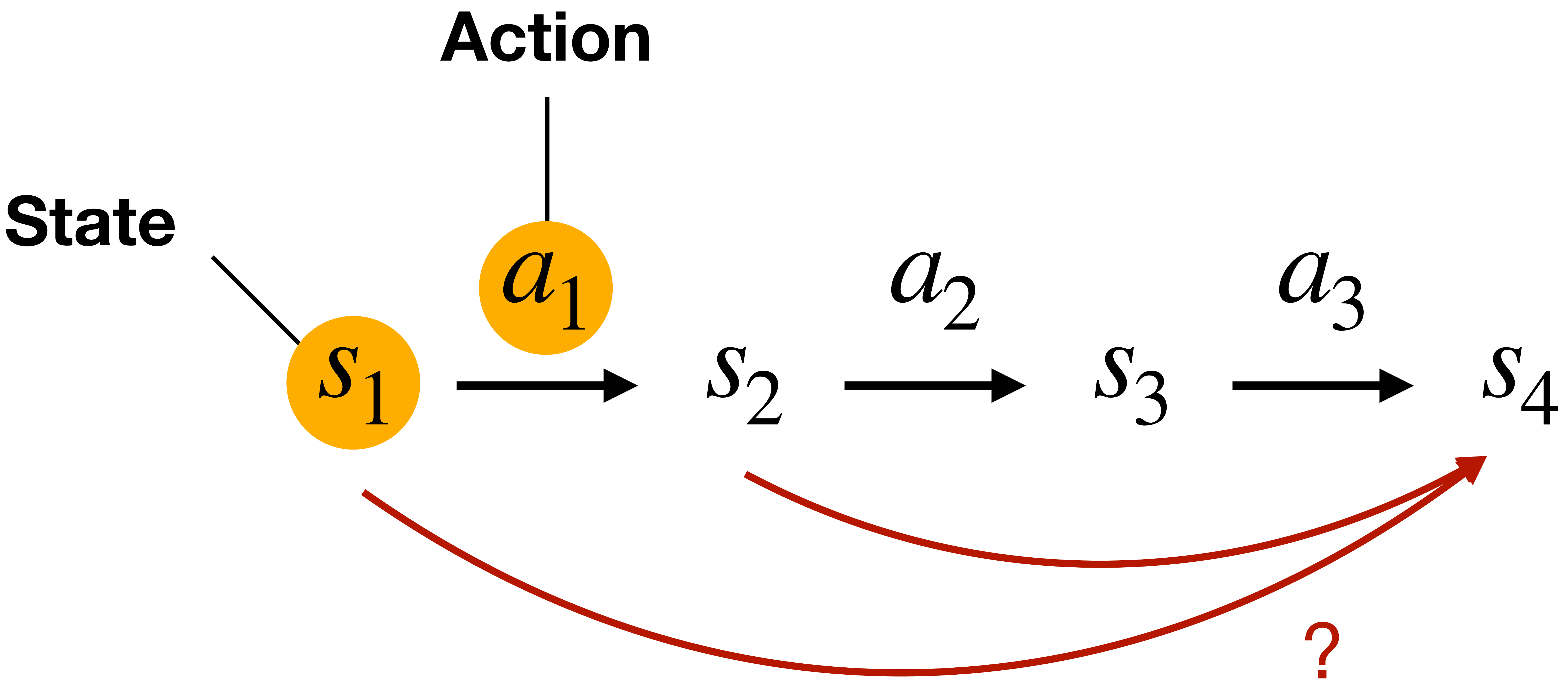




Markov Decision Process (MDP)

The Markov assumption:
transition probability only depends on previous state

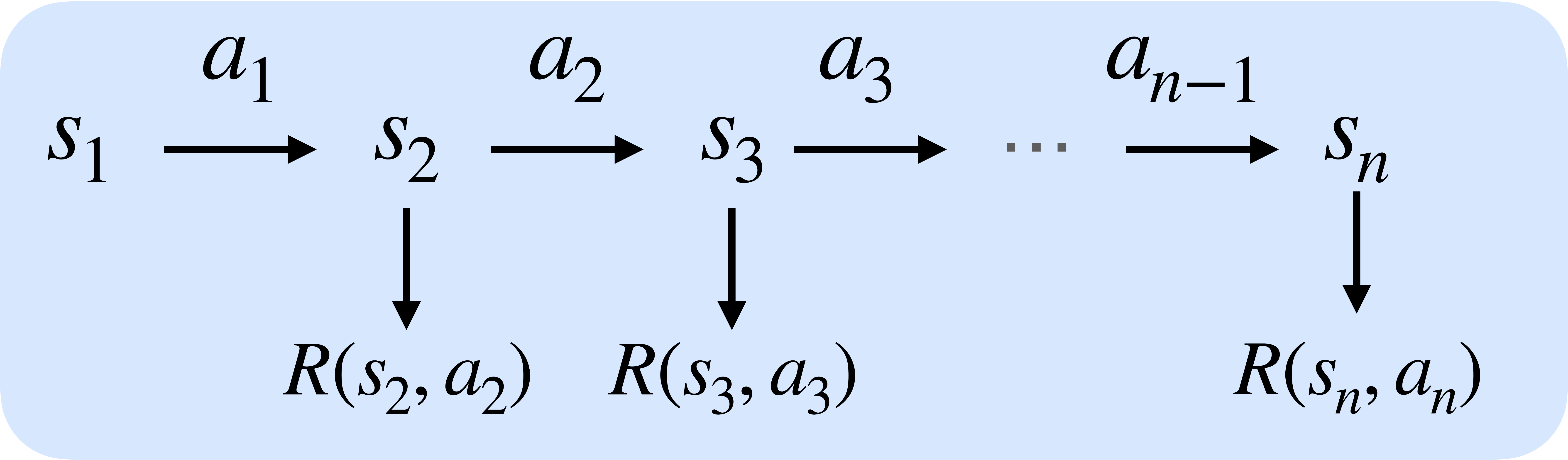




$$P(s_i \mid s_{i-1}, s_{i-2}, s_{i-3}, a_i, a_{i-1}, \ldots) = P(s_i \mid s_{i-1}, a_i)$$

Goal: Maximize Total Rewards

Definition of *Total Rewards*



$$\text{Total Rewards} = R(s_1, a_1) + \gamma R(s_2, a_2) + \gamma^2 R(s_3, a_3) + \ldots$$

