# CMU 15-781

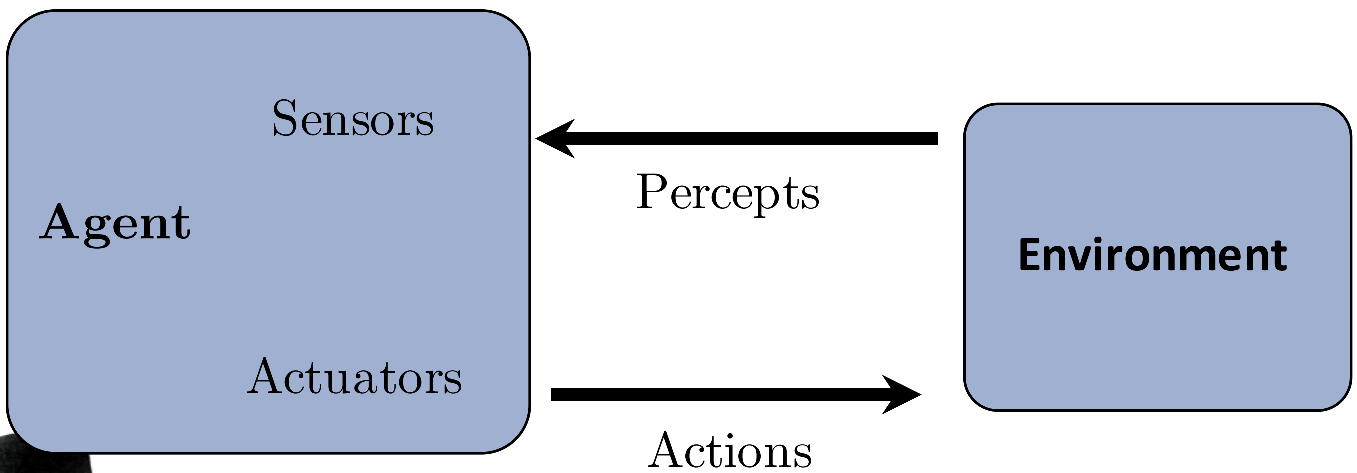
## Lecture 10:
## Markov Decision Processes I

Teacher:
Gianni A. Di Caro

# Decision-Making, so far ...

- Known environment
- Full observability
- Deterministic world

*Plan*: Sequence of actions with **deterministic consequences**, each next state is known with certainty




# ACTIONS' OUTCOMES ARE USUALLY *UNCERTAIN* IN THE REAL WORLD!

Action effect is *stochastic*: probability distribution over next states

Deterministic, one single successor state: $(s, a) \rightarrow s'$

Probabilistic, conditional distribution of successor states:

$(s, a) \rightarrow P(s'|s, a)$

In general, we need a sequence of actions (decisions):

