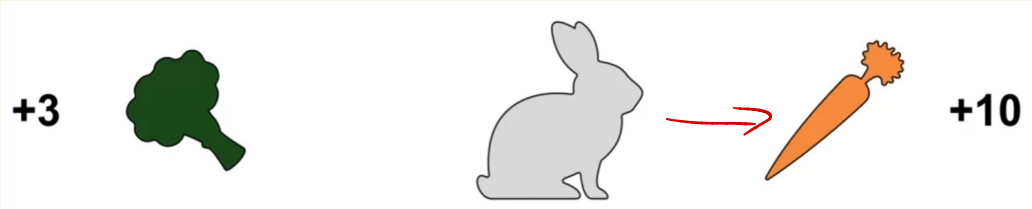
# Chapter 3 - Finite Markov Decision Processes:

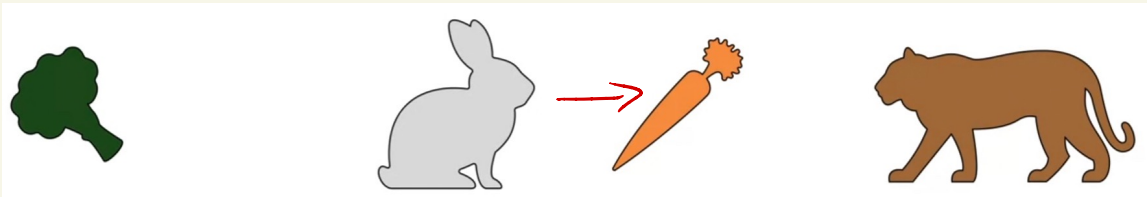
Dec 24, 2021

MDP's involve sequential decision making, where actions not only influence immediate rewards, but also subsequent states/situations and future rewards.

↳ Involves delayed reward and tradeoff between immediate and delayed rewards.

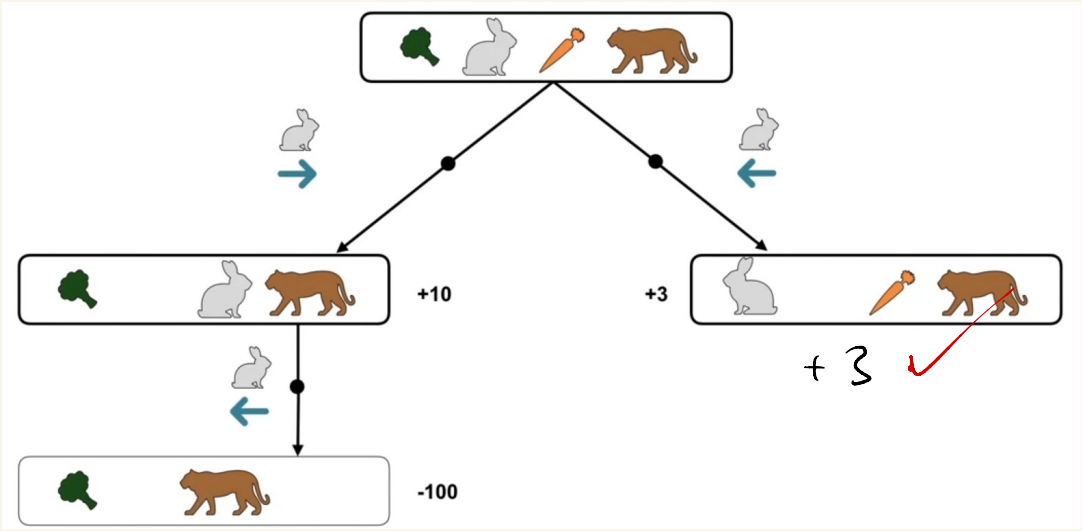
e.g.



✡ Rabbit will still go towards carrot if using k-bandit model because it is the best immediate action. It does not care if it gets eaten afterwards.

* We want the agent to take into account all sequences and their rewards.




