Last Time:

- ☐ Intro to HRI

Lecture 2

HRI, FALL '25

Andrea Bajcsy

This Time:

- ☐ sequential decision-making
- ☐ MDPs
- ☐ Policies, Values, Bellman Eqn.

## What is sequential decision-making?

Sequential decision-making involves making a sequence of decisions over time, where each decision affects future outcomes and decisions.

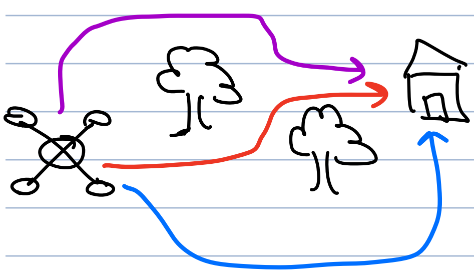
↳ different from one-shot decisions which are a self-contained single decision (e.g. img. classification)

## Why do we care?

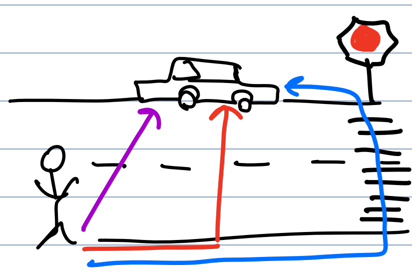
Sequential decision-making is everywhere – playing games, planning a trip or career – and its present in interaction!

In HRI, sequential decision-making will form the "mathematical backbone" of how we model people, robots, and their interaction:

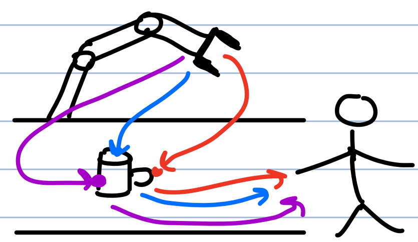
- robots must plan ahead, considering long-term consequences
- humans are sequential decision-makers too: must model them well
- interaction is an ongoing exchange where agents influence each other



How will R fly home?



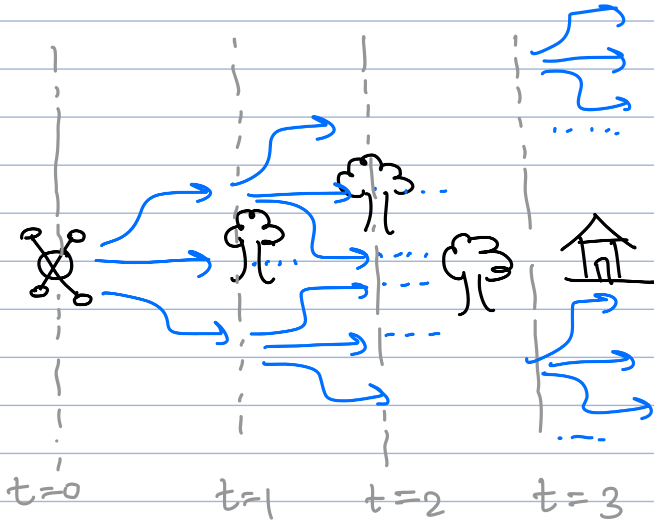
How will H go to their car?



How should R hand H the mug?

## Q Why is it hard?

**A1** decision tree grows exponentially in time horizon.
↳ too many possibilities to compute

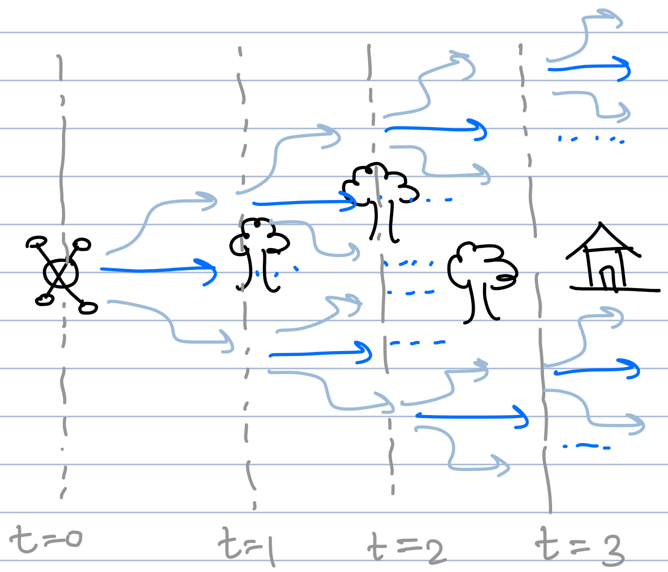



$|A| = 3$

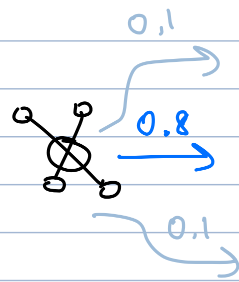
starting from state
there are
$|A|^{T+1}$
sequences of decisions
$= 3^4 = 81$ sequences
with time-horizon of 3!