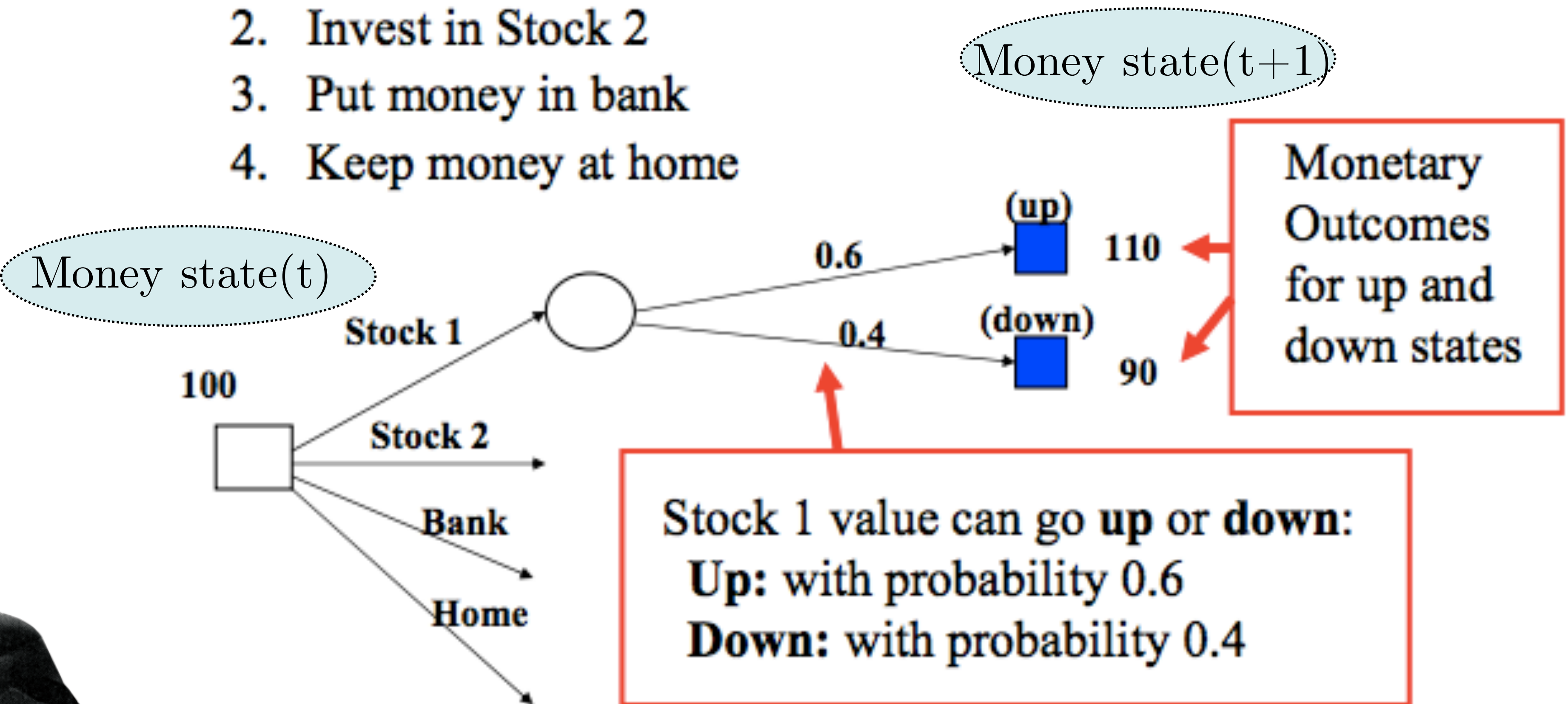
$$(s_t, a_t) \rightarrow P(s_{t+1} = s' \mid s_t = s,\ a_t = a)$$

In general, the outcome can depend on all history of actions:

$$P(s_{t+1} = s' \mid s_t, s_{t-1}, \ldots, s_0, a_t, a_{t-1}, \ldots, a_0) = P(s_{t+1} = s' \mid s_{t:0}, a_{t:0})$$

