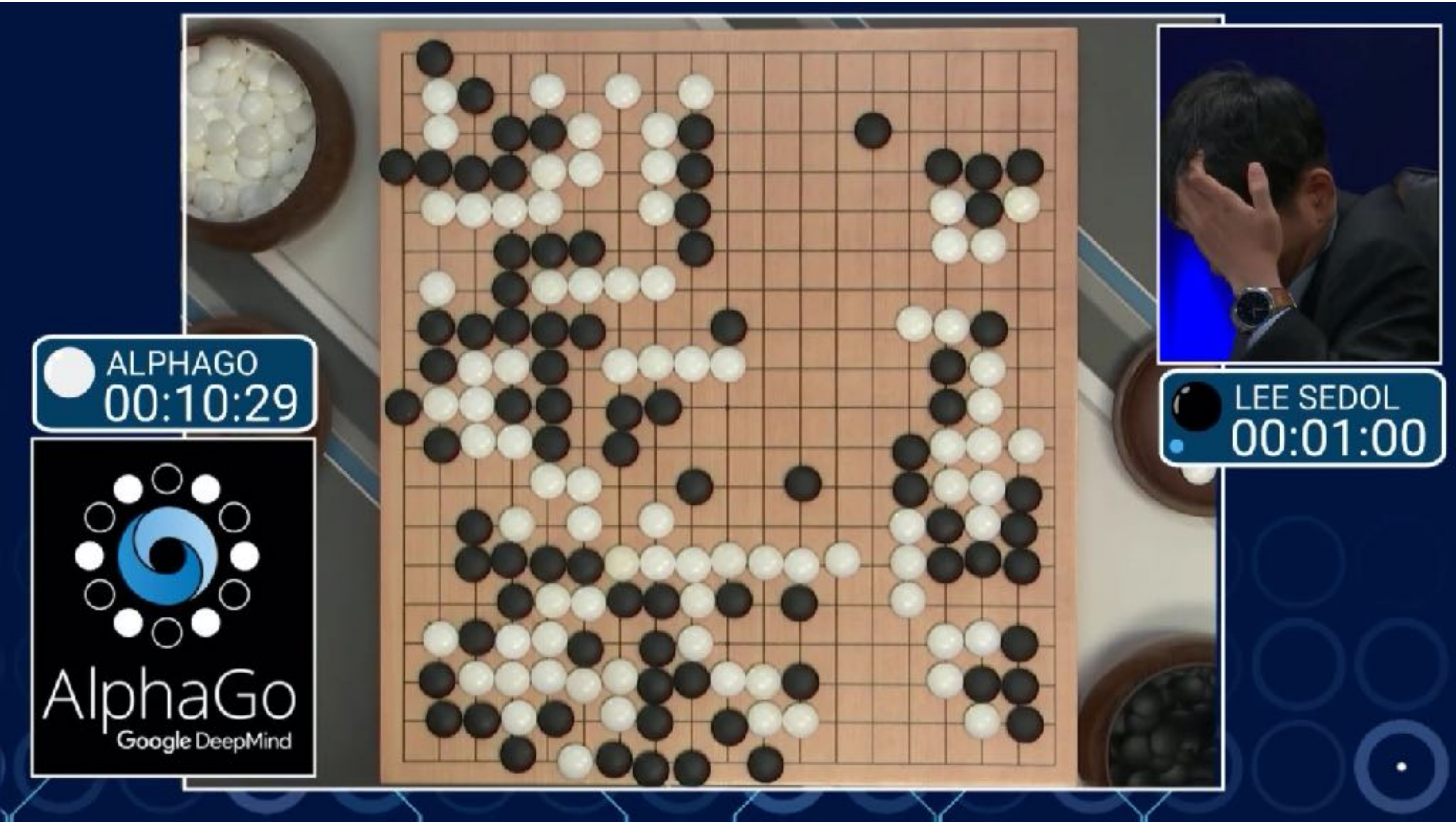
Lecture 21: Reinforcement Learning 1

Artificial Intelligence

Julian Togelius



ALPHAGO
00:10:29
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind

