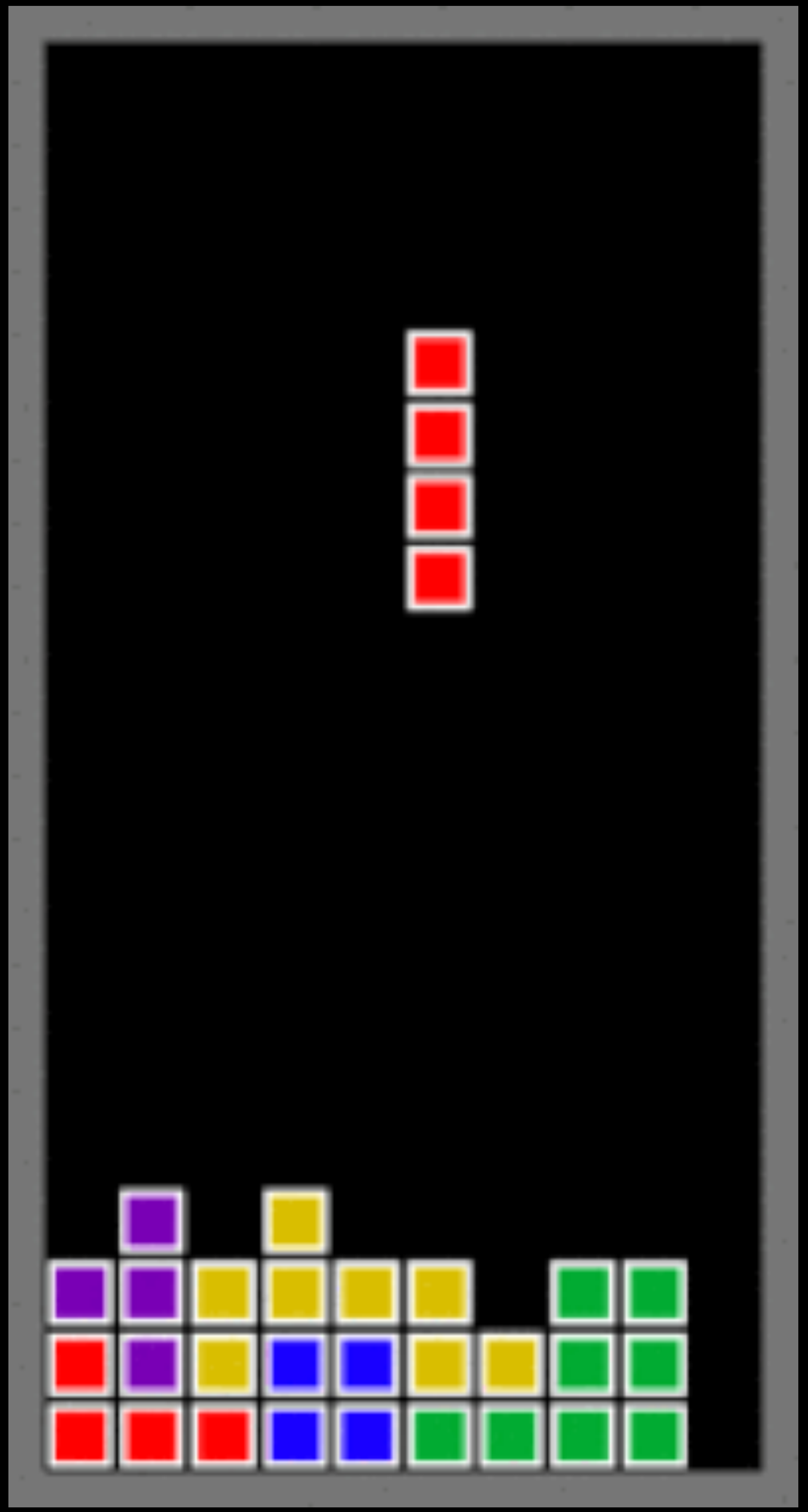
Robots as
Markov Decision Problems

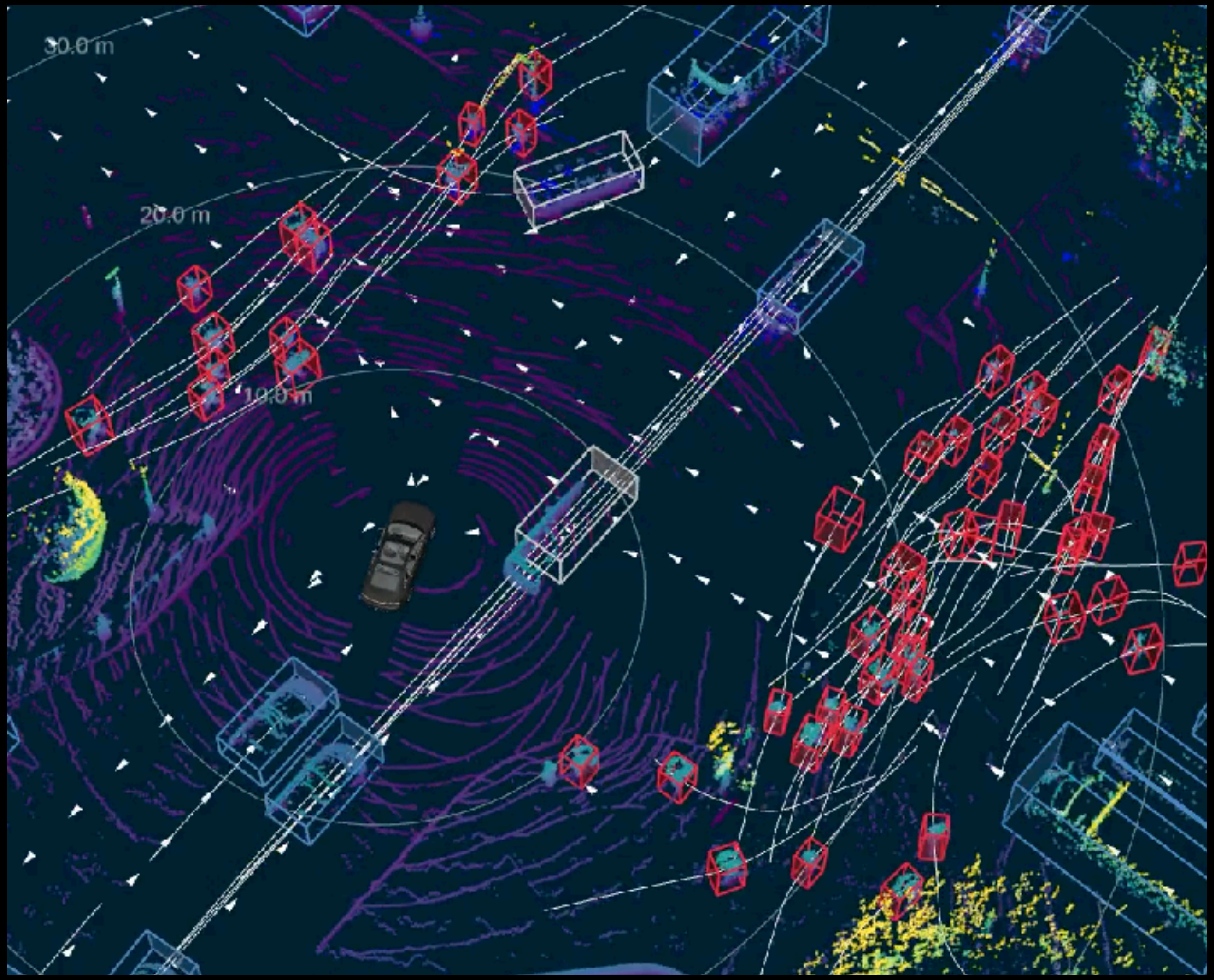
Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

Sequential Decision Making



Tetris

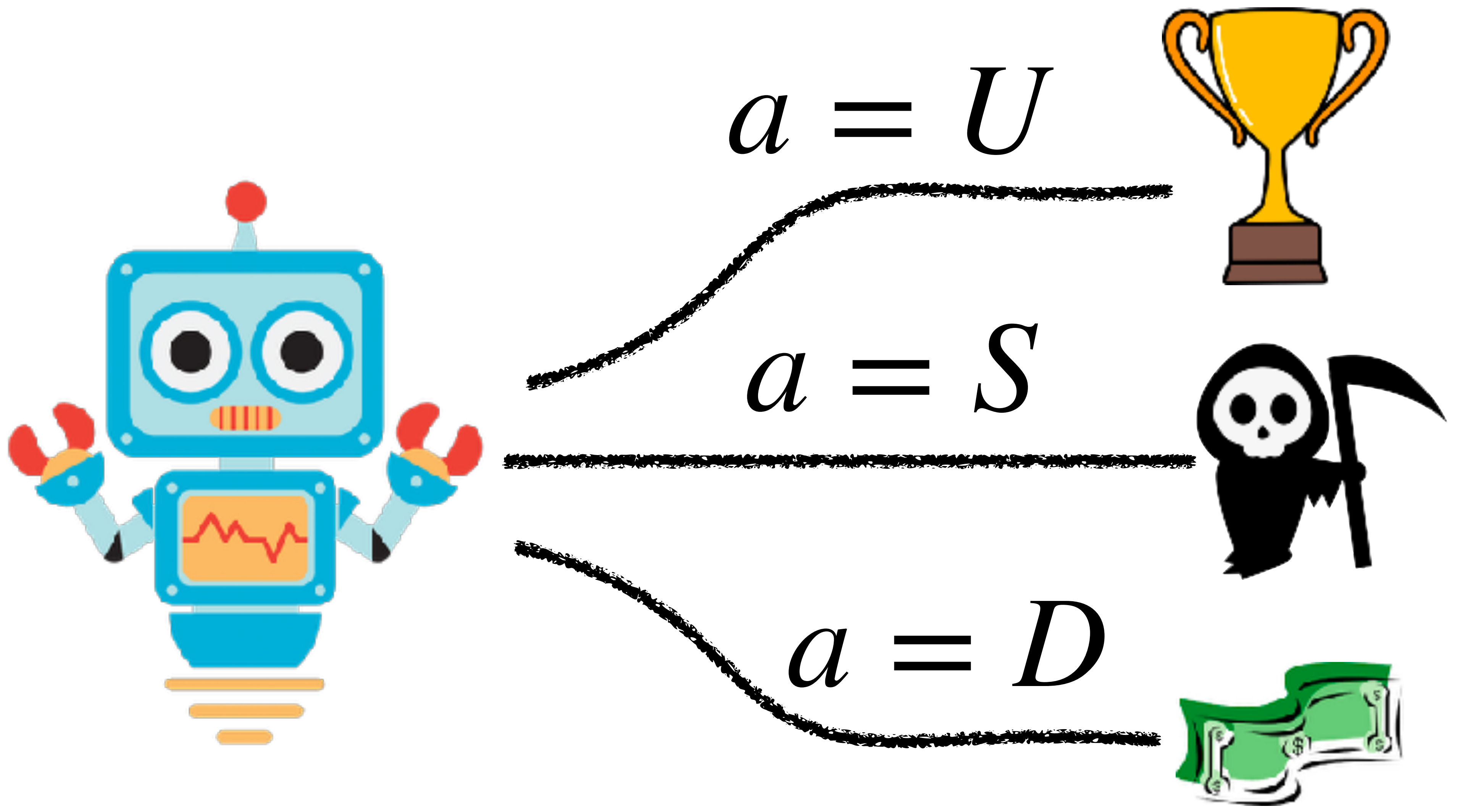


Self-driving

Robot Baristas

What makes *sequential* decision making hard?