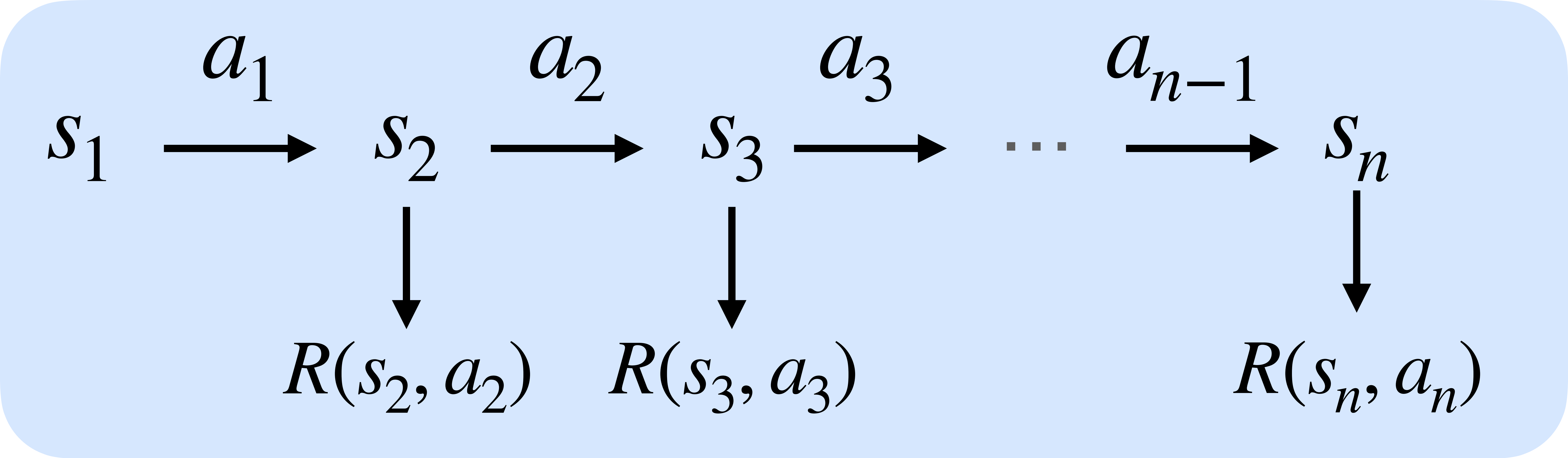
Discount Factor

$\gamma < 1$

Favors immediate rewards over future ones

Goal: Maximize Total Rewards

Definition of *Total Rewards*



Total Rewards $= R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \ldots$

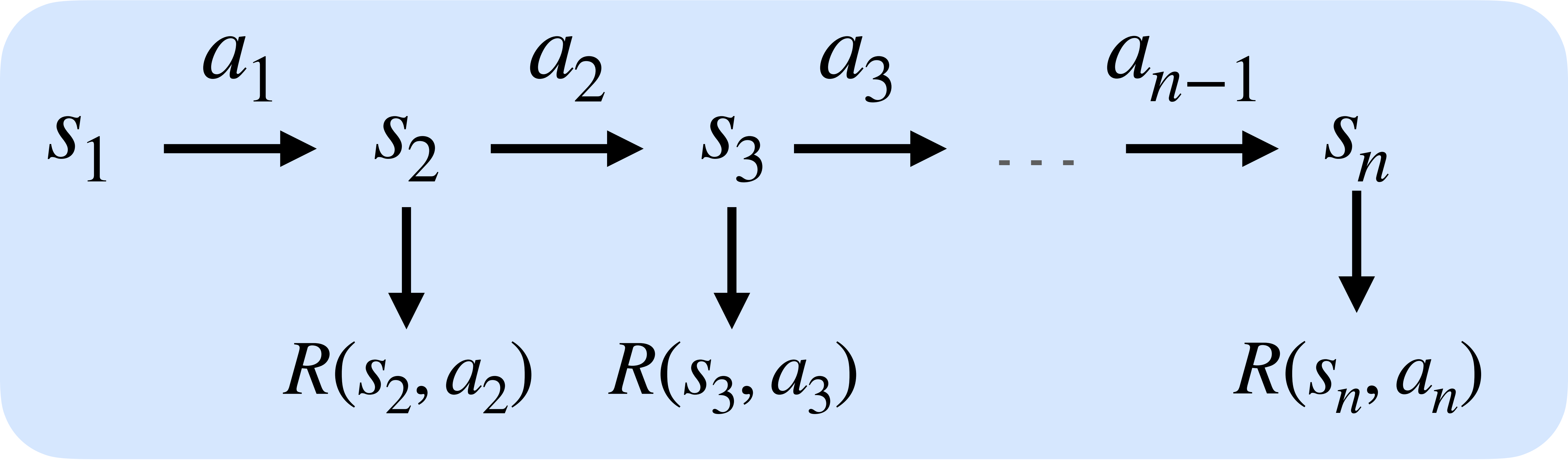
Independence of actions

Discount Factor

Favors immediate rewards over future ones

Goal: Maximize **Expectation** of Total Rewards

Given that decisions are probabilistic, we want to maximize the **expected** rewards



$$\text{Total Expected Rewards} = \mathbb{E}\left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \ldots\right]$$

Expectation or average

Goal: Maximize Expectation of Total Rewards

Given that decisions are probabilistic, we want to maximize the **expected** rewards

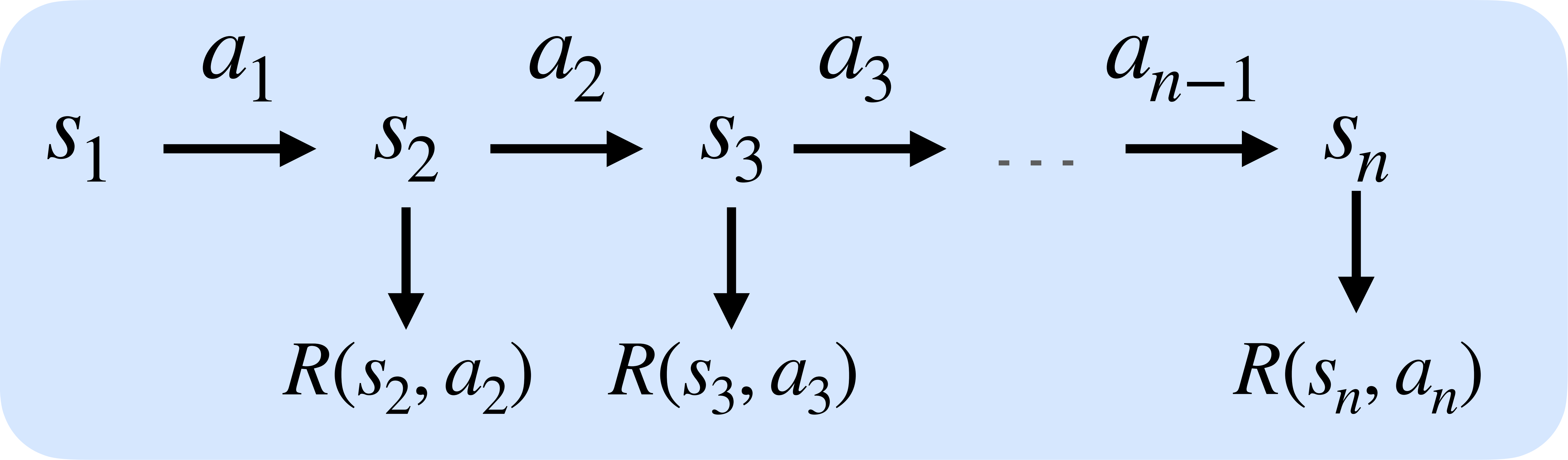
$$\text{Total Expected Rewards} = \mathbb{E}\left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \ldots\right]$$