⇒ Naively planning quickly becomes impossible!

**A2** outcomes of taking actions can be stochastic
↳ same action doesn't always have the same result.

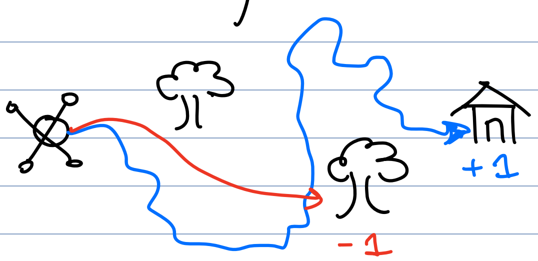



e.g. R chooses to go straight,
but wind might push it
left or right.




⇒ Even with same action sequence,
outcomes may not be the same!

**A3** Hard to assign credit to the right past action(s)
↳ delayed rewards make it unclear which action was responsible.



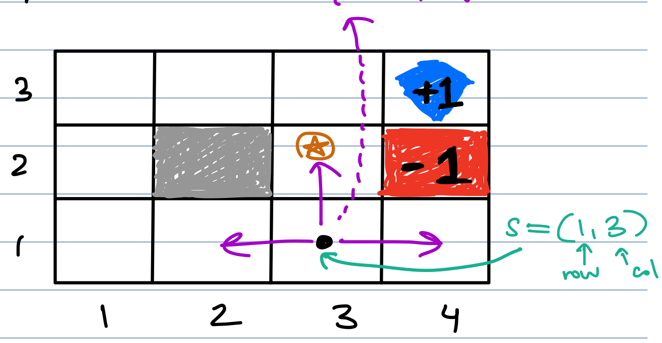

✓ successfully home, but was it all good...?
✗ fail to get home, but was it all bad...?

⇒ If rewards come late, hard to tell which
actions were good / bad!

# Markov Decision Processes (MDPs)

MDPs are a mathematical model for sequential decision-making in a fully observable, stochastic, discrete-time environment with a Markovian transition model.

$$M = \langle S, A, T, r \rangle$$




- **states**: $s \in S$ — state space
- **actions**: $a \in A$ — action space
- **transition function**: $T : S \times A \rightarrow S$

$T(s, a, s')$ OR $T(s=(1,3), a=\uparrow, s'=(2,3)) = 0.8$

$P(s' \mid s, a)$ OR $P(s'=(2,3) \mid s=(1,3), a=\uparrow) = 0.8$

"is distributed as"

$s' \sim P(\cdot \mid s, a)$

$$s' \sim P(\cdot \mid s=(1,3), a=\uparrow) = \begin{cases} (2,3) \rightarrow 0.8 \\ (1,2) \rightarrow 0.1 \\ (1,4) \rightarrow 0.1 \end{cases}$$

- **reward function**: $r : S \times A \times S \rightarrow \mathbb{R}$

$r(s, a, s')$ OR $r(s')$ OR $r(s, a)$

$r(s=(2,3), a=\rightarrow, s'=(2,4)) = -1$

$r(s'=(2,4)) = -1$

$r(s'=(3,4)) = +1$

**Q** What is "Markov" about this?

The transition function is where the Markov part comes into play.

- **Markov Property**: the future is independent of the past, given the present.

This means that action outcomes depend only on the current state, not history!

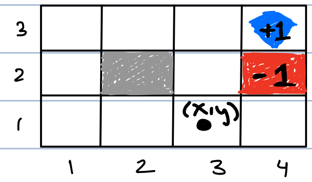
$$\Downarrow$$

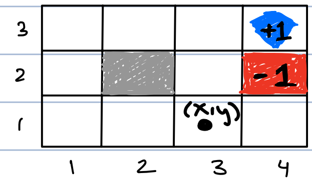
$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots s_0) = P(s_{t+1} \mid s_t, a_t)$$ ! makes planning easier

"Get the Gem" — "Get the gem after visiting A"

$s = (x,y)$

markov holds!




Markov does not hold!

BUT, if $s = (x, y, A\_visited)$ ... yes!

! If a problem is Markov depends on how we define the state

# Policies

The object we ultimately want to <u>solve</u> for is the policy: OR, knowing what is the best decision to make @ any state.

$\pi : S \to A$ "deterministic, $\pi(s)$"

$\pi : S \to \Delta A$ "stochastic: $\pi(a|s)$"

probability simplex, "distribution over actions"

$\pi : S \to A$

ex. policy (not very good one tho...)

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | → | → | → | +1 |
| 2 | → | | → | -1 |
| 1 | → | → | → | → |

Evaluate the quality of a policy $\pi$ by the <u>expected cumulative reward</u> induced by the policy. ⊛ we will get to the "expected" bit in a bit...