+3 ✓

-90 ✗

## 3.1 - The Agent-Environment Interface:

Agent: the learner and decision maker

Environment: everything outside the agent that it interacts with

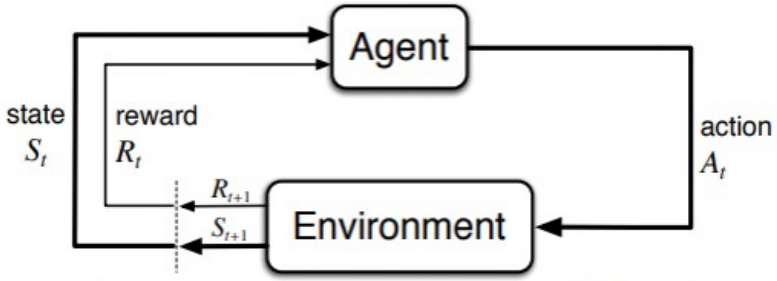



**Figure 3.1:** The agent–environment interaction in a Markov decision process.

Agent and environment interact at a series of discrete time steps: $t = 0, 1, 2, 3 \ldots$

↳ At each time step $t$:

1. Agent receives some representation of the environment's state, $S_t \in S$.
2. Selects an action based on $S_t$, $A_t \in A(s)$.

↳ At next time step:

1. Agent receives a numerical reward, $R_{t+1} \in R \subset \mathbb{R}$.
2. Reaches a new state, $S_{t+1}$.

## Overall Sequence / Trajectory:

$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3 \ldots$

## Finite MDP:

Finite MDP's have finite sets of states, actions, and rewards $(S, A, R)$.

↳ $R_t$, $S_t$ have well-defined, discrete probability distributions that only depend on the preceding state and action.

## Dynamics of the MDP:

$$p(s', r | s, a) \equiv Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$$

for all $s', s \in S$, $r \in R$, $a \in A(s)$

✡ p is just a probability distribution and defines the dynamics of the MDP.

↳ $p: S \times R \times S \times A \to [0, 1]$ ✡ p must be non-negative

↳ $\sum_{s' \in S} \sum_{r \in R} p(s', r | s, a) = 1 \quad \forall s \in S, a \in A(s)$

✡ sum of all possible next states and rewards must add up to 1

## Markov Decision Process:

In a Markov decision process, the probabilities given by p completely characterize the environment's dynamics.

↳ The probability for each $S_t$, $R_t$ value completely depends on previous state and action: $S_{t-1}$, $A_{t-1}$.

↳ Not at all on anything earlier.

→ The state is said to have the Markov property if it includes info on all aspects of past agent-environment interaction that can affect the future. ✡ Not present!

# Recycling Robot Example!

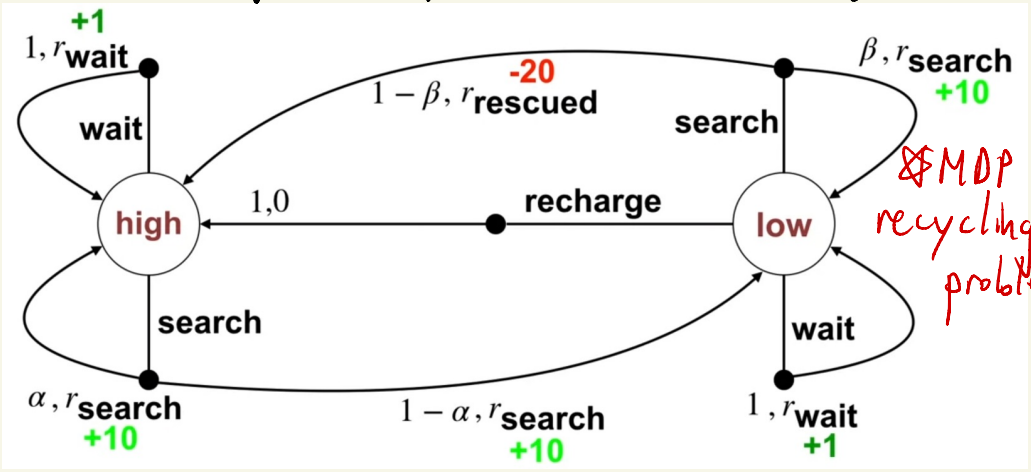
$S = \{low, high\}$ # energy

$A(low) = \{search, wait, recharge\}$

$A(high) = \{search, wait\}$


## Dynamics of the Recycling Bot!

1. Searching w/ energy high has a chance to reduce it to low → $1-\alpha$.
2. Waiting for cans does not drain the battery.
   ↳ State doesn't change, but gives $r_{wait} = +1$.
3. Searching w/ low energy may cause it to need rescuing → $1-\beta$, $r_{rescued} = -20$.
4. Recharging battery restores it to high, +0.