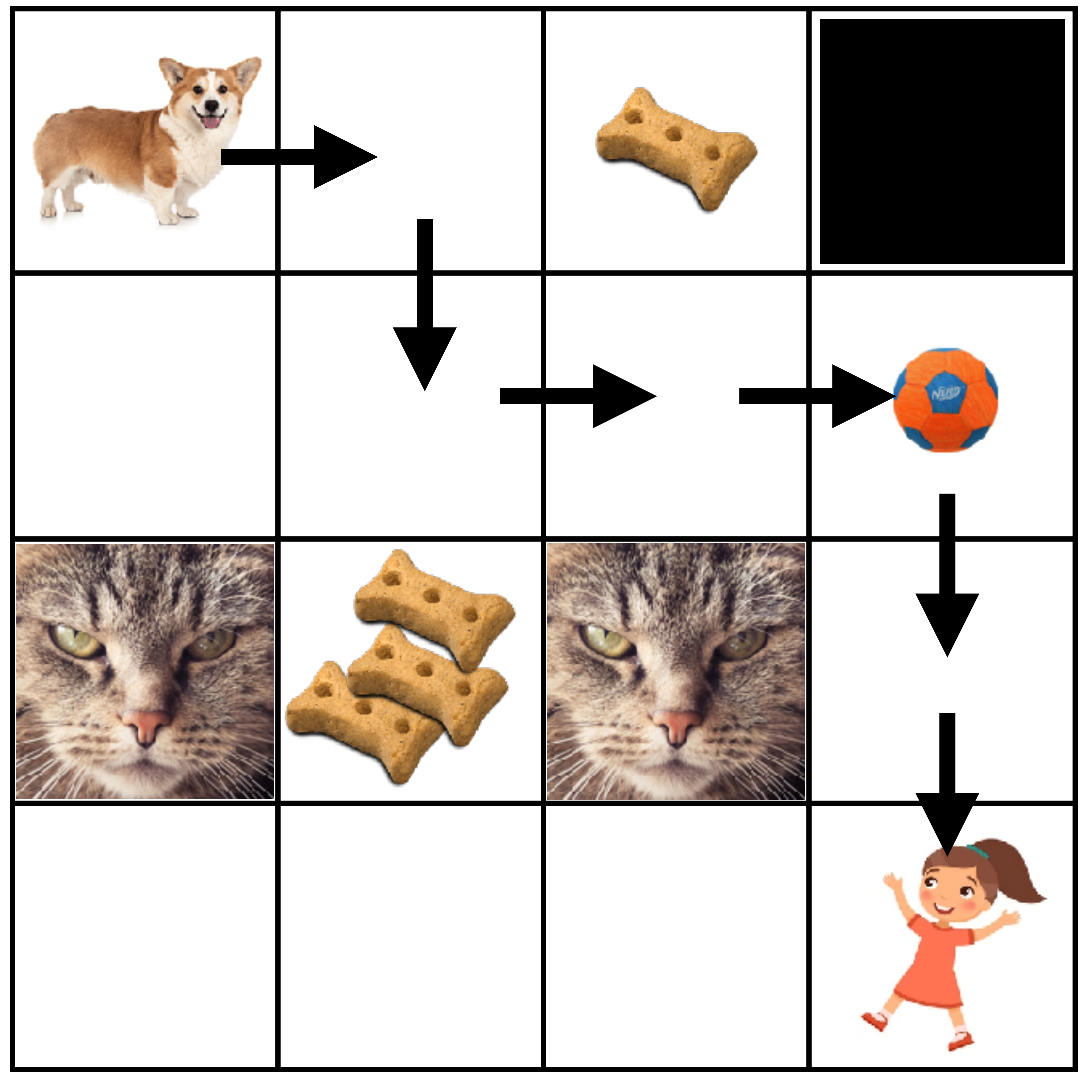
Expectation or Average

- Example: given a die with 6 faces, numbered 1 to 6. If you throw the die and get face 1, you make $1, if you get 2, you make $2, etc. What's your expected return?

$$\sum_i P(i)R(i) = \frac{1}{6}1 + \frac{1}{6}2 + \ldots = \frac{21}{6}$$

Markov Decision Process (MDP)