An Easy Example: *Non-sequential* decision making

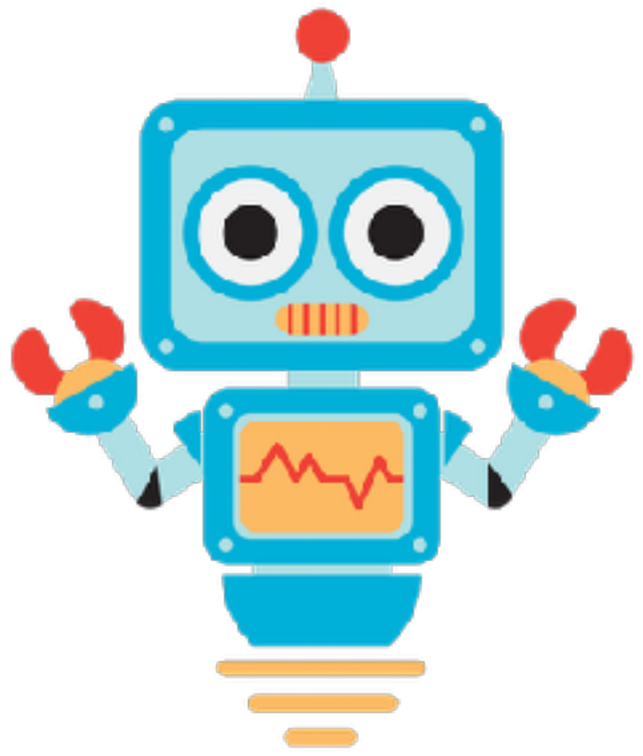


	Reward
$a = U$	100
$a = S$	-1000
$a = D$	1

Goal: Pick the action that maximizes reward $\arg\max_a R(a)$

What is the complexity of this optimization problem?

A Hard Example: *Sequential* decision making



A sequence of actions
for T time steps
$a_1, a_2, a_3, a_4, \ldots a_T$

Black Box

Reward
-100

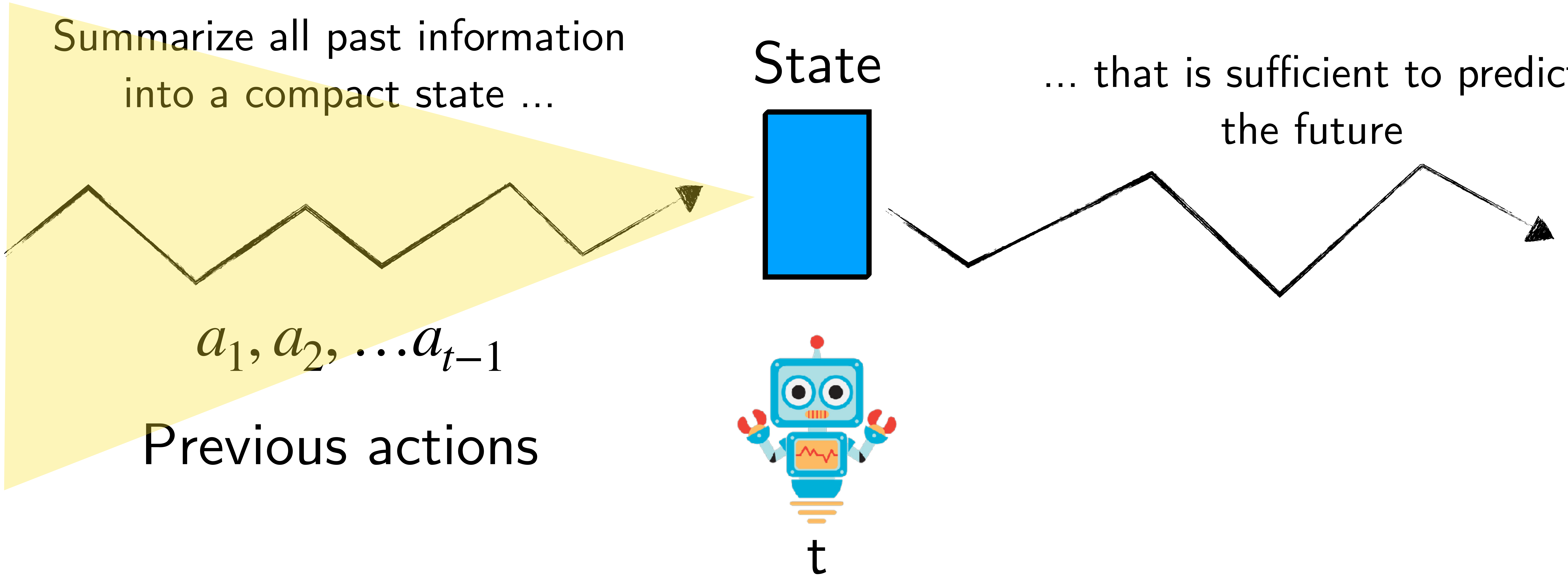
Goal: Pick the sequence of actions that maximizes reward

$$\arg \max_{a_1, a_2, \ldots, a_T} R(a_1, a_2, \ldots, a_T)$$

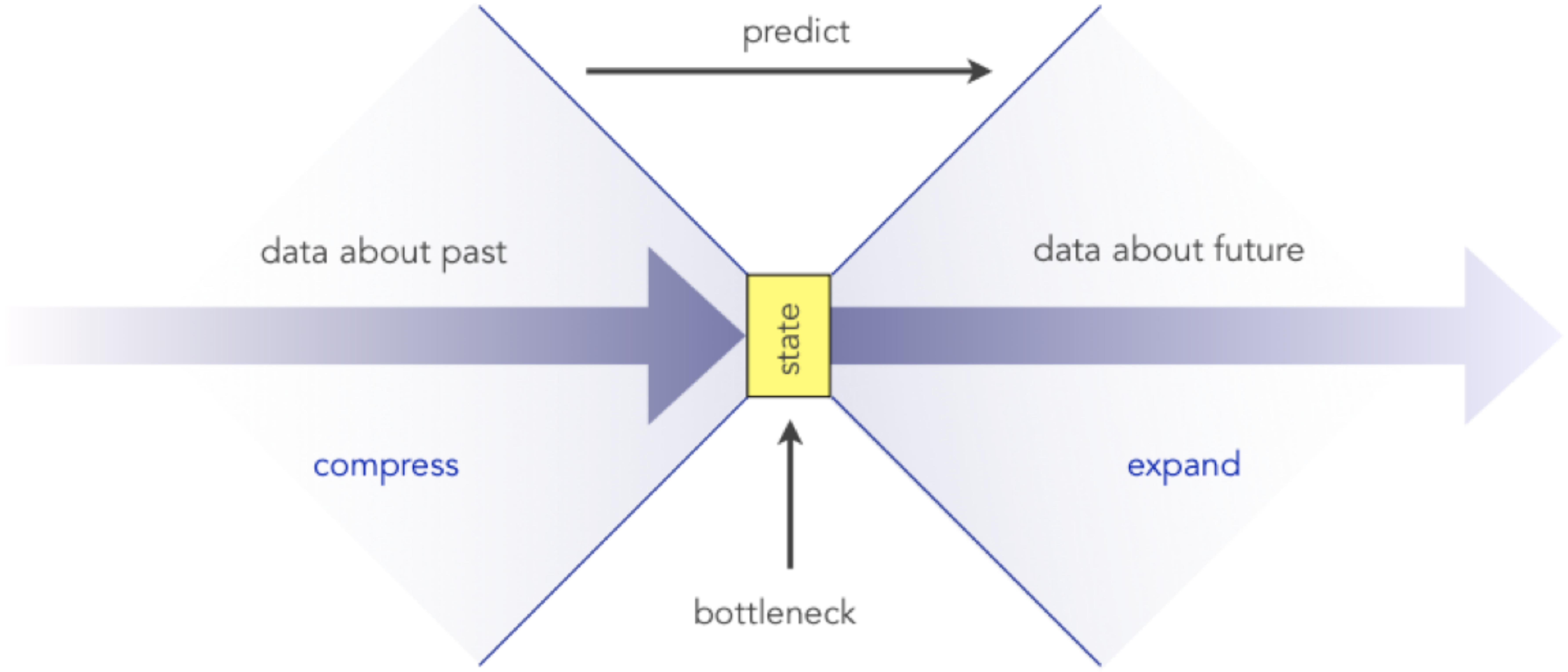
What is the complexity of this optimization problem?

What assumption makes the optimization problem tractable?