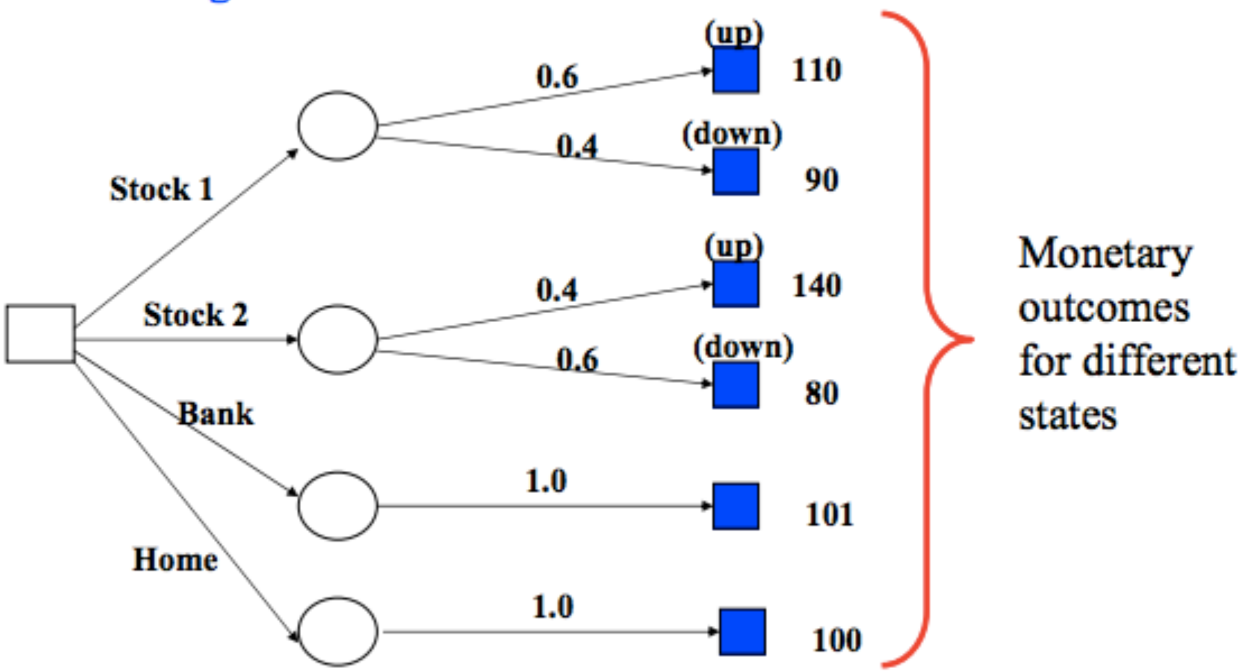
# STOCHASTIC DECISION MAKING EXAMPLE

1. Invest in Stock 1
2. Invest in Stock 2
3. Put money in bank
4. Keep money at home




# Stochastic Decision making example

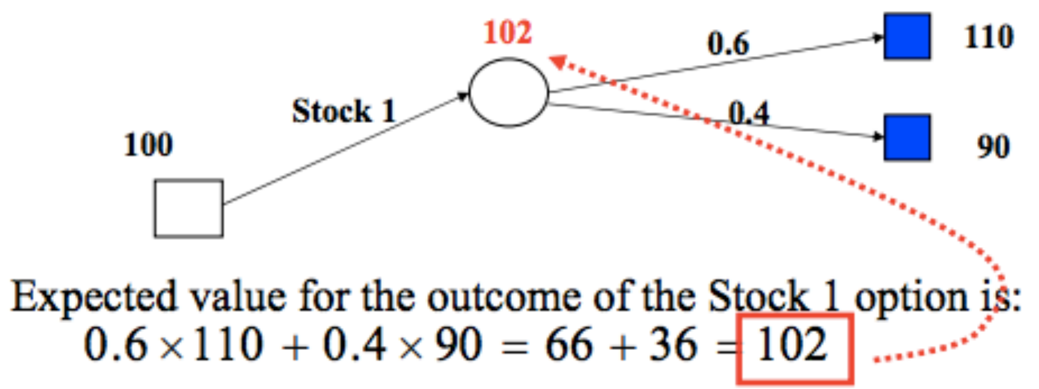
Investing of $100 for 6 months




How a *rational agent* makes a choice, given that its *preference* is to make money?