# MDP for recycling bot problem

## MDP Formalism:

Can be abstract and flexible.

- ↳ States can be low-level sensory readings to high-level abstractions.
- ↳ Actions can be low-level motor controls to high-level decisions.
- ↳ Time steps can be very small or very large.
  - ↳ Can also be based on time or when actions are performed.

## 3.2 - Goals and Rewards:

### The Reward Hypothesis:

The goal of an agent is to maximize the total amount of reward it receives.

- ↳ At each time step, the reward is a scalar $R \in \mathbb{R}$.
- ↳ The reward (a special signal) is passed from the environment to the agent.
  - ↳ The reward signals what you want the agent to achieve but not how.

# 3.3 - Returns and Episodes!

How do we formally define the reward hypothesis?

$$G_t \doteq \underbrace{R_{t+1} + R_{t+2} + R_{t+3} \ldots + R_T}_{\text{sequence of rewards}} \rightarrow \text{final time step}$$

$G_t$: return

Overall, we seek to maximize the expected return!

$$E[G_t] = E[R_{t+1} + R_{t+2} + R_{t+3} \ldots + R_T]$$

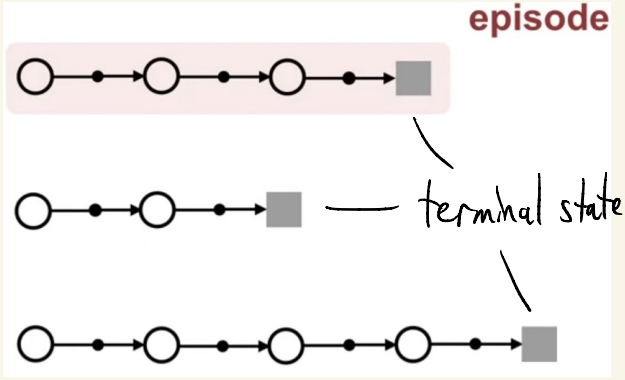
## Episodic Tasks!

Episodes are subsequences of an agent-environment interaction which possesses a final step, T.

↳ Each episode ends in a terminal state, followed by a reset to a standard starting state.

↳ Each episode begins independent of how the last one ended.

↳ Tasks w/ episodes like this are episodic tasks.




✡ Same terminal state for each episode, but different rewards for different outcomes.

e.g. Chess
One game = one episode.
Each game starts w/ reset pieces.
Game ends w/ checkmate (terminal state).
White wins, white gets rewarded. Same for black.

## Continuing Tasks:

Continuing tasks do not break into identifiable episodes, but instead go on continually without limit.

↳ Natural way to formulate an ongoing process-control task.

↳ We can't use standard return since $T = \infty$ and maximizing $G_t$ could mean $G_t = \infty$.

∴ We use discounting and maximize a discounted return.

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

↳ $\gamma$ = discount rate, $0 \leq \gamma \leq 1$

∴ $G_t$ is finite as long as reward sequence $\{R_t\}$ is bounded.

$\gamma = 0$ : Maximizes immediate reward

$\gamma = 1$ : Strongly takes into account future rewards

If $\gamma = 1$, it is an episodic task

For successive time steps:

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots$$
$$= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} \ldots)$$
$$= R_{t+1} + \gamma G_{t+1}$$

$$\therefore \boxed{G_t = R_{t+1} + \gamma G_{t+1}}$$

If reward is constant:

$$R_{t+1} = R_{t+2} = R_{t+3} \ldots = C$$

$$\hookrightarrow \boxed{G_t = C \sum_{k=0}^{\infty} \gamma^k = \frac{C}{1-\gamma}}$$ Constant reward

Additional Note:

The Markov property does not mean that the state representation tells us everything we need to know, only that it has not forgotten anything useful to know.

e.g. $\gamma = 0.8$, $R_1 = 5$ followed by infinite sequence of 10's. Find $G_0$.

$$G_t = R_{t+1} + \gamma G_{t+1}$$

$$\hookrightarrow G_0 = R_1 + \gamma G_1$$

$$= 5 + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

$$= 5 + 0.8 \cdot 10 \cdot \frac{1}{1-0.8}$$

$$= 45$$

Final Note!

Remember that MDP's are just a way to model RL tasks! We have yet to cover algorithms to actually solve them and/or teach the agent.