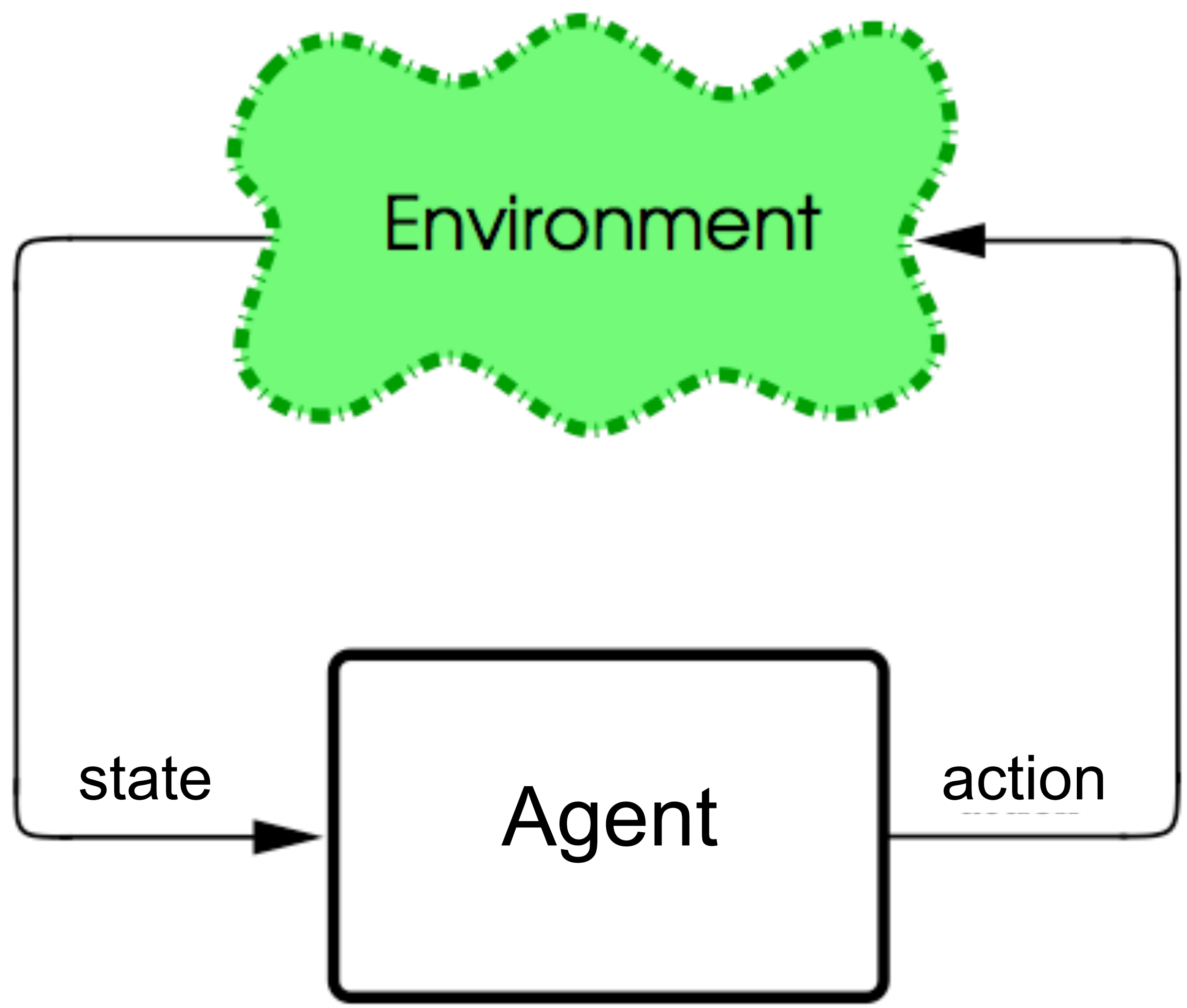

ima...
217 4 1


PHILADELPHIA
ZOO

Limits of tree search

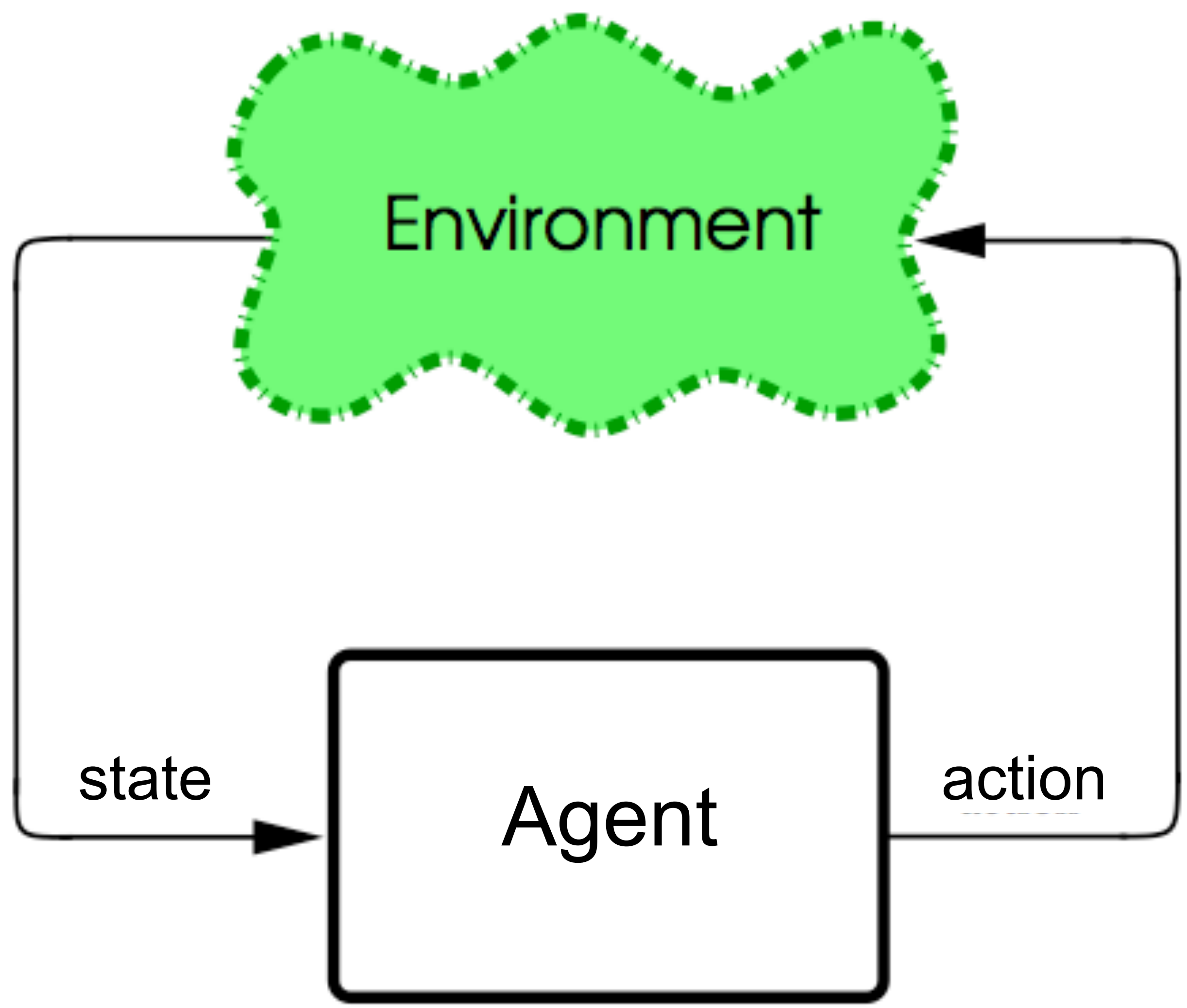
- Need a proper forward model
- Trees might be huge
- State evaluation might be hard
- Each action selection takes a lot of time

Sequential Decision Tasks



Agent sees state of the environment at each time-step and must select best action to achieve a goal. How can we learn what the best actions are?