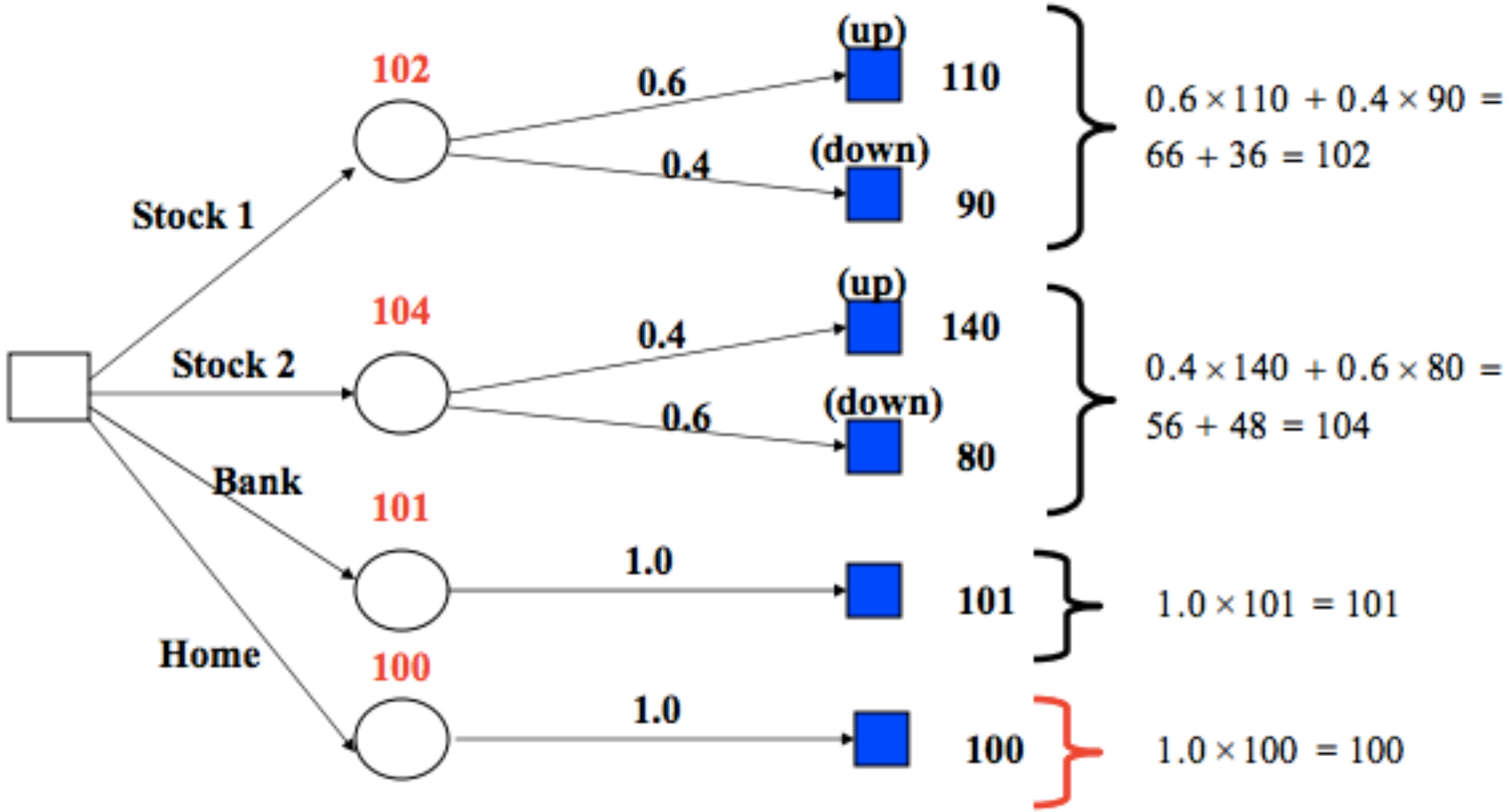
# Expected values

- $X$ = random variable representing the monetary outcome for taking an action, with values in $\Omega_X$ (e.g., $\Omega_X = \{110, 90\}$ for action Stock 1)
- Expected value of $X$ is: $$E(X) = \sum_{x \in \Omega_X} x P(X = x)$$
- Expected value summarizes all stochastic outcomes into a single quantity