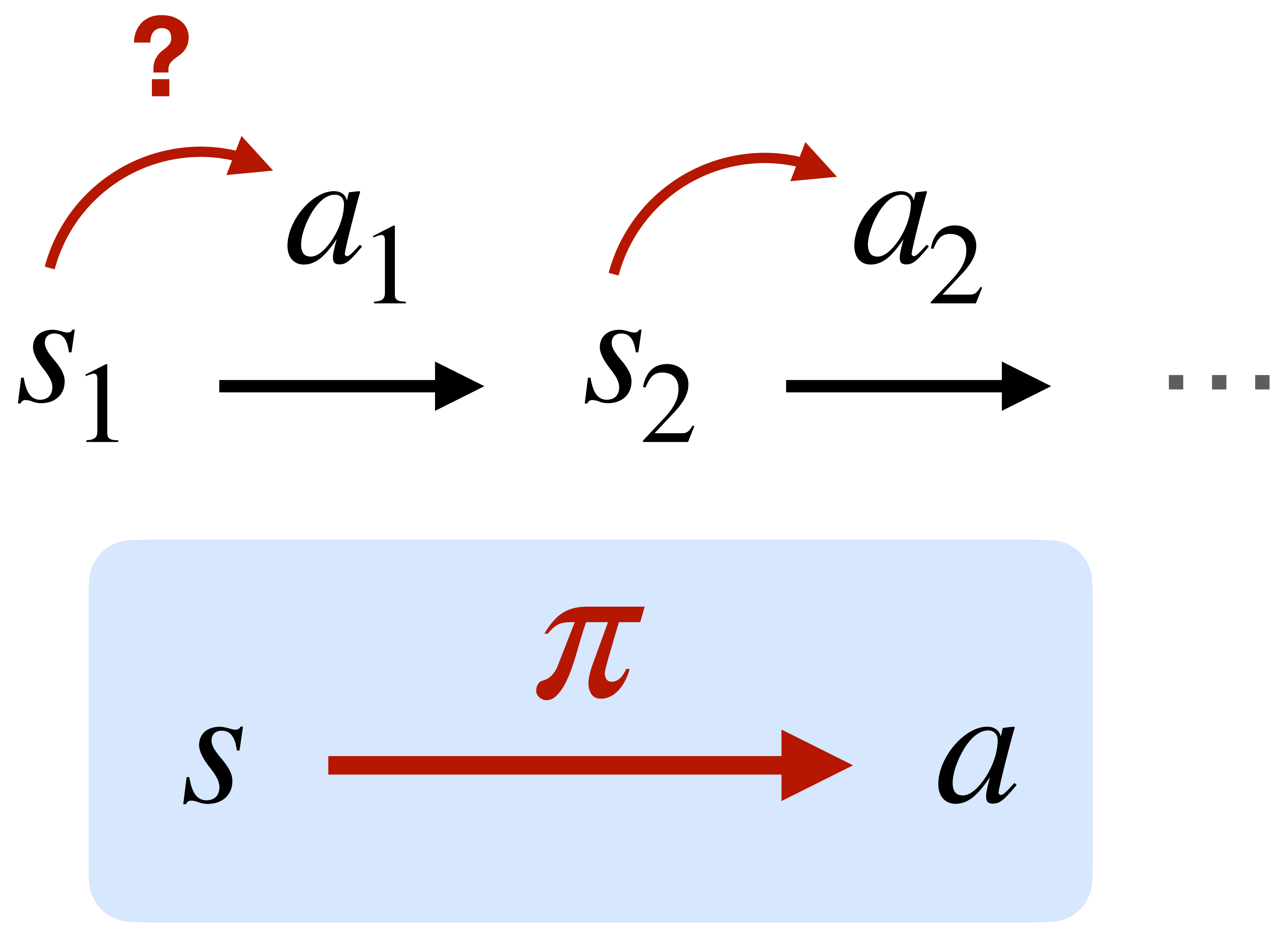




1. Set of **states**: S
2. Set of **actions**: A
3. State **transition probability**: $P(s' \mid s, a) \equiv P_{sa}(s')$, for $s, s' \in S$ and $a \in A$
4. **Reward** function: $R(s, a)$ or $R(s)$ that maps states (and actions) to a real number $\mathbb{R}$
5. **Discount** factor: $\gamma < 1$

How should the agent make decisions?

The agent needs a **policy** for taking action



We are executing some policy π if
whenever we're at s, we take action a according to

$$a = \pi(s)$$

How **good** is a policy π?

We can define a **value function** associated with a policy & Markov Decision Process

$$V^\pi(s) = \mathbb{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \,|\, s_0 = s, \pi\right]$$

$V^\pi(s)$ is the **expected sum of discounted rewards** upon starting in state s, and taking actions according to π

How **good** is a policy π?

We can define a **value function** associated with a policy & Markov Decision Process

$$V^\pi(s) = \mathbb{E}\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots \mid s_0 = s, \pi\right]$$

$$V^\pi(s) = \mathbb{E}\left[R(s_0) + \gamma\left(R(s_1) + \gamma R(s_2) + \ldots\right) \mid s_0 = s, \pi\right]$$

$$V^\pi(s) = R(s) + \gamma \mathbb{E}\left[\left(R(s_1) + \gamma R(s_2) + \ldots\right) \mid s_1 = s, \pi\right]$$

$$V^\pi(s) = R(s) + \gamma \mathbb{E}\left[V^\pi(s_1)\right]$$

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, a) V^\pi(s') \longrightarrow$$

s' is the state after s

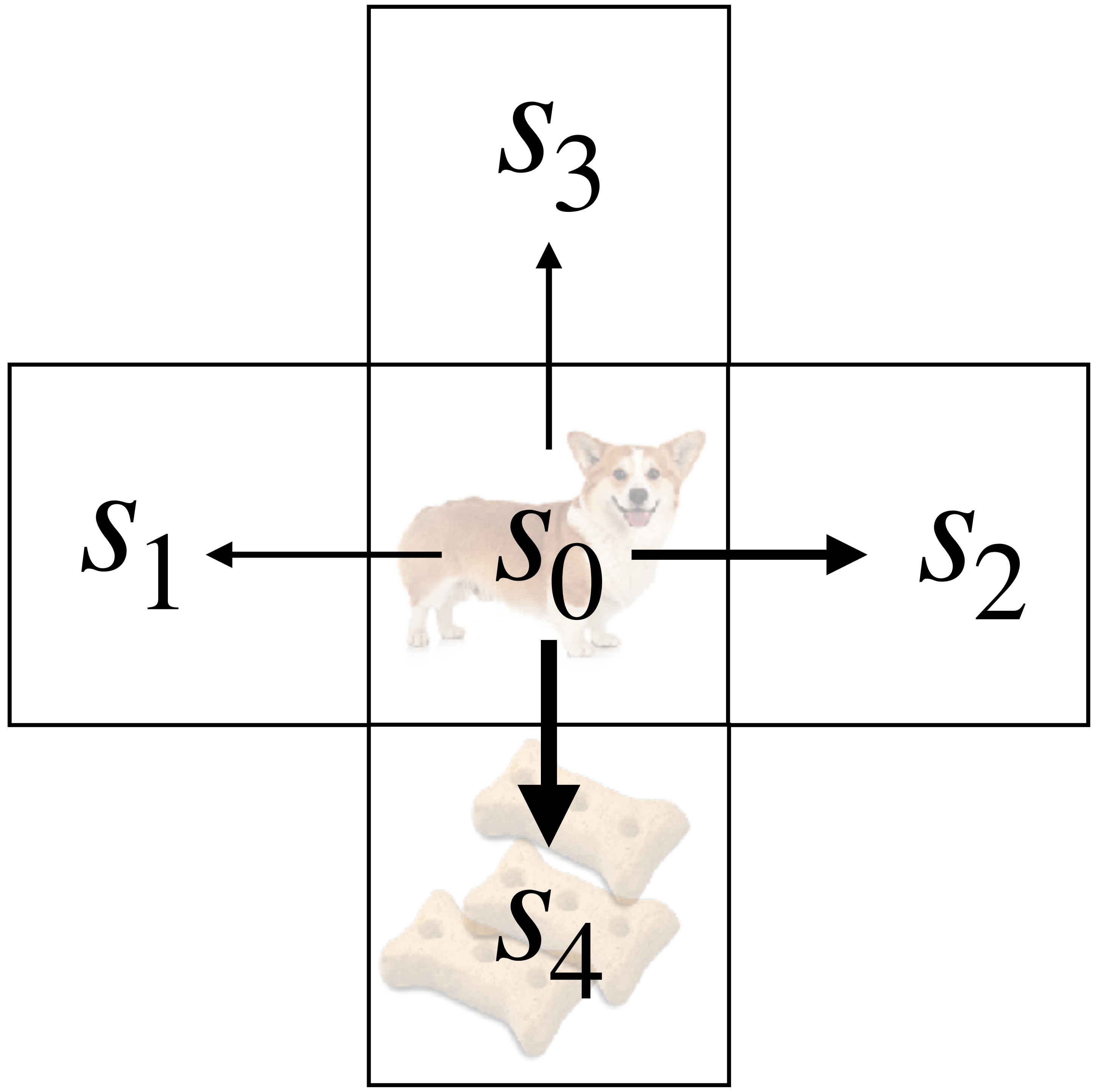
The Bellman Equation

$$V^\pi(s) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, \pi(s)) V^\pi(s')$$