Sequential Decision Tasks



s_t : state of the environment at time *t*

a_t : action taken by agent at time *t* after seeing s_t

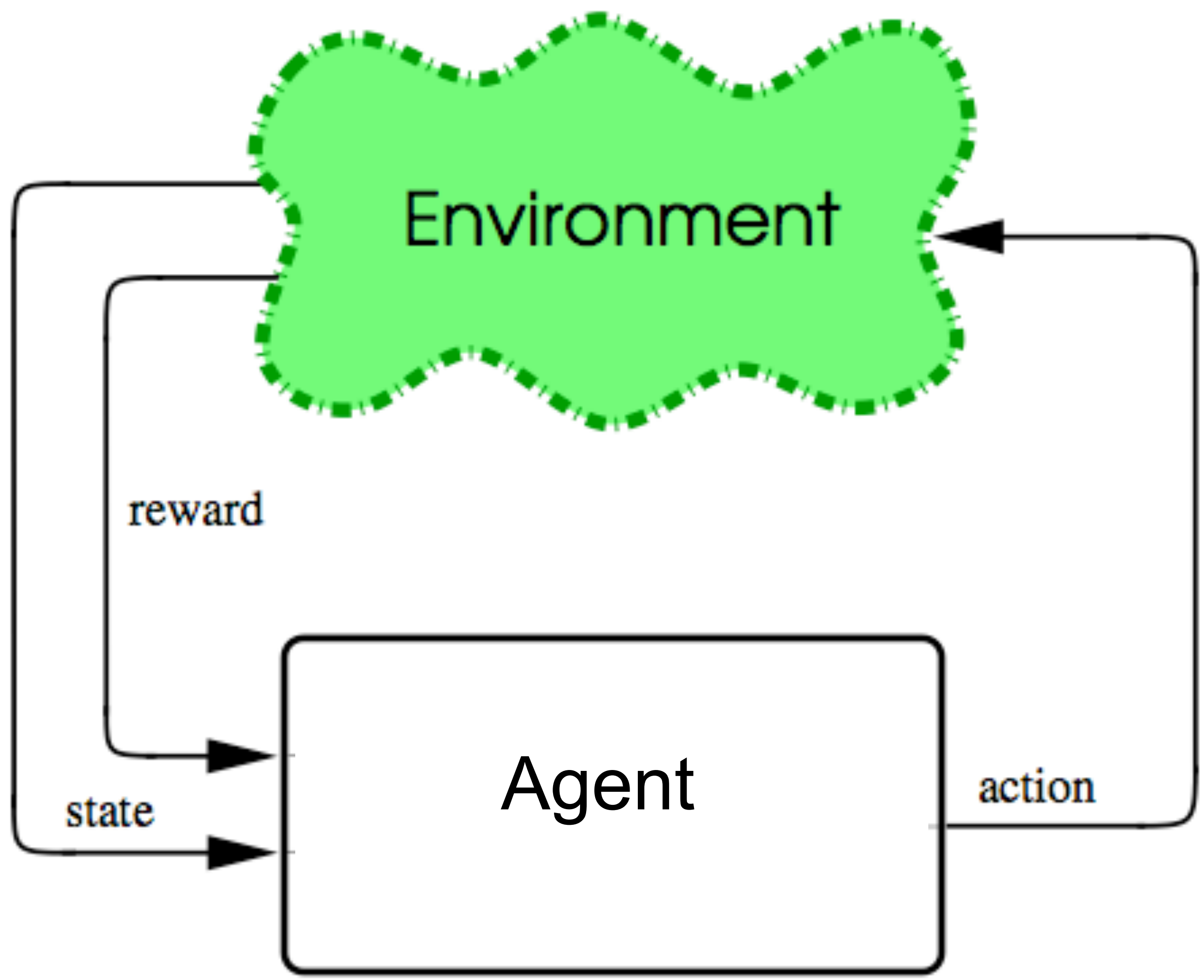
Decision sequence:

$$s_t \xrightarrow{a_t} s_{t+1} \xrightarrow{a_{t+1}} s_{t+2} \xrightarrow{a_{t+2}} s_{t+3} \xrightarrow{a_{t+3}} \ldots$$