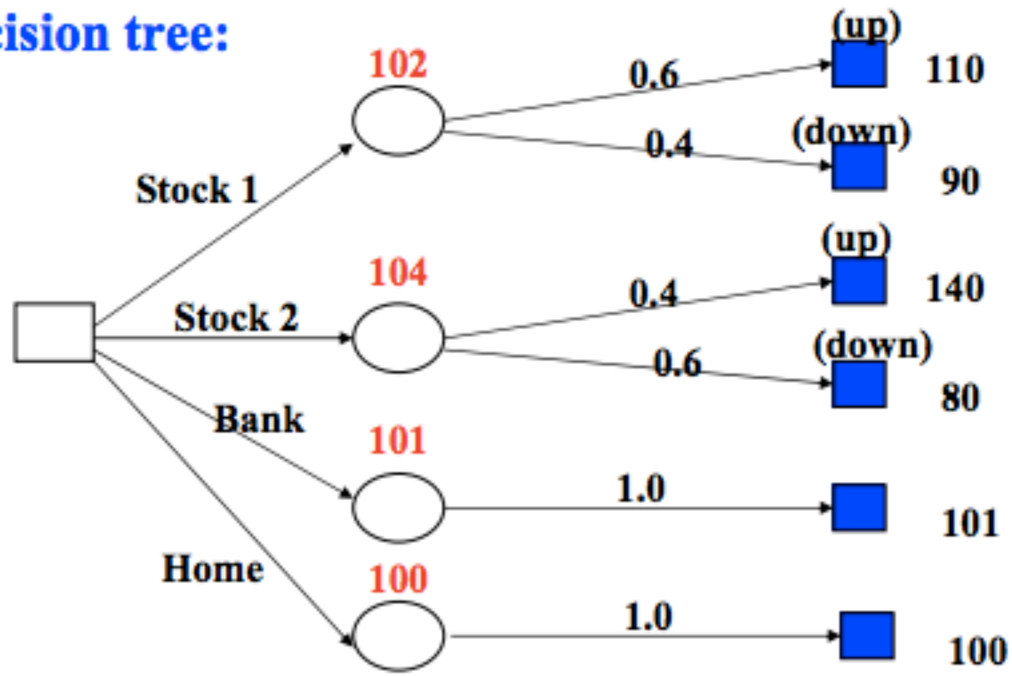

# Expected values

## Investing $100 for 6 months




# Optimal decision

- **Decision tree:**

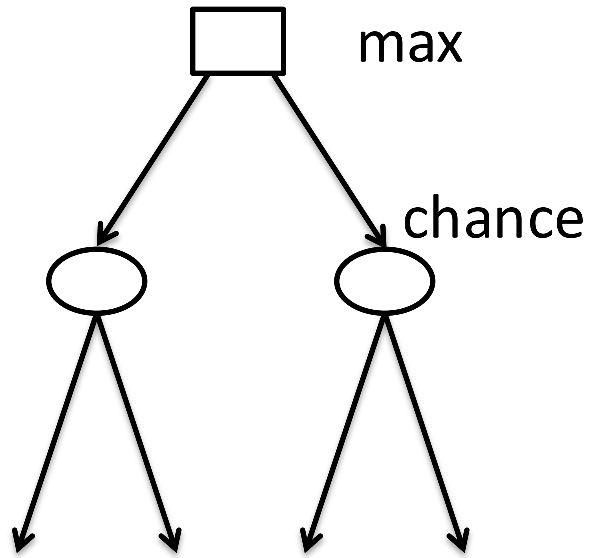



The optimal decision is the action that maximizes the expected outcome

- **decision node**
- **chance node**
- **outcome (value) node**

# Where do probabilities values come from?

- Models
- Data
- For now assume we are *given* the probabilities for any chance node

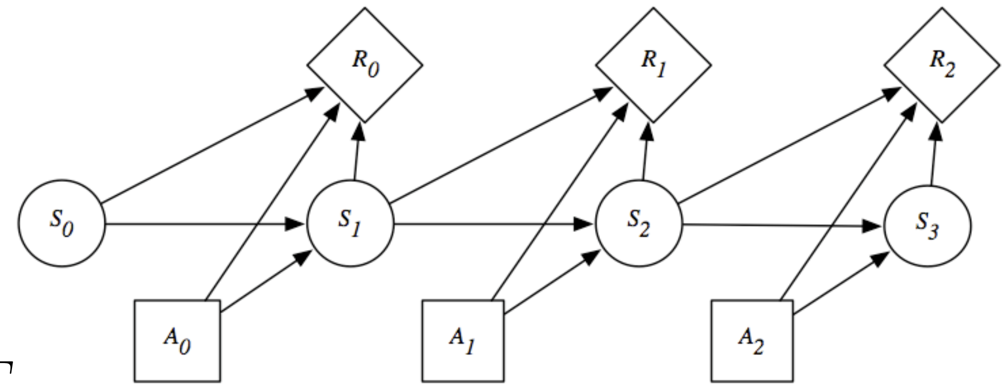


# Markov Decision Processes (MPDs)

- Consider multi-step decisions under stochastic action effects
- Add a state-dependent reward (cost) for each action taken
- Assume as known the probability model (system dynamics)
- Assume that only the current state and action matters for taking a decision **Markov property (memoryless)**:

$$P(s_{t+1} = s' \mid s_{t:0}, a_{t:0}) = P(s_{t+1} = s' \mid s_t, a_t)$$

# Markov Decision Processes (MPD)




- A set $S$ of world states
- A set $A$ of feasible actions
- A stochastic transition matrix $T$,
  $T : S \times S \times A \times \{0, 1, \ldots H\} \mapsto [0, 1]$, $T(s, s', a) = P(s'|s, a)$
- A reward function $R$
  $R(s)|R(s, a), |R(s, a, s'), \; R : S \times A \times S \times \{0, 1, \ldots H\} \mapsto \mathbb{R}$
- A start state (or a distribution of initial states)
- Terminal (goal) states