Bellman Equation

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P(s' \mid s, \pi(s)) V^{\pi}(s')$$

Gives a system of $|S|$ linear equations for each unknown $V^{\pi}(s)$



Equation for $V^{\pi}(s_0)$

$$\begin{aligned} V^{\pi}(s_0) = R(s_0) + \gamma \Big[& P(s_1 \mid s_0, \pi(s_0)) V^{\pi}(s_1) \\ & + P(s_2 \mid s_0, \pi(s_0)) V^{\pi}(s_2) \\ & + P(s_3 \mid s_0, \pi(s_0)) V^{\pi}(s_3) \\ & + P(s_4 \mid s_0, \pi(s_0)) V^{\pi}(s_4) \Big] \end{aligned}$$

What's the optimal value function?

In other words, what's the **best expected sum of discounted rewards** if one gets to choose *any* policy π ?

$$V^\star(s) = \max_{\pi} V^\pi(s)$$

What's the optimal policy the agent can take if they start at state s?

$$\pi^\star(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s' \mid s, a) V^\star(s')$$

Solving for the optimal value function

Given the **optimal policy:**

$$\pi^\star(s) = \underset{a \in A}{\text{arg max}} \sum_{s' \in S} P(s' \mid s, a) V^\star(s')$$

We can write its associated **Bellman equation:**

$$V^\star(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s' \mid s, a) V^\star(s')$$

How to find the optimal value function?

Given:

- Reward function $R(s)$
- State transition probability $P(s' | s, a)$

Value iteration algorithm

initialize $V(s) = 0$, for each state s

while not converged:

  for every state s:

$$V(s) = R(s) + \gamma \max_{a \in A} \sum_{s'} P(s' | s, a) V(s')$$

  end

end

Synchronous
compute new $V(s)$ for all s
then overwrite old $V(s)$

Asynchronous
update $V(s)$ one at a time

How to find the optimal value function?

Given:

- Reward function $R(s)$
- State transition probability $P(s' | s, a)$

Value iteration algorithm

initialize $V(s) = 0$, for each state s

while not converged:

  for every state s:

$$V(s) = R(s) + \gamma \max_{a \in A} \sum_{s'} P(s' | s, a) V(s')$$

  end

end

Synchronous
compute new $V(s)$ for all s
then overwrite old $V(s)$

Asynchronous
update $V(s)$ one at a time

Having found $V^\star$,
compute optimal policy

$$\pi^\star(s) = \text{arg max}_{a \in A} \sum_{s' \in S} P(s' | s, a) V^\star(s')$$

How to find the optimal policy?

Given:

- Reward function $R(s)$
- State transition probability $P(s' | s, a)$

$$\pi^\star(s) = \arg\max_{a \in A} \sum_{s' \in S} P(s' | s, a) V^\star(s')$$

Policy iteration algorithm

initialize π randomly

while not converged:

  Let $V = V^\pi$ ← (by solving linear system)

  for every state s:

$$\pi(s) = \arg\max_{a \in A} \sum_{s'} P(s' | s, a) V(s')$$

  end

end

Policy vs. Value Iteration

Policy iteration algorithm

initialize π randomly
while not converged:
  Let $V = V^\pi$
  for every state s:
    $\pi(s) = \arg\max_{a \in A} \sum_{s'} P(s' | s, a) V(s')$
  end
end

Value iteration algorithm

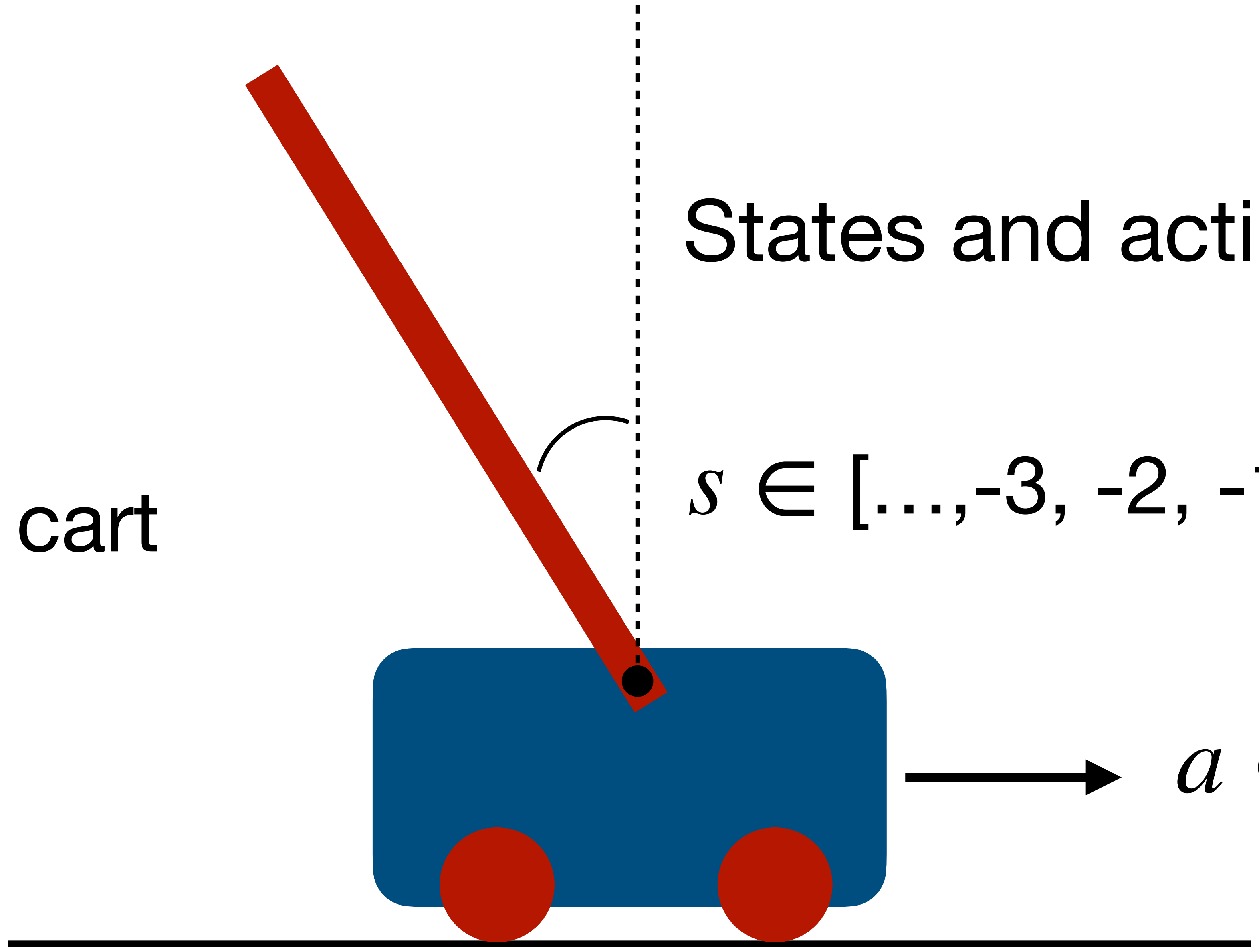
initialize $V(s) = 0$, for each state s
while not converged:
  for every state s:
    $V(s) = R(s) + \gamma \max_{a \in A} \sum_{s'} P(s' | s, a) V(s')$
  end
end