The Markov Assumption



The Markov Assumption

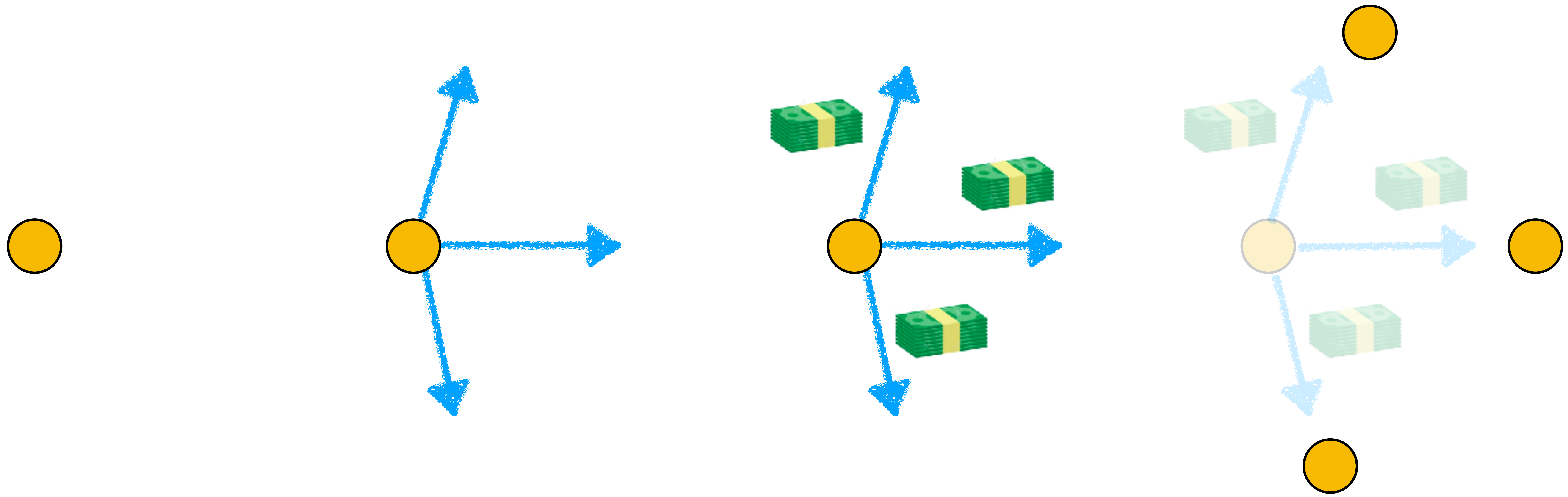


State: statistic of history sufficient to predict the future

Markov Decision Process

A mathematical framework for modeling sequential decision making

$$< S \, , A \, , C \, , \mathcal{T} >$$

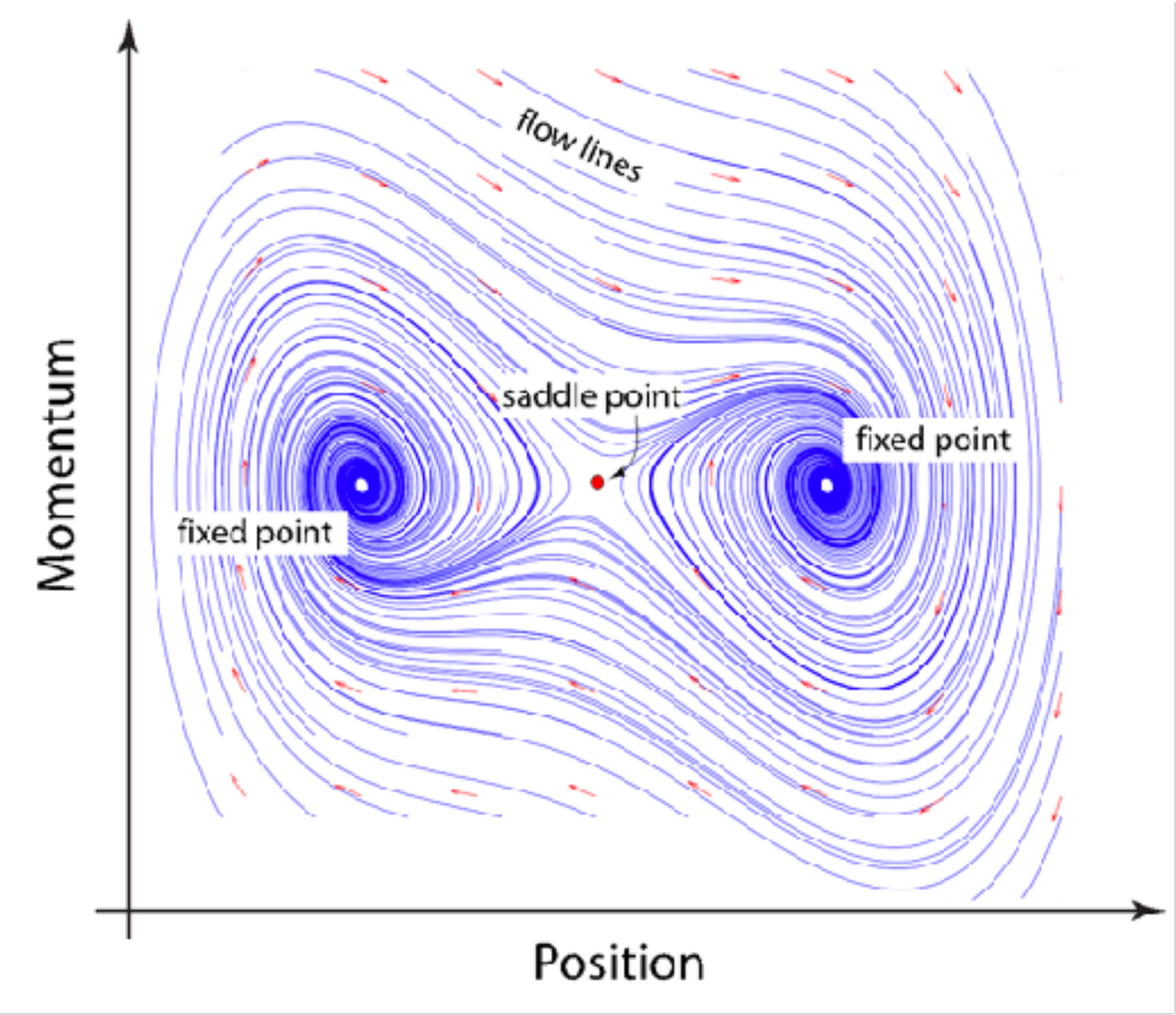


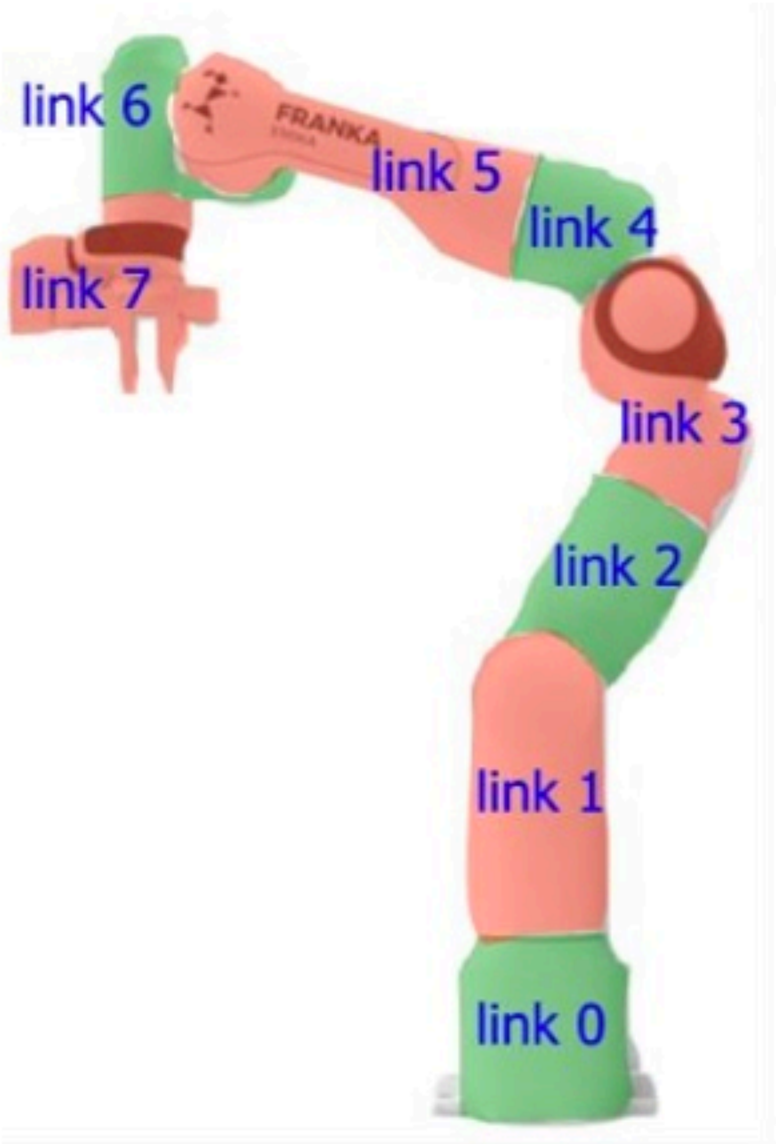
State

$< S, A, C, \mathcal{T} >$

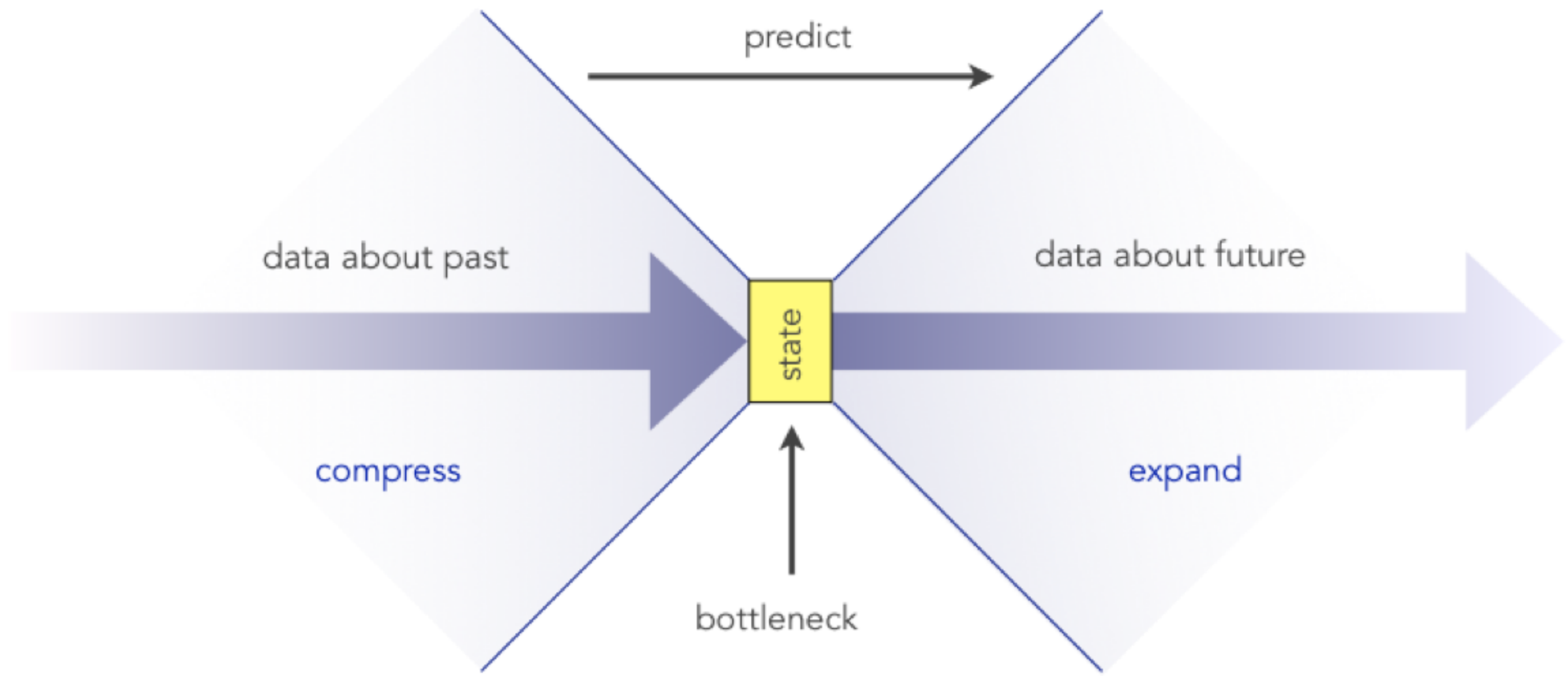
Sufficient statistic of the system to predict future disregarding the past

- $s \in S$





States can be shallow or deep



Shallow state looks at only the past few time steps

Deeps state requires looking far back into the past

Activity!