**Goal:** define decision sequences that maximize a given function of the rewards

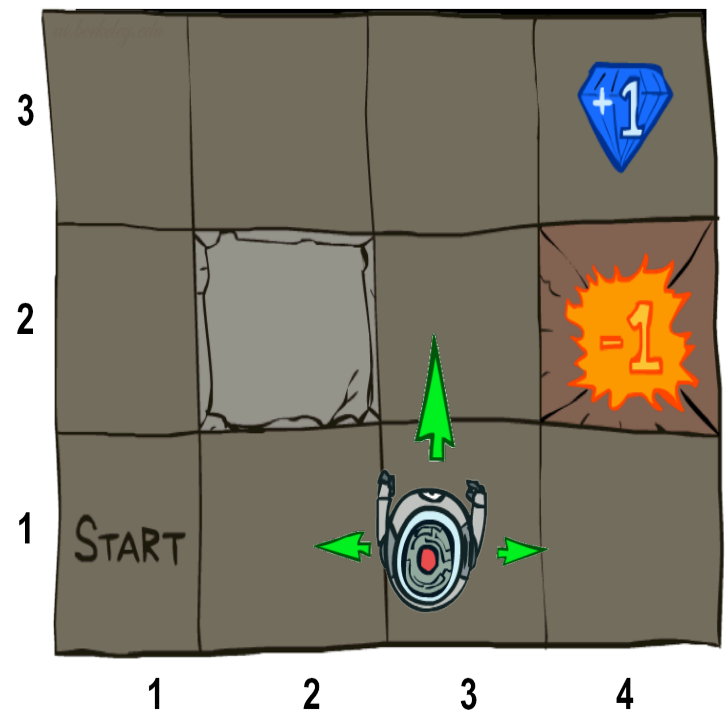
# Taxonomy of Markov Processes

- Markov decision process (MDP)
- Markov reward process MDP $\setminus$ {Actions}
- Markov chain: MDP $\setminus$ {Actions} $\setminus$ {Rewards}

All share the *state set* and the *transition matrix*, that defines the internal stochastic dynamics of the system

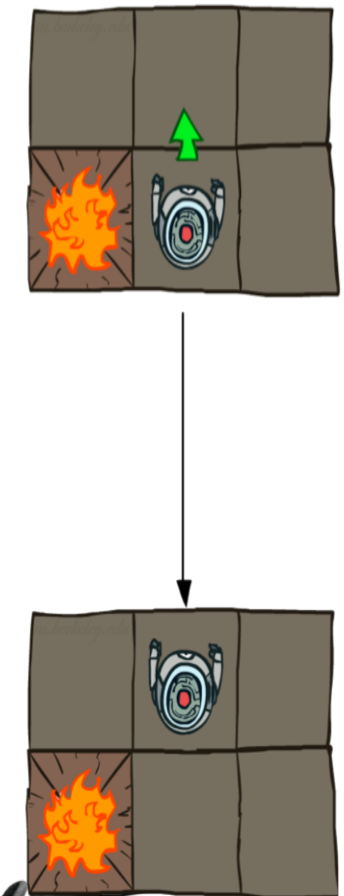
# Example: Grid World

- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- The agent receives rewards each time step
  - Small "living" reward each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action takes the agent in the desired direction (if there is no wall there)
  - 10% of the time, the action takes the agent to the direction perpendicular to the right; 10% perpendicular to the left.
  - If there is a wall in the direction the agent would have gone, agent stays put