Sequential Decision Tasks

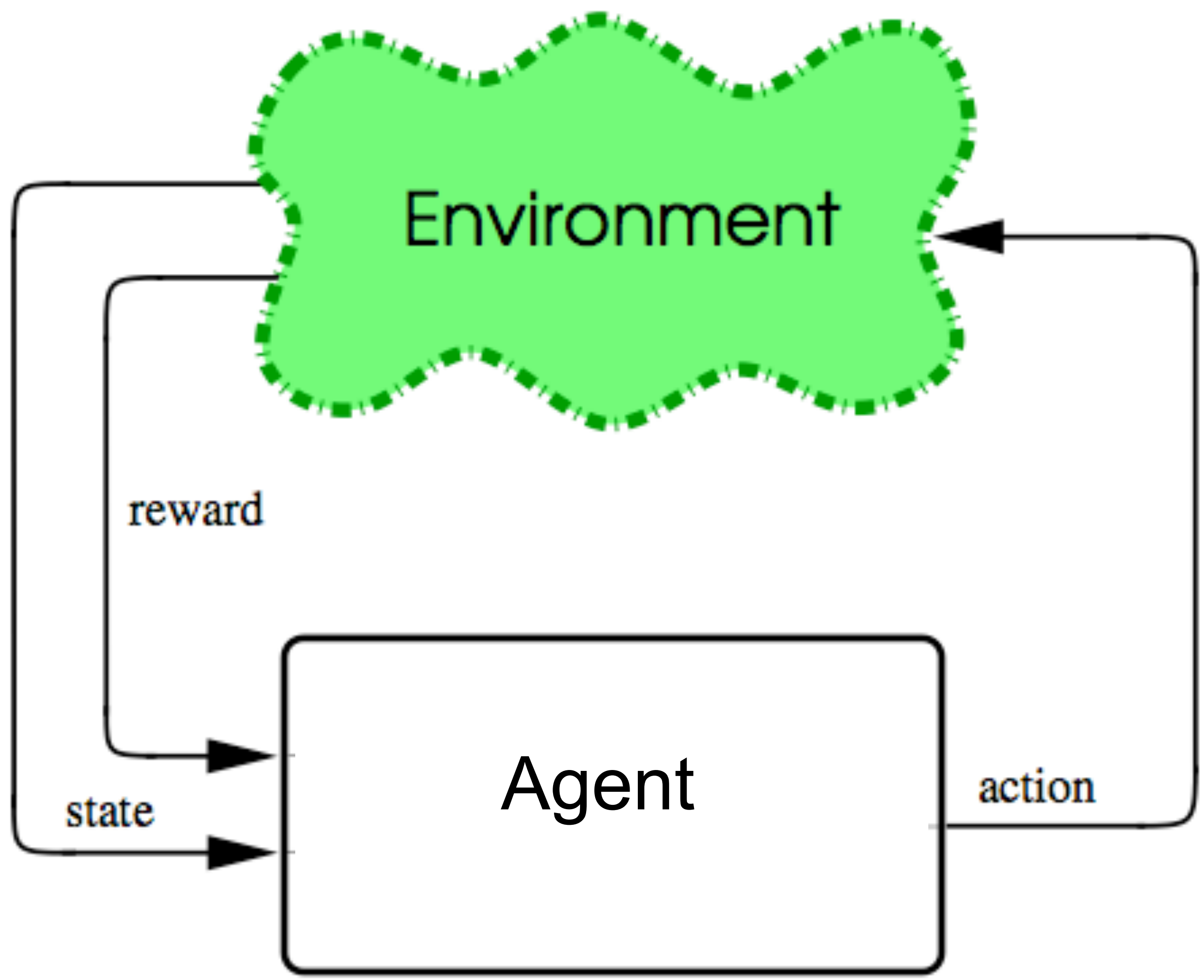
- Autonomous robotics
- Controlling chemical processes
- Network routing
- Game playing
- Stock trading

Reinforcement Learning Problem



Agent receives a *reinforcement* (“good” or “bad” behavior)