- There is no agreement which is better
- For small MDPs, policy iteration is very fast.
- But the linear solve step is slow for large state spaces, so value iteration is used more often

What if we don't know $P(s' \mid s, a)$ and $R(s)$?

We are not given state transitions and rewards explicitly, but are given data

- State s is the angle of the pole
- Actions a is the velocity of the cart



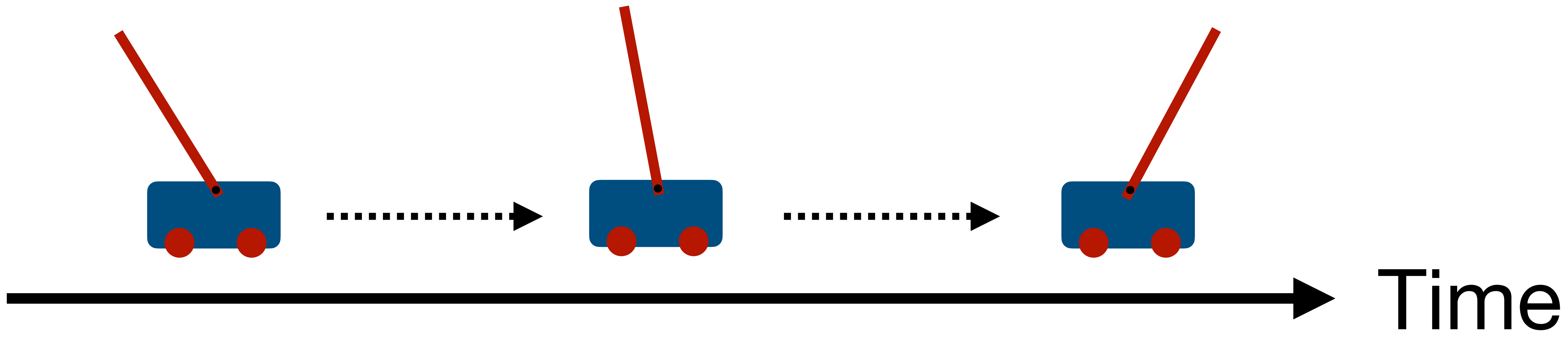
States and actions are discretized

$s \in [\ldots,-3, -2, -1, 0, 1, 2, 3,\ldots]$ degrees

$a \in [\ldots,-3, -2, -1, 0, 1, 2, 3,\ldots]$ m/s

What if we don't know $P(s' \mid s, a)$ and $R(s)$?

We are given a number of trials



Trial 1 $\quad s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} s_3^{(1)} \xrightarrow{a_3^{(1)}} \cdots$

Trial 2 $\quad s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} s_3^{(2)} \xrightarrow{a_3^{(2)}} \cdots$

$\vdots \quad \vdots \quad \vdots \quad \vdots$

$s_i^{(j)}$: state at time i of trial j

$a_i^{(j)}$: action taken from that state

Learning $P(s' \mid s, a)$

$$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} s_3^{(1)} \xrightarrow{a_3^{(1)}} \cdots$$

$$s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} s_3^{(2)} \xrightarrow{a_3^{(2)}} \cdots$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$P(s' \mid s, a) = \frac{\text{\# of } s \xrightarrow{a} s'}{\text{\# of } s \xrightarrow[a]{}}$$

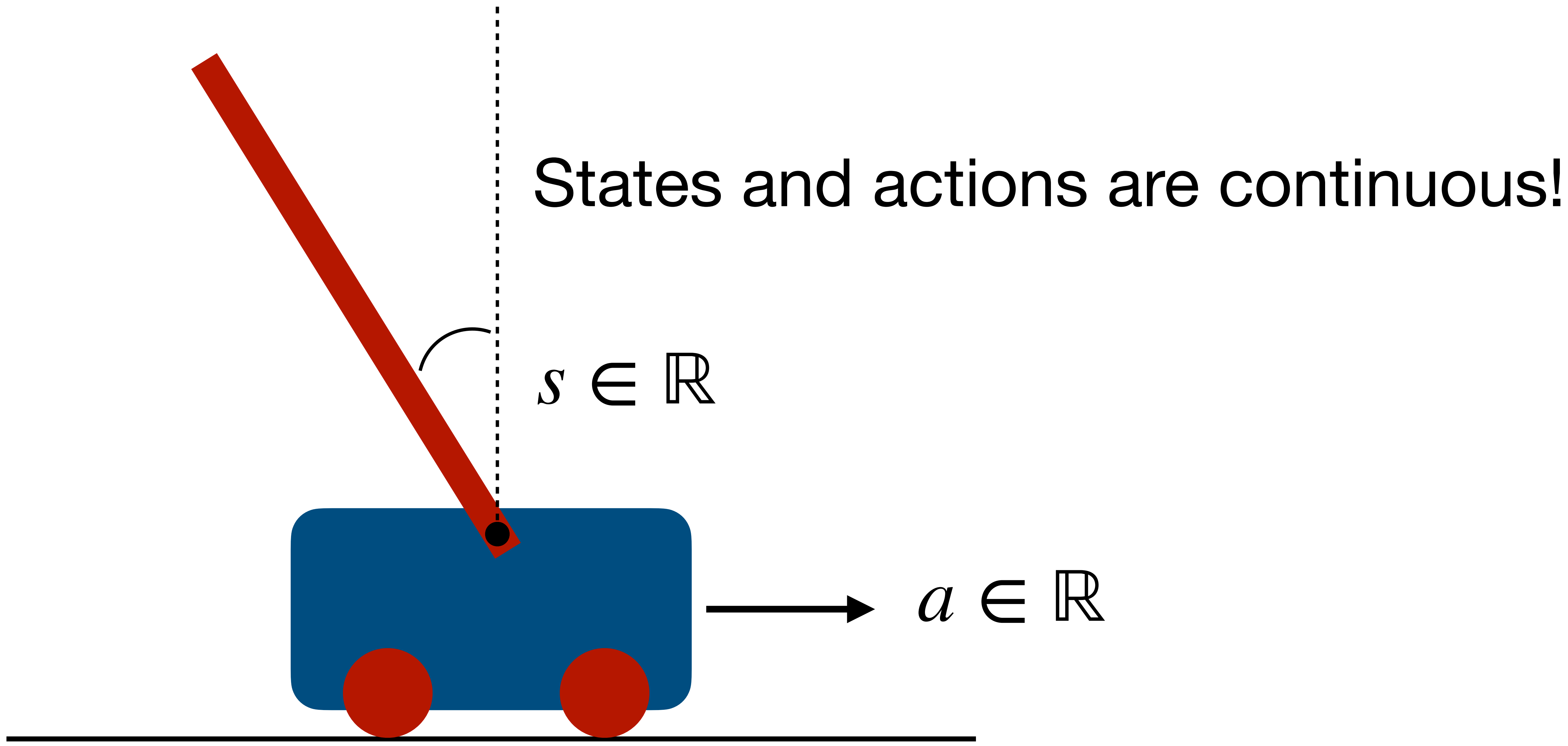
$$R(s) = \frac{\text{sum of } R \text{ in } s}{\text{\# of times in } s}$$