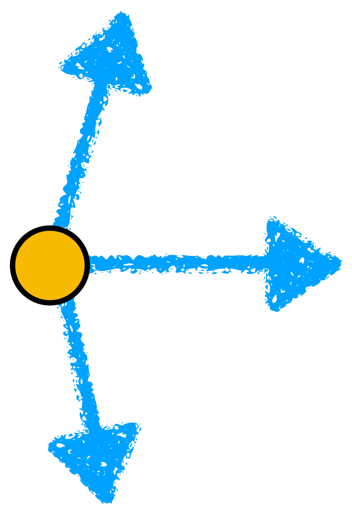
Give an example of deep state

Join by Web **PollEv.com/sc2582**

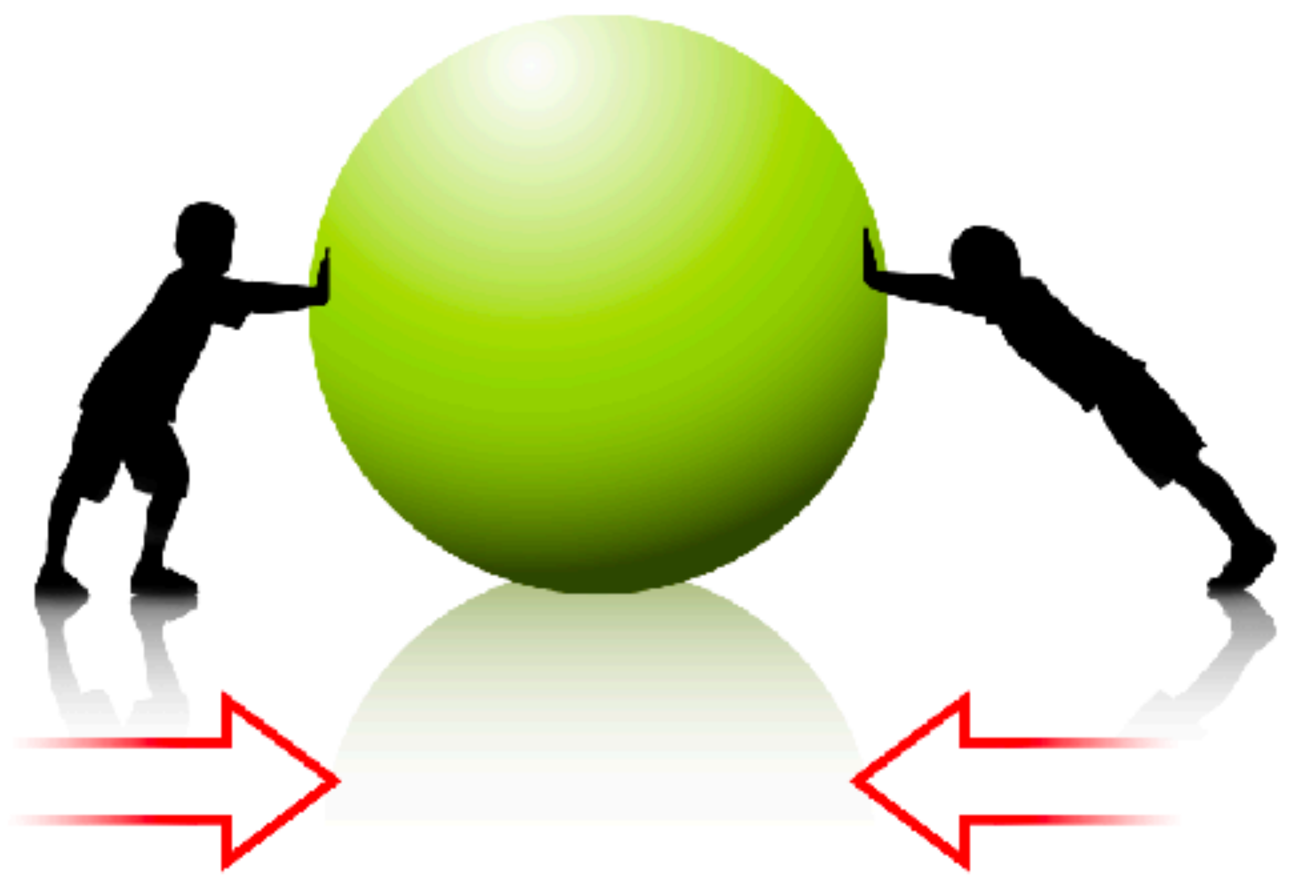
Action

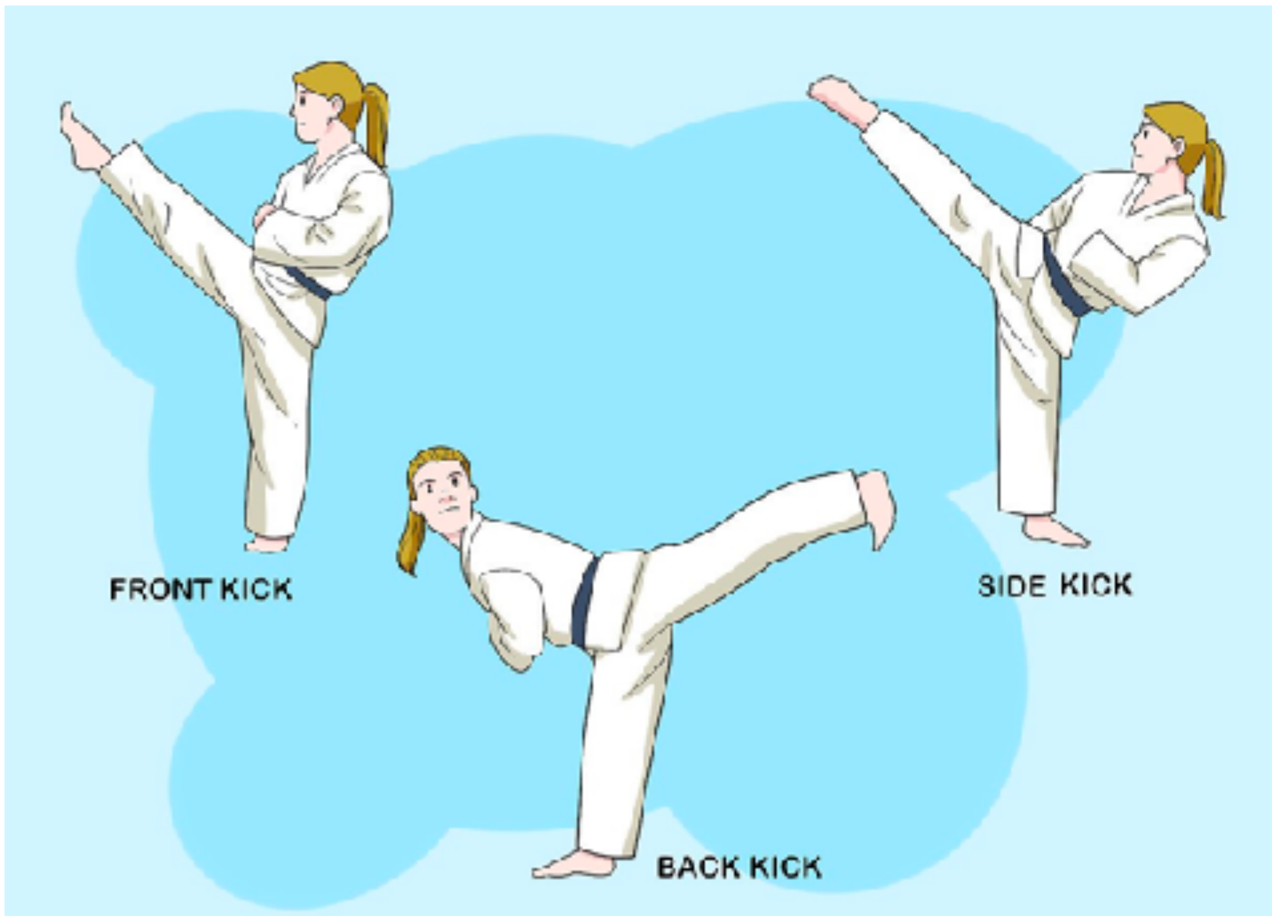
$< S , A , C , \mathcal{T} >$

Doing something:
Control action / decisions



$$a \in A$$



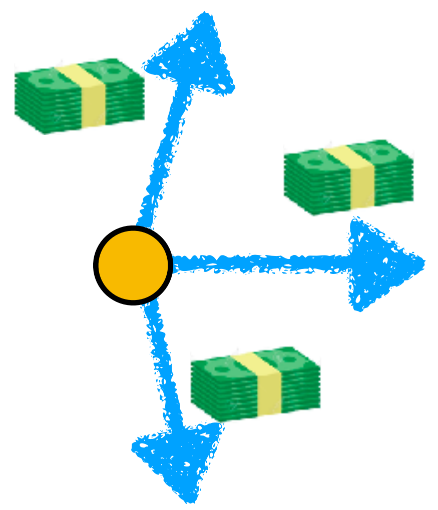


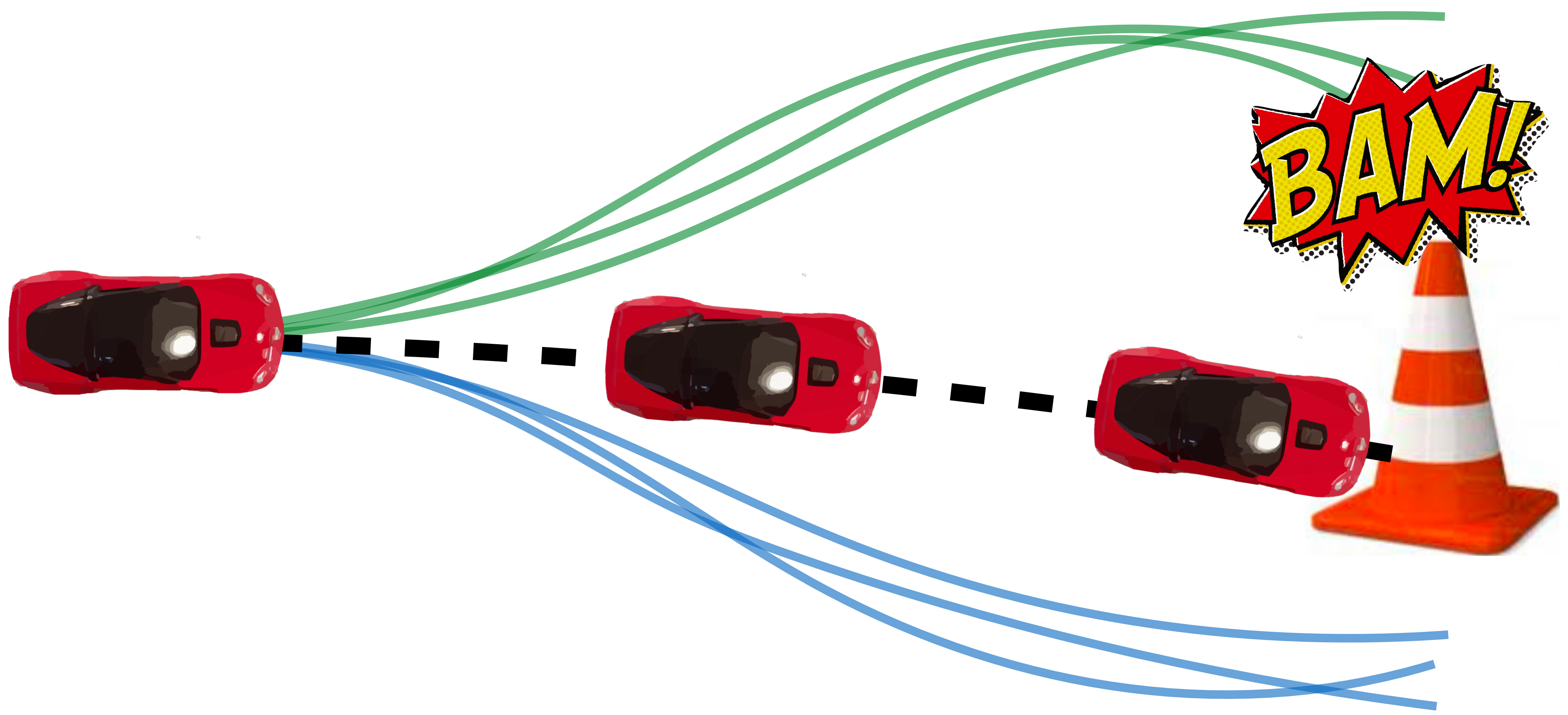
Cost

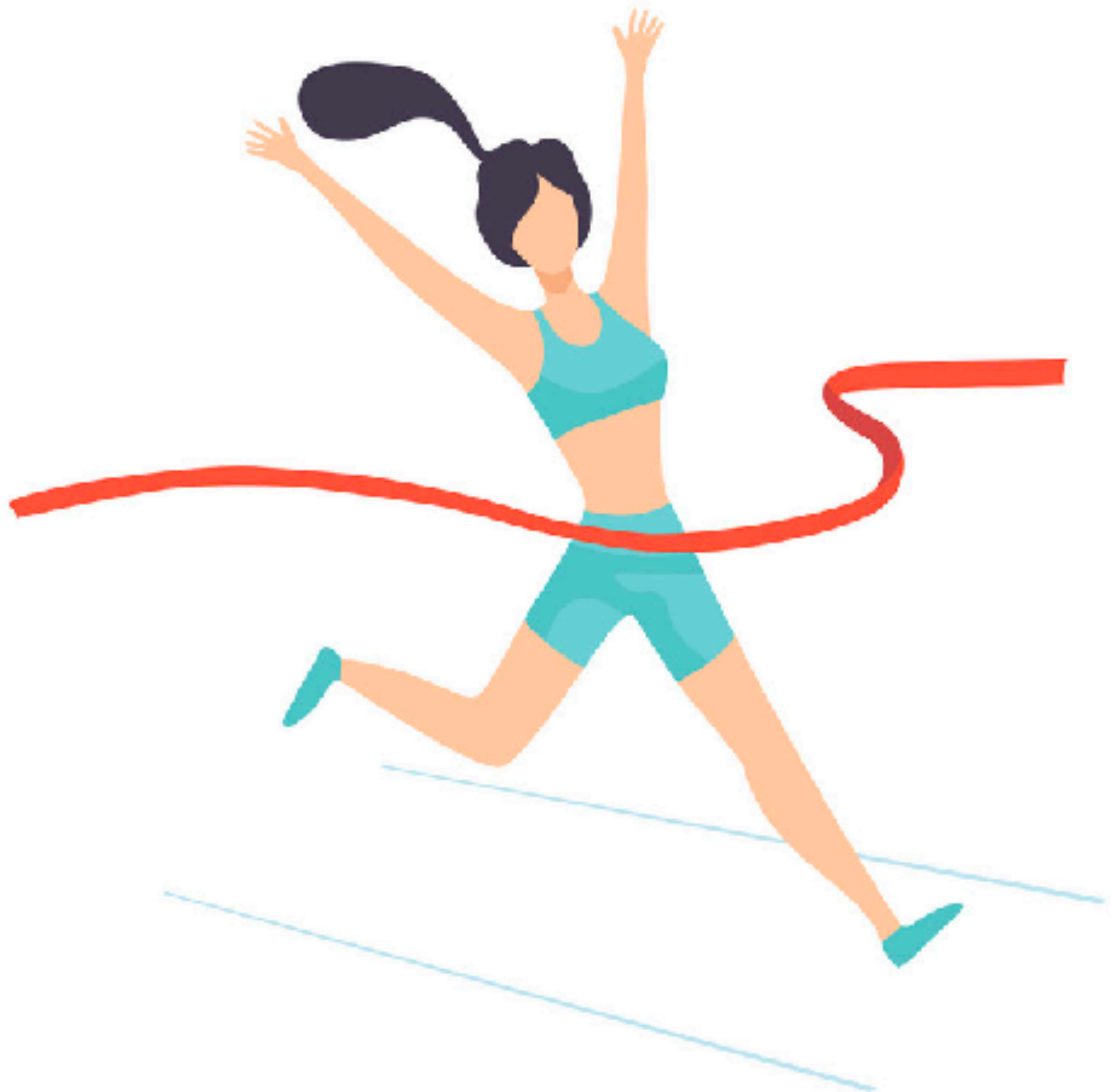
$$< S, A, C, \mathcal{T} >$$