Reinforcement Learning Problem

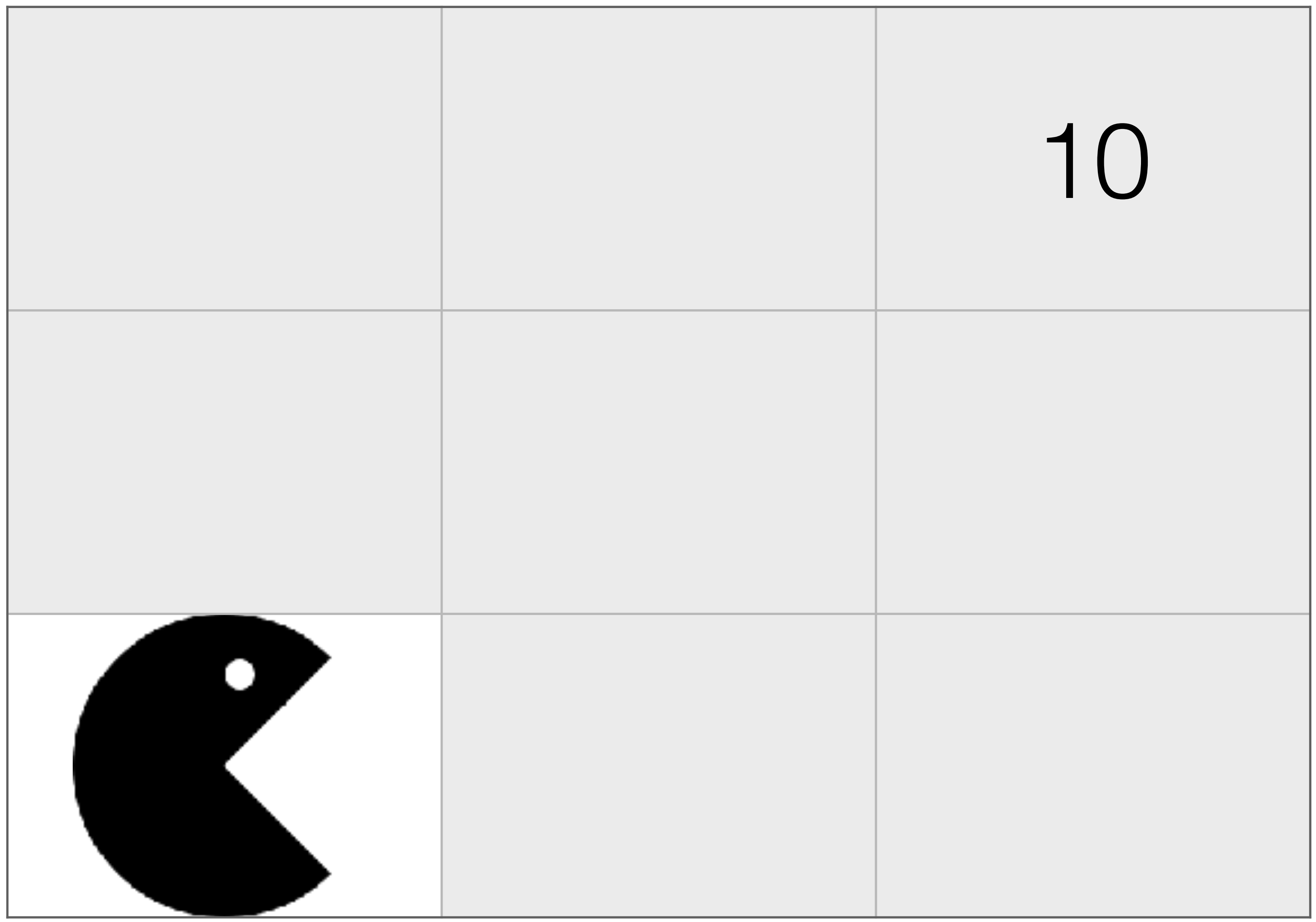


$$s_t \xrightarrow{a_t} \overset{r_{t+1}}{\overset{\uparrow}{s_{t+1}}} \xrightarrow{a_{t+1}} \overset{r_{t+2}}{\overset{\uparrow}{s_{t+2}}} \xrightarrow{a_{t+2}} \overset{r_{t+3}}{\overset{\uparrow}{s_{t+3}}} \xrightarrow{a_{t+3}} \ldots$$