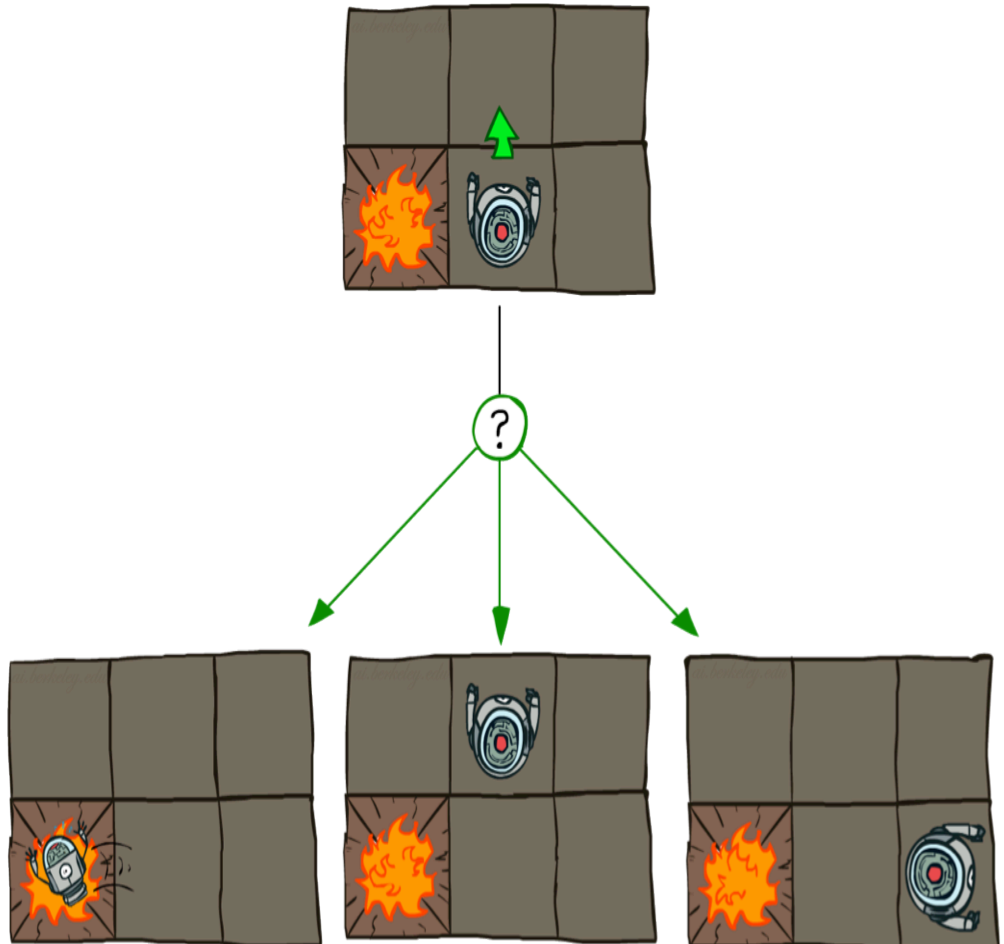


# Grid World Actions

Deterministic Grid World



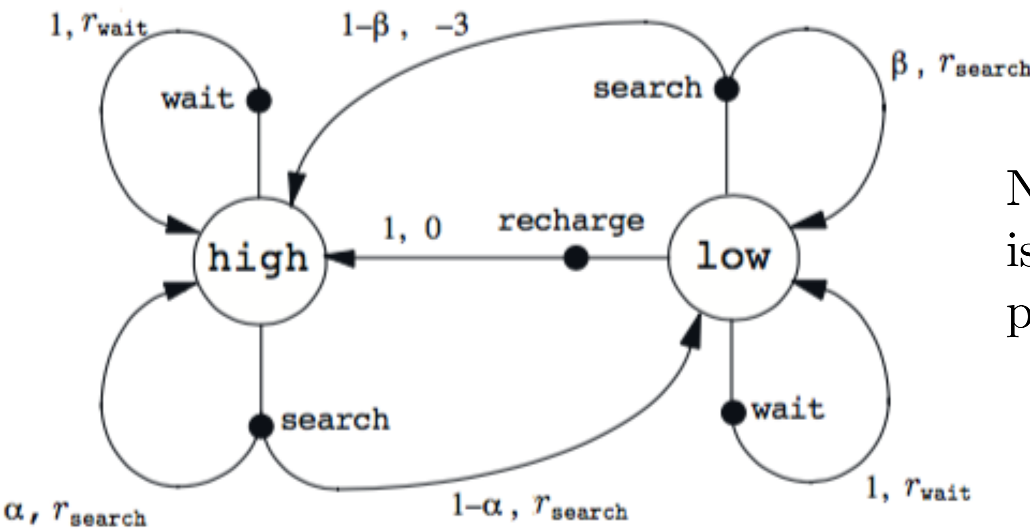
Stochastic Grid World



# Recycling robot

- At each step, robot has to decide whether it should: search for a can; wait for someone to bring it a can; go to home base and recharge. Searching is better but runs down the battery; if runs out of power while searching, has to be rescued.
- States are battery levels: high, low.
- Reward = number of cans collected.

| $s$ | $s'$ | $a$ | $p(s' \mid s, a)$ | $r(s, a, s')$ |
|---|---|---|---|---|
| high | high | search | $\alpha$ | $r_{\text{search}}$ |
| high | low | search | $1 - \alpha$ | $r_{\text{search}}$ |
| low | high | search | $1 - \beta$ | $-3$ |
| low | low | search | $\beta$ | $r_{\text{search}}$ |
| high | high | wait | 1 | $r_{\text{wait}}$ |
| high | low | wait | 0 | $r_{\text{wait}}$ |
| low | high | wait | 0 | $r_{\text{wait}}$ |
| low | low | wait | 1 | $r_{\text{wait}}$ |