The instantaneous cost of taking an action in a state

$$c(s, a)$$





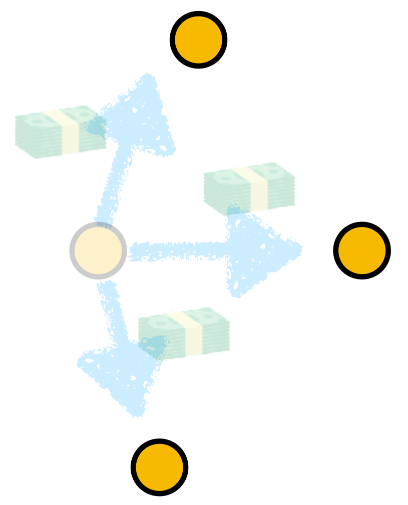


Cost = -Reward

We will use these two interchangeably based on what makes sense

Transition

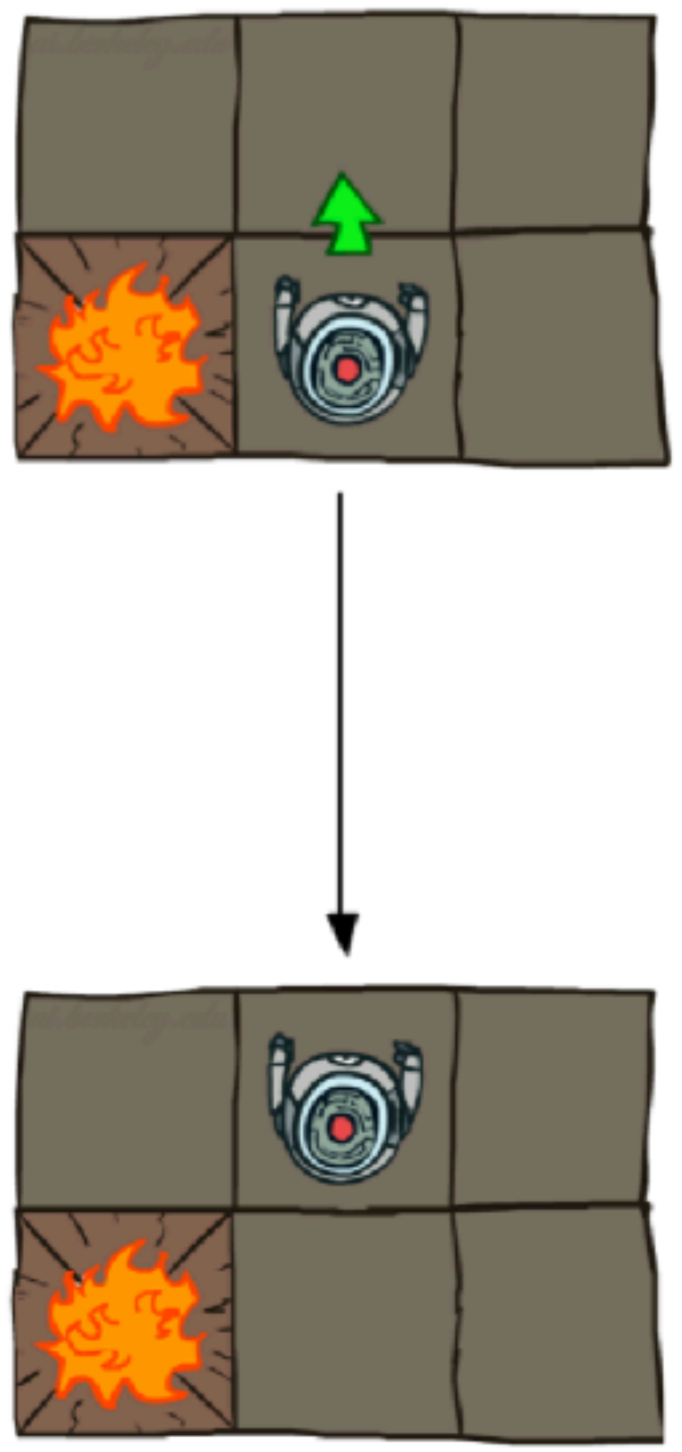
$< S, A, C, \mathcal{T} >$



The next state given state and action

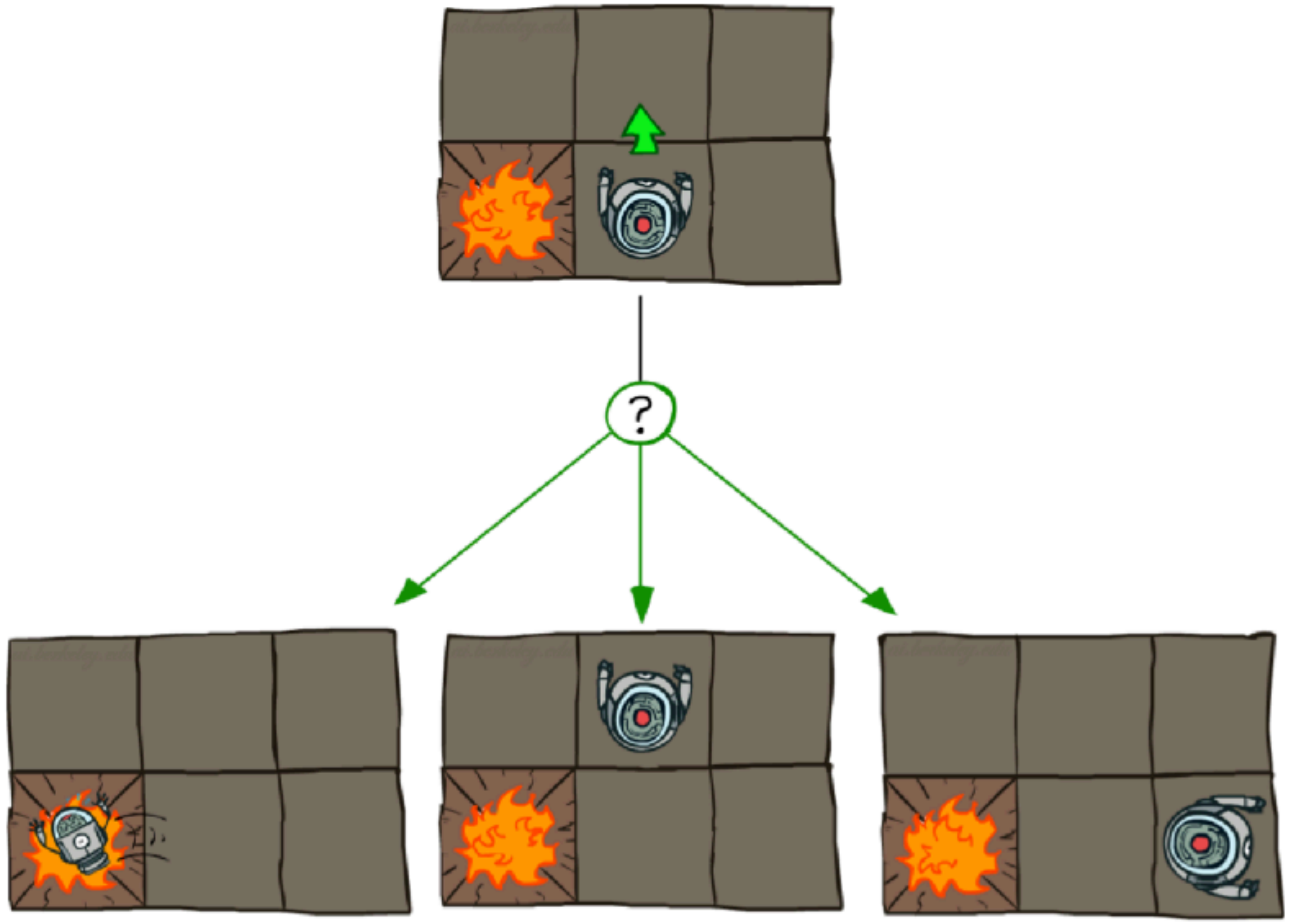
$s' = \mathcal{T}(s, a)$

Deterministic



$s' \sim \mathcal{T}(s, a)$