Pac-Man

- One example:
  - 1 if you eat a pill
  - -10 if you get caught by a ghost
  - 2 if you eat a power pill or eat a ghost
  - 0 otherwise
- Another example:
  - -1 every time step
  - 1000000 if you win the level

Pac-man Rewards



Actions: a & b

Model

a = down b = right	a = down b = right	10
a = up b = right	a = down b = left	a = down b = left
a = up b = right	a = up b = right	a = up b = left

Actions: a & b

State Utilities

8.1	9	10
7.29	6.56	5.9
6.56	5.9	5.3

Discount (γ): 0.9

Pac-man Outcome

b	b	10
a		
a		

Actions: a & b

Bellman Equation

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} p(s' | s, \pi(s)) V^{\pi}(s')$$

- $R(s)$ - The reward for the current state
- γ - How much to discount future rewards ($0 \leq \gamma \leq 1$)
- $P(s' | s, a)$ - model of future states
- $\pi(s)$ - action taken given state s
- Sum up the utilities of all future states

Agent Policy

- The policy implements the agents behavior
- In general, the policy could be stochastic:

$$\pi(s,a) = \Pr\{a_t = a \mid s_t = s\}$$

policy gives the probability of taking action *a* in state *s*

- Often the policy is deterministic and we can write:

$$\pi(s) \rightarrow a$$

for each state the policy says which action to use

Markov Decision Processes

a finite set of states: $s \in S$ also known as the *state-space*

a finite set of actions: $a \in A$ also known as the *action-space*

state transition probabilities: $P_{ss'}^{a} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$

reward function: $R_{ss'}^{a} = \mathrm{E}\{r_{t+1} \mid s_t = s, a_t = a\}$

a policy: $\pi(s,a) = \Pr\{a_t = a \mid s_t = s\}$

…and the Markov property must hold.

Markov Decision Processes

a finite set of states: $s \in S$

a finite set of actions: $a \in A$

model

state transition probabilities: $P^a_{ss'} = \Pr\{s_{t+1} = s' \mid s_t = s, a_t = a\}$

reward function: $R^a_{ss'} = E\{r_{t+1} \mid s_t = s, a_t = a\}$

a policy: $\pi(s,a) = \Pr\{a_t = a \mid s_t = s\}$

…and the Markov property must hold.

The Markov Property

$$P_{ss'}^{a} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, r_{t-1}, \ldots, s_0, a_0, r_0\}$$

$$\equiv$$

$$P_{ss'}^{a} = \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

This just means that the probability of the next state and reward only depend on the immediately preceding state and action!

It doesn't matter what the happened before that!

Example Transition Matrix

Each action will have a transition matrix $P_{ss'}^{a}$

To:

From:

	s_1	s_2	s_3	s_4
s_1	0	0.1	0.3	0.6
s_2	0.5	0	0	0.5
s_3	0	0.4	0	0.6
s_4	0	0.2	0.3	0.5