Find the optimal policy with unknown P and R

initialize π randomly
for trials:
    for trials:
        Execute π in MDP
    end
    update estimates of $P(s' \mid s, a)$ and $R(s)$
    apply **value iteration** to find new $V(s)$
    update π
end

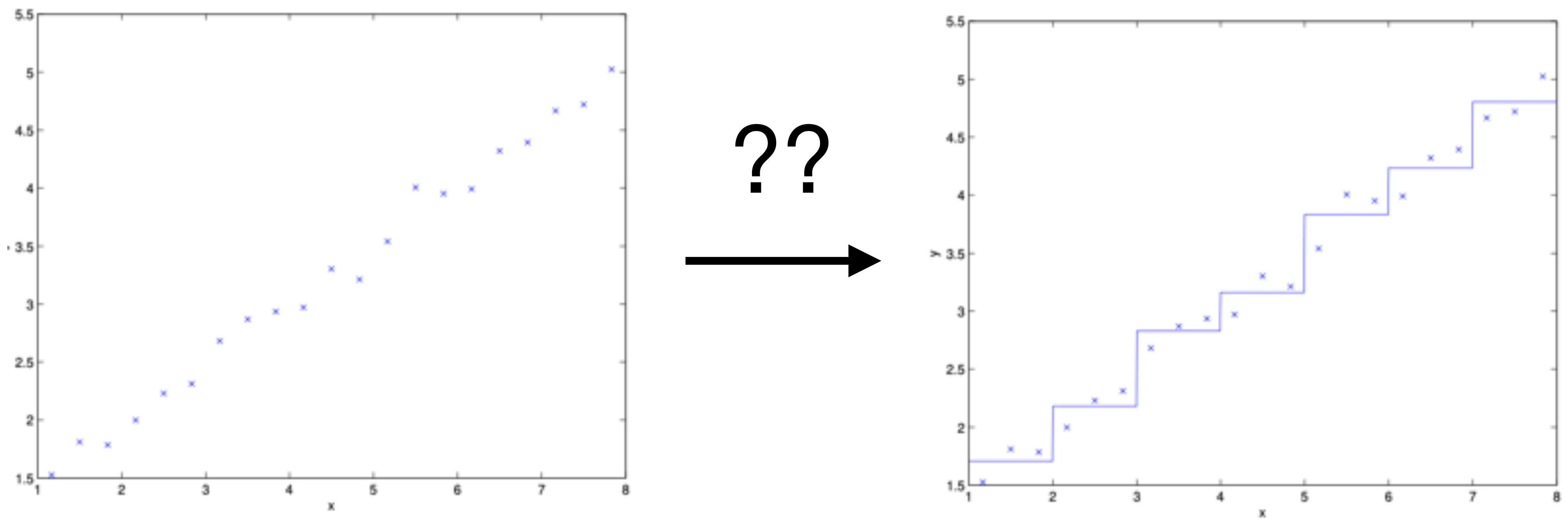
Continuous Markov Decision Processes (MDP)