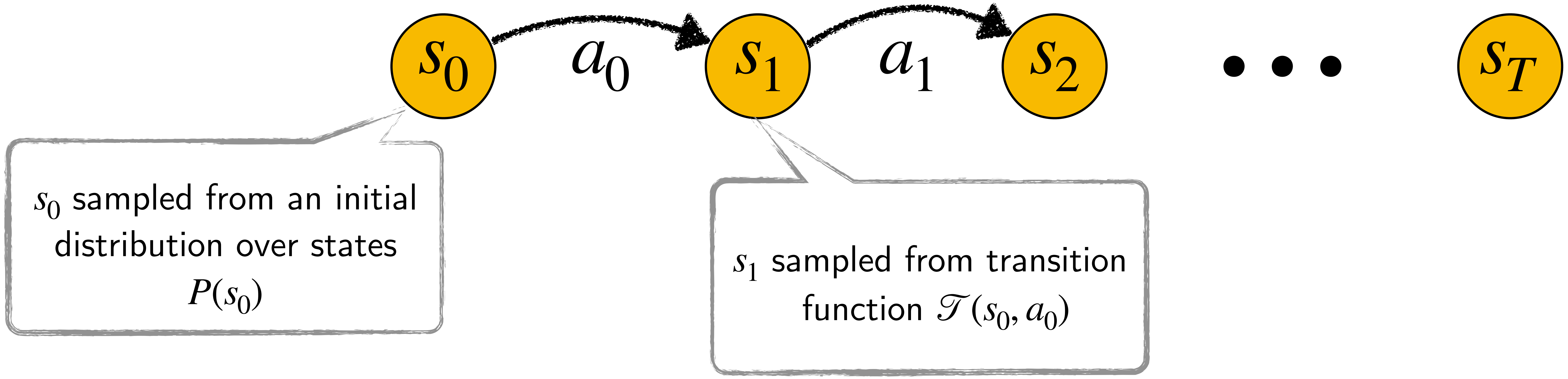
Stochastic



Credit: Dan Klein

State, action, cost, next state ..

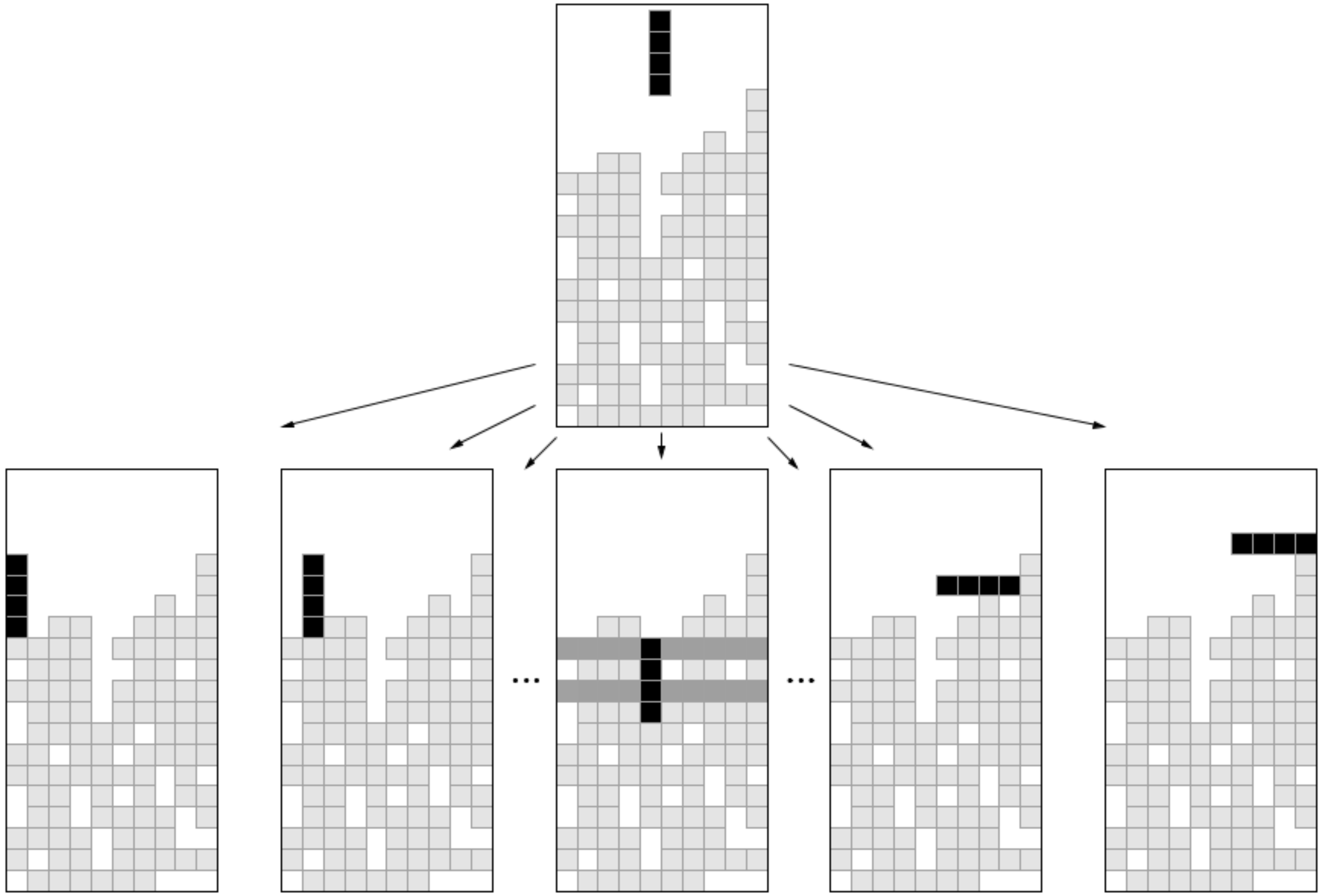
Cost $c(s_0, a_0)$ $c(s_1, a_1)$



"Episode":
A sequence of state, action, costs

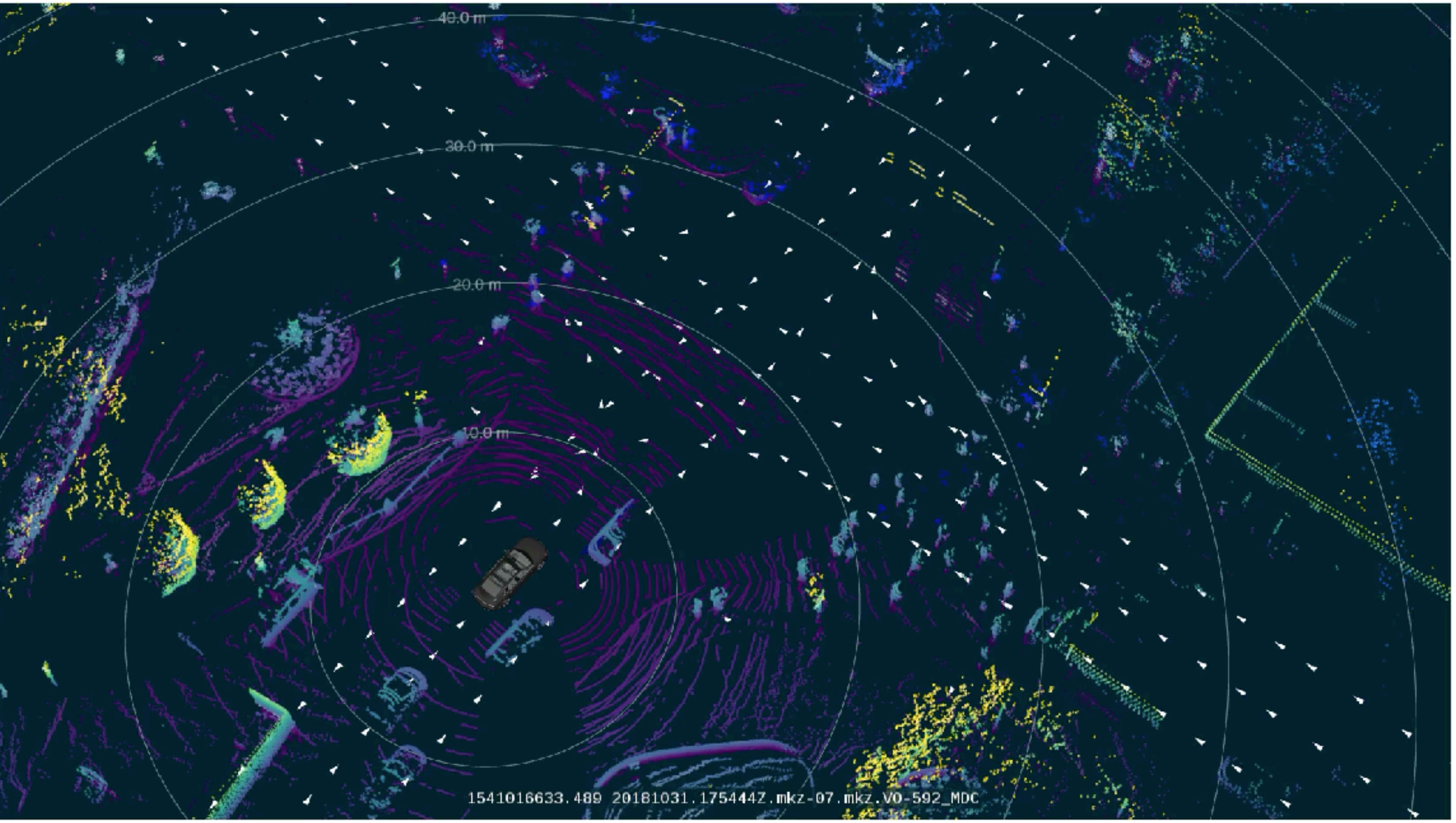
Goal: Minimize total sum of costs $\sum_{t=0}^{T-1} c(s_t, a_t)$

Example 1: Tetris!



$$< S\ , A\ , C\ , \mathscr{T}\ >$$

Example 2: Self-driving



$$< S , A , C , \mathcal{T} >$$

?