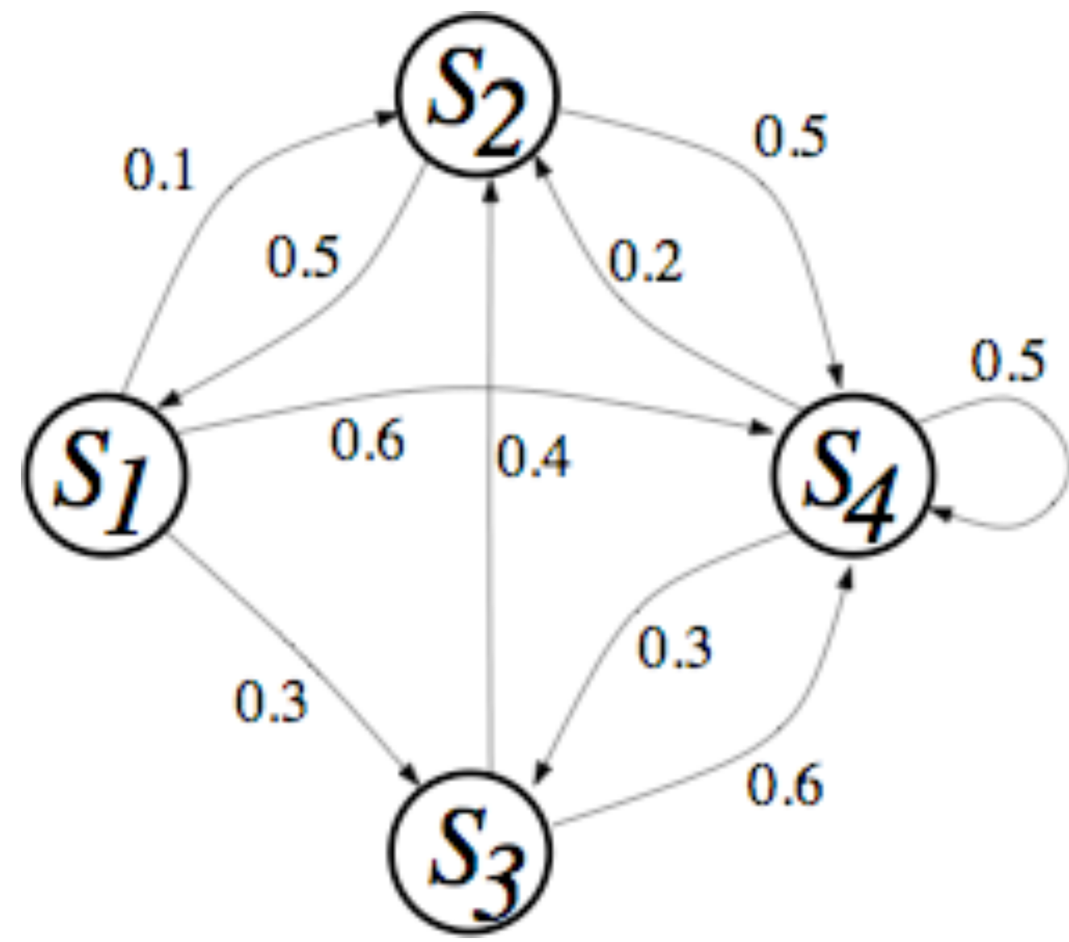
- example with four states
- each entry gives the probability of going from one state to another *if* the action is taken
- each row sums to 1.0

e.g. $P_{s_3 s_4}^{a} = 0.6$, there is a 60% chance of going to state 4 when action *a* is taken in state 3

State Transitions (cont.)

	s_1	s_2	s_3	s_4
s_1	0	0.1	0.3	0.6
s_2	0.5	0	0	0.5
s_3	0	0.4	0	0.6
s_4	0	0.2	0.3	0.5

=



transition graph

All edges leaving state add to 1.0

More About R

$$R_t = \sum_{k=0}^{T} \gamma^k \overbrace{r_{t+k+1}}^{\text{reward}}, \quad 0 \le \gamma \le 1$$

- R is also known as the return. It is how much reward the agent will receive from time t into the future.

- If γ is close to 0, then the agent cares more about selecting actions that maximize immediate reward: shortsighted

- if γ is close to 1, then the agent takes future rewards into account more strongly: farsighted

Policy

The goal is to learn a *policy* that maximizes the reward r over the long term:

$$R_t = \sum_{k=0}^{T} \gamma^k \overbrace{r_{t+k+1}}^{\text{reward}}, \quad 0 \le \gamma \le 1$$

where γ is the discount rate. γ=1 means the all rewards received matter equally. γ<1 means rewards further in the future are less important.

Why Are Reinforcement Learning Problems Hard to Solve?

- Have to discover behavior from scratch
- Only have scalar reinforcement to guide learning
- Reinforcement may be infrequent
- Credit assignment problem: How much credit should each action in the sequence of actions get for the outcome

Solving Reinforcement Learning Problems

- “Classical” approaches:
  - based on approximate dynamic programming (this and next lecture)
  - based on policy gradients (not covered here)
- Evolutionary approaches:
  - based on evolution, or other stochastic optimization methods (early part of the course)