| low | high | recharge | 1 | 0 |
| low | low | recharge | 0 | 0. |




Note: the "state" (robot's battery status) is a parameter of the agent itself, not a property of the physical environment

# Policies

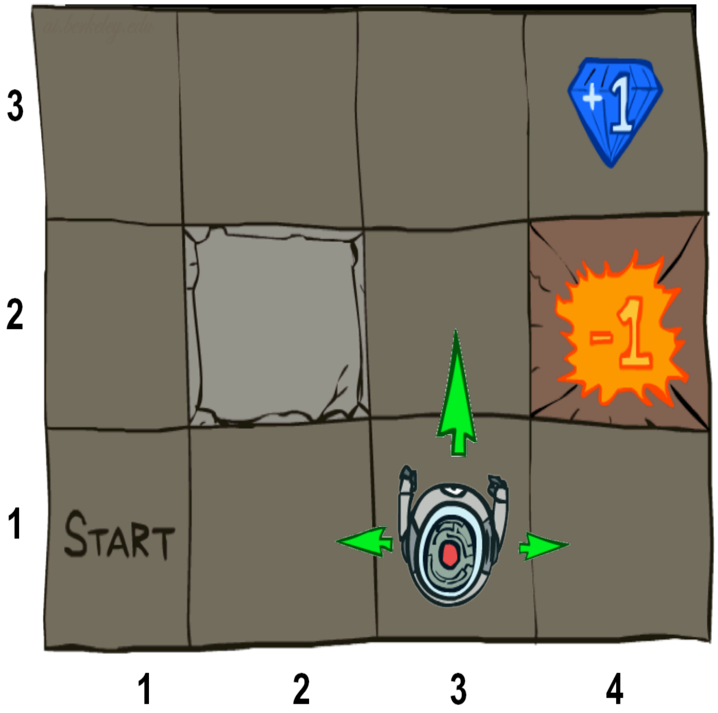
- In deterministic single-agent search problems, we wanted an optimal plan, or sequence of actions, from start to a goal

- In MDPs instead of plans, we have a **policy, a mapping from states to actions:** $\pi$: S $\rightarrow$ A
  - $\pi(s)$ specifies what action to take in each state $\rightarrow$ deterministic policy
  - An explicit policy defines a *reflex agent*

- A policy can also be *stochastic*, $\pi(s, a)$ specifies the probability of taking action a in state $s$ (in MDPs, if $R$ is deterministic, the *optimal* policy is deterministic)

# How Many Policies?

- How many non-terminal states?
- How many actions?
- How many deterministic policies over non-terminal states?
- 9, 4, $4^9$



# Utility of a