Example 3: Coffee making robot

$$< S , A , C , \mathscr{T} >$$

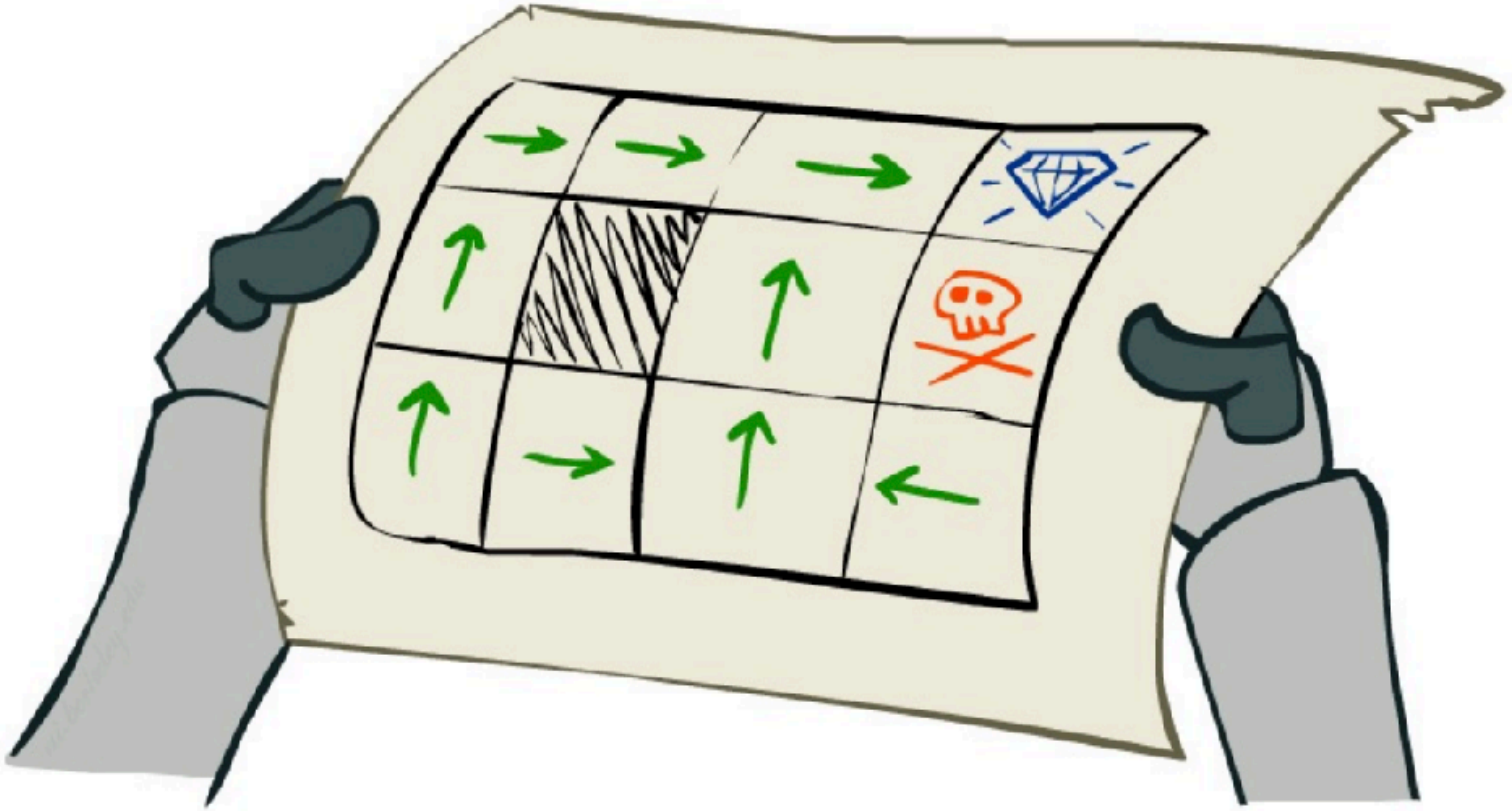
?

What does it mean to solve a MDP?

Solving an MDP means finding a Policy

$$\pi : s_t \rightarrow a_t$$

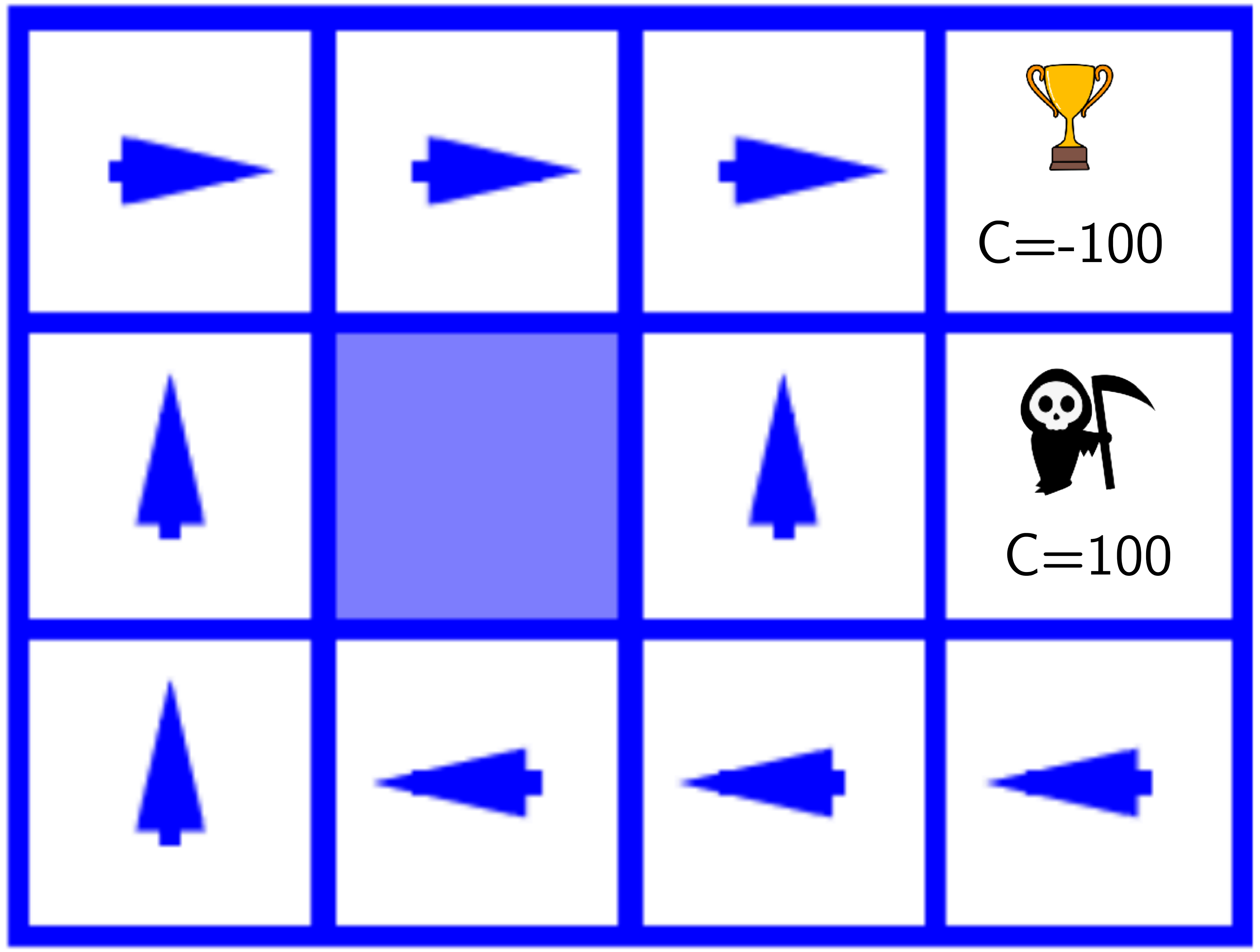
A function that maps state (and time) to action



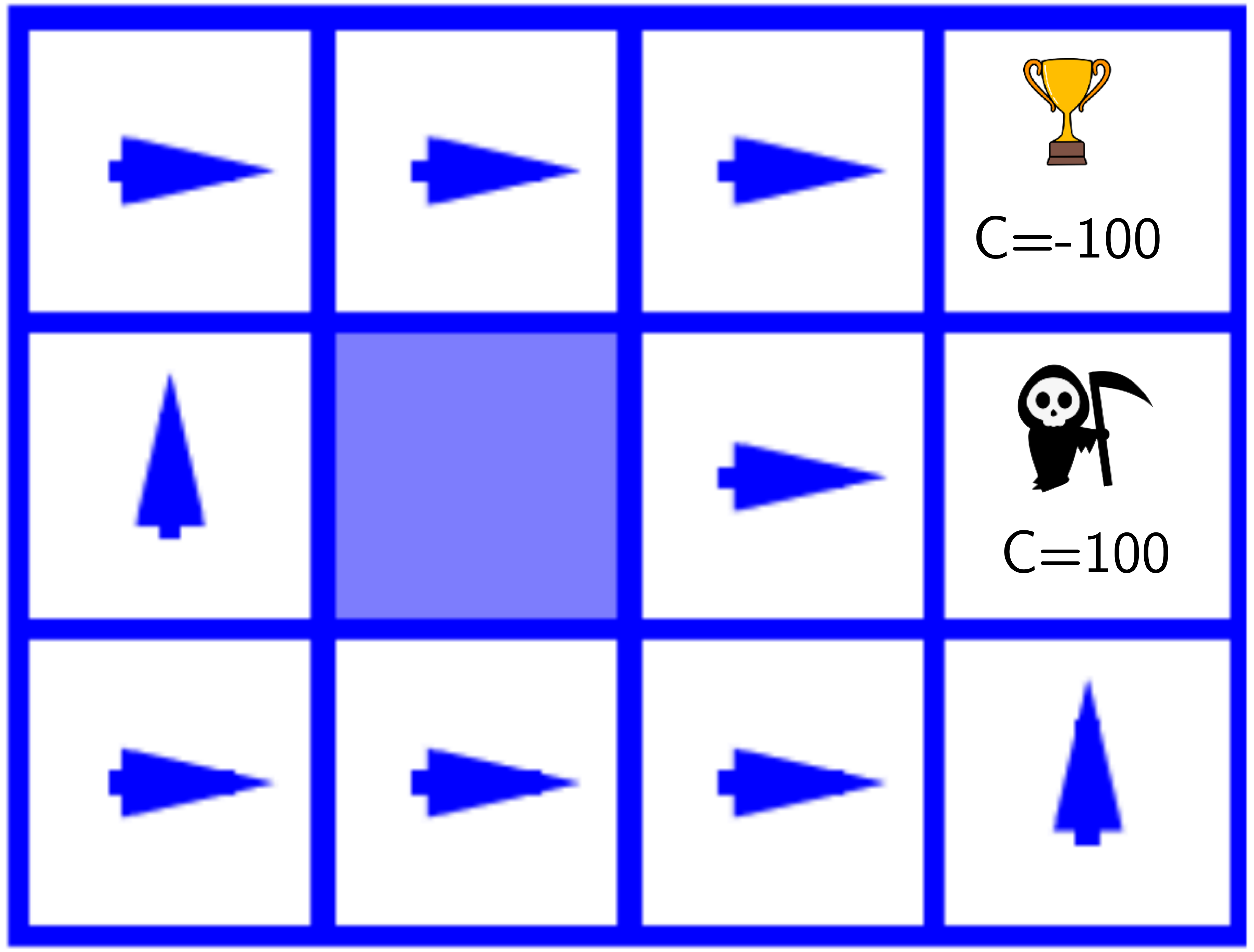
Policy: What action should I choose at any state?

What makes a policy *optimal?*

Which policy is better?