- State s is the angle of the pole
- Actions a is the velocity of the cart

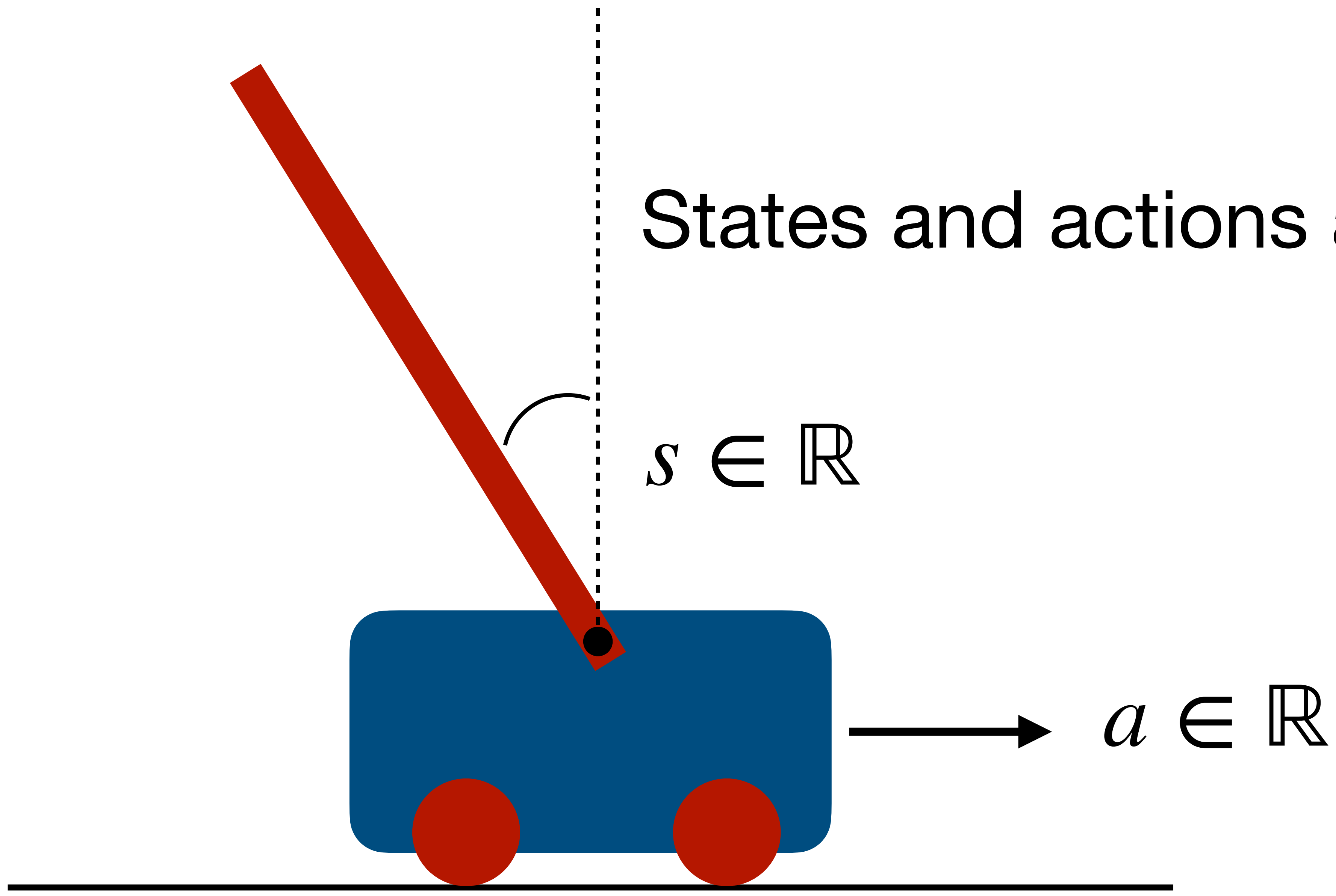


Downsides of discretization:

- Step-wise fit to a continuous problem
- Curse of dimensionality



Continuous Markov Decision Processes (MDP)



States and actions are continuous!

- State s is the angle of the pole
- Actions a is the velocity of the cart

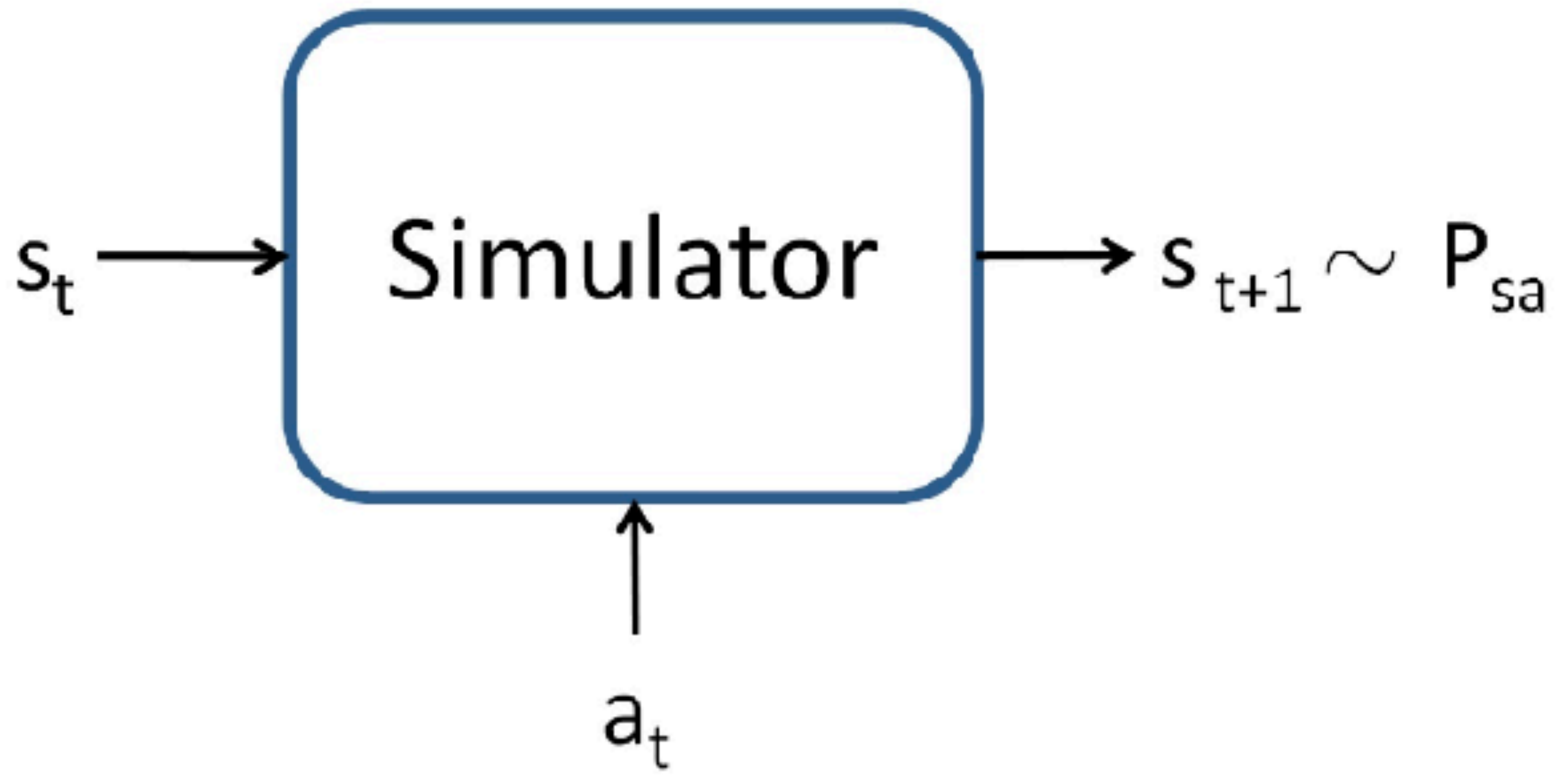
Downsides of discretization:

- Step-wise fit to a continuous problem
- Curse of dimensionality

$S \in \mathbb{R}^d \longrightarrow k^d$ States

Discretize each dim.
into k values

Value Function Approximation



Given a physical model $$\frac{ds}{dt} = f(s)$$

Value Function Approximation

Trial 1 $$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)}} s_3^{(1)} \xrightarrow{a_3^{(1)}} \cdots$$

Trial 2 $$s_0^{(2)} \xrightarrow{a_0^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)}} s_3^{(2)} \xrightarrow{a_3^{(2)}} \cdots$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

Learn a state-space model

$$s_{t+1} = As_t + Ba_t$$

With A and B as fitting parameters

Fitting Value Iteration

$$
\begin{aligned}
V(s) &:= R(s) + \gamma \max_a \int_{s'} P_{sa}(s') V(s') ds' \\
&= R(s) + \gamma \max_a \mathrm{E}_{s' \sim P_{sa}} [V(s')]
\end{aligned}
$$

See page 188 for more