① "roll out" or simulate $\pi$ from each $s_0 \in S$ to get a trajectory sequence $\{s_0, \underbrace{\pi(s_0)}_{a_0}, s_1, \underbrace{\pi(s_1)}_{a_1} \dots\}$

② Evaluate the <u>utility</u> of the rollout, called <u>cumulative reward</u>:

$$R(s_0, s_1, s_2 \dots) \triangleq r(s_0) + r(s_1) + \dots = \sum_{t=0}^{\infty} r(s_t)$$

or $r(s_t, a_t)$, $r(s_t, a_t, s_{t+1})$

$\pi_1 : S \to A$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | → | → | → | +1 |
| 2 | → | | → | -1 |
| 1 | → | → | → | → |

$\pi_2 : S \to A$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 3 | → | → | → | +1 |
| 2 | ↑ | | ↑ | -1 |
| 1 | ↑ | → | ↑ | ← |

better policy b/c more reward!

- an <u>optimal policy</u> ($\pi^* : S \to A$) yields the highest cumulative reward

(!) But what counts as a "good" policy depends on the reward function!

## Sparse vs. Dense Rewards

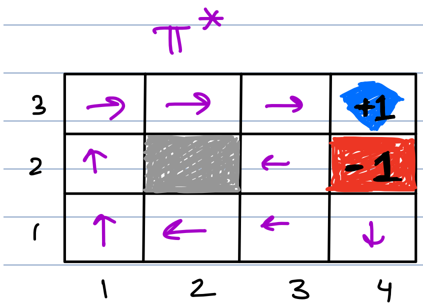
<u>Problem</u>: If the only reward is a +1 at the goal (and -1 @ lava), many policies look equally good because there is no penalty for inefficiency! In other words, sparse rewards often <u>underspecify</u> what we "actually" want the agent to do!

⊛ Underspecified rewards is a HUGE problem in HRI and AI Alignment!!!

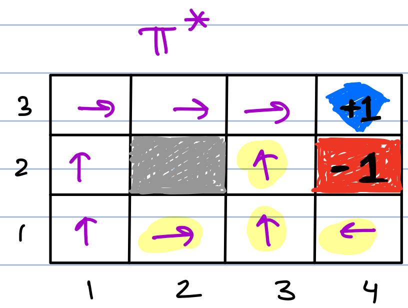
**Solution:** Dense rewards — frequent feedback (good / bad) to the agent over time, instead of just rewarding outcomes at the end.

⊕ helps guide the agent to behave in the way we actually want.
⊕ specify not just what the agent should do (i.e. goal), but how to do it.
⊕ faster agent learning
⊖ HARD TO SPECIFY + PROVIDE → see later lectures on this!

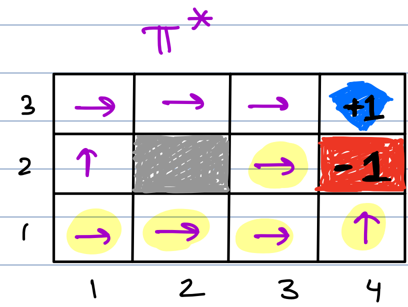
**Example:** "Living" rewards (time step) are a kind of dense reward that incentivizes efficiency by penalizing unecessary steps.




$r(s) = -0.01$
↑ small living penalty
⇒ detours around obstacle + lava, & "bumps" away from walls / obstacle




$r(s) = -0.4$
↑ med. living penalty
⇒ no bumping against walls, direct path




$r(s) = -2.0$
↑ large living penalty
⇒ This one wants to exit ASAP!

## Discount Factor

- denoted by $\gamma \in [0,1]$ ⇒ sometimes MDP tuple is written as $\langle S, A, T, r, \gamma \rangle$
- describes an agent's preference for current rewards over future ones when $\gamma < 1$. If $\gamma = 1$, our agent wants max reward over all tsteps.

$$R(s_0, s_1, s_2, \ldots) = \gamma^0 r(s_0) + \gamma^1 r(s_1) + \gamma^2 r(s_2) + \cdots = \sum_{t=0}^{\infty} \gamma^t r(s_t)$$

discounted cumulative reward

⊛ discounting is a good model of human & animal preferences

↳ See: "Models of Temporal Discounting 1937–2000" by Till Grüne-Yanoff.
→ connections w/ psych + econ.

Ultimately, we are searching for a policy $\pi^*: S \to A$ that maximizes the expected discounted cumulative reward:

trade off shorter vs. longer horizon rewards

$$\pi^* = \underset{\pi}{\text{argmax}} \; \mathbb{E}_{\tau \sim p_\pi(\tau)} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right]$$

"return the policy which maximizes..."

rollouts are stochastic so take expectation over all trajectories under the policy $\pi$!

let $\tau := (s_0, a_0, s_1, a_1, \ldots)$

$$p_\pi(\tau) = p(s_0) \prod_{t=0}^{\infty} p(s_{t+1} | s_t, a_t)$$

(*) note: this is for deterministic $\pi$! for stochastic we have

$$= p(s_0) \prod_t \underline{\underline{\pi(a|s)}} \, p(s_{t+1}|s_t, a)$$