Policy π_1



Policy π_2

What makes a policy *optimal?*

$$\min_{\pi} \mathbb{E}_{\substack{a_t \sim \pi(s_t) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \left[\sum_{t=0}^{T-1} c(s_t, a_t) \right]$$

(Search over Policies)

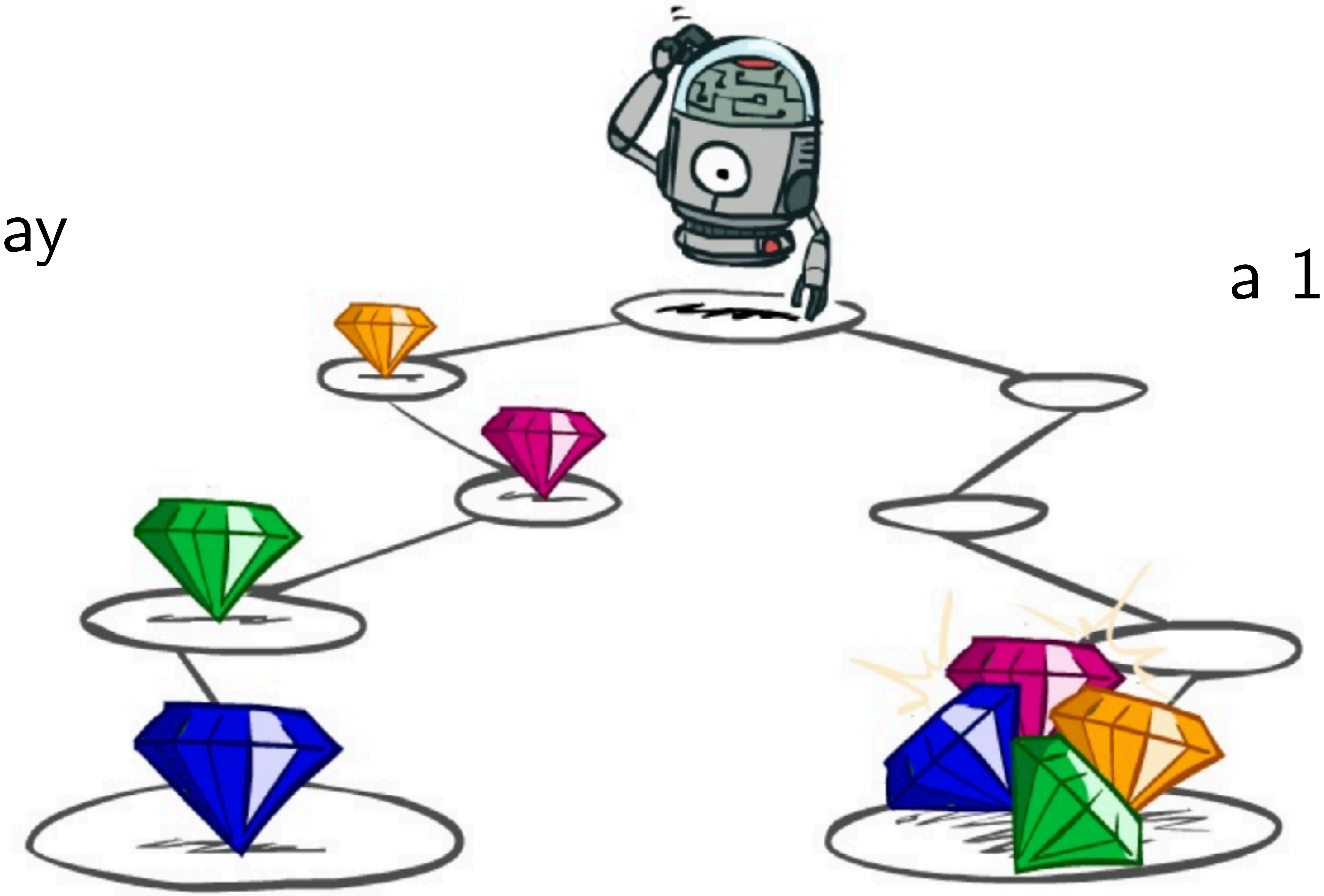
(Sample a start state, then follow π till end of episode)

(Sum over all costs)

One last piece ...

Which of the two outcomes do you prefer?

$50 today



$1 million
a 1000 days later

Discount: Future rewards / costs matter less

1	γ	γ^2
Worth Now	Worth Next Step	Worth In Two Steps

At what discount value does it make sense to take $50 today than $1million in 1000 days?

What makes a policy *optimal?*

$$\min_{\pi} \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \left[\sum_{t=0}^{T-1} \gamma^t c(s_t, a_t) \right]$$

(Search over Policies)

(Sample a start state, then follow π till end of episode)

(Discounted sum of costs)

How do we solve a MDP?

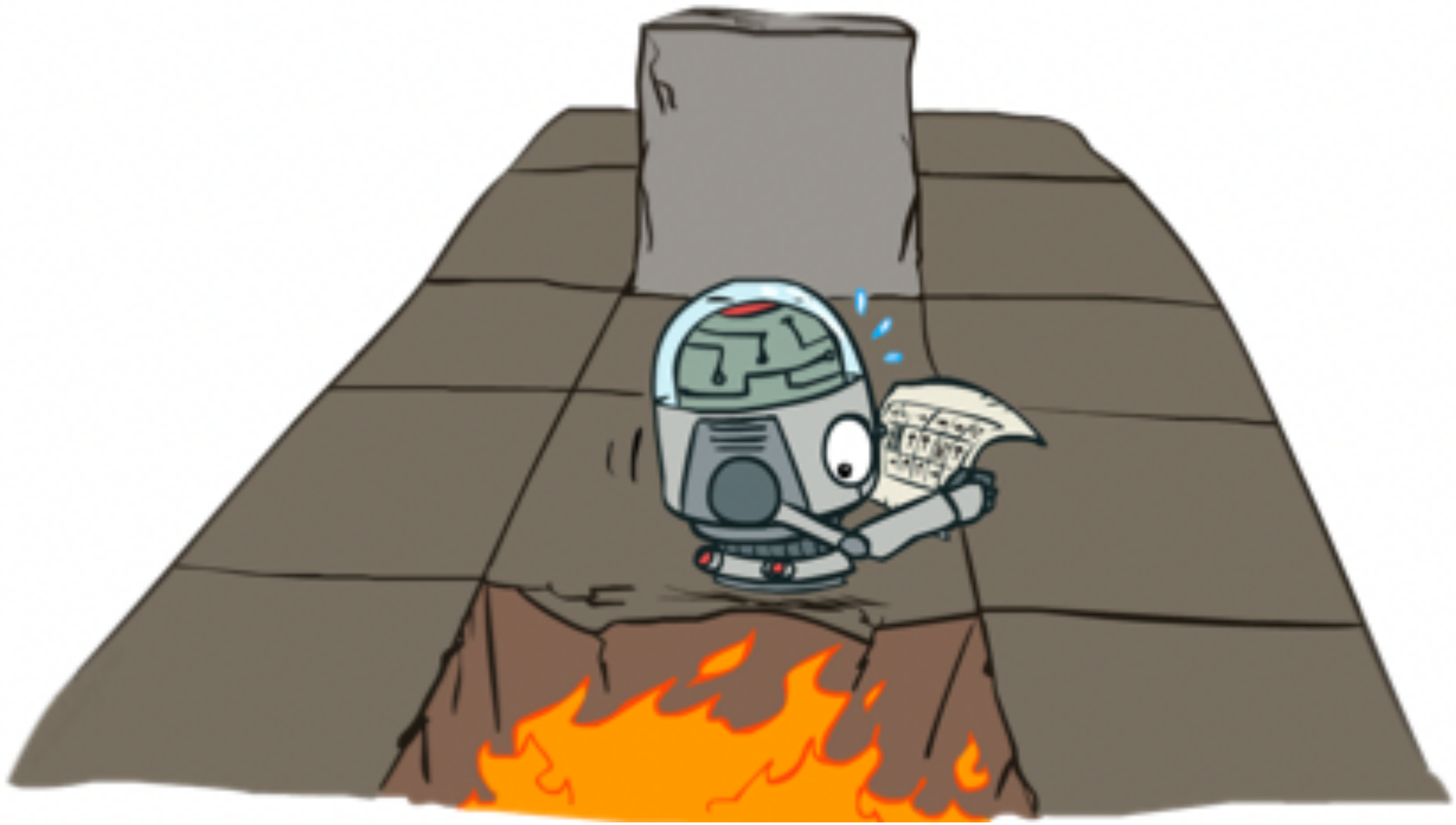


Image courtesy Dan Klein

Let's start with how NOT to solve MDPs

What would brute force do?

$$\min_{\pi} \mathop{\mathbb{E}}_{\substack{a_t \sim \pi(s_t) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \left[\sum_{t=0}^{T-1} \gamma^t c(s_t, a_t) \right]$$

How much work would brute force have to do?

What would brute force do?

$$\min_{\pi} \mathbb{E}_{\substack{a_t \sim \pi(s_t) \\ s_{t+1} \sim \mathcal{T}(s_t, a_t)}} \left[\sum_{t=0}^{T-1} \gamma^t c(s_t, a_t) \right]$$

1. Iterate over all possible policies

2. For every policy, evaluate the cost

3. Pick the best one

There are $(A^S)^T$ Policies!!!!

MDPs have a very special structure

Introducing the "Value" Function

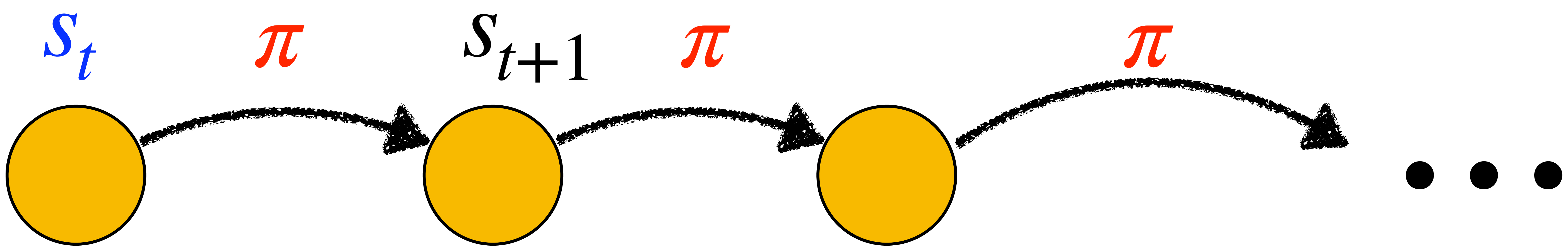
$$V^{\pi}(s_t)$$

Read this as: Value of a policy at a given state and time

Introducing the "Value" Function

$$V^{\pi}(s_t)$$

Read this as: Value of a policy at a given state and time



$$V^{\pi}(s_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \dots$$

The Bellman Equation

$$V^{\pi}(s_t) = c(s_t, \pi(s_t)) + \gamma \mathbb{E}_{s_{t+1}} V^{\pi}(s_{t+1})$$

Value of current state

Cost

Value of future state

Why is this true?

Optimal policy

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{s_0} V^{\pi}(s_0)$$

Bellman Equation for the Optimal Policy

$$V^{\pi^*}(s_t) = \min_{a_t} \left[c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} V^{\pi^*}(s_{t+1})) \right]$$

Optimal Value

Cost

Optimal Value of Next State

Why is this true?

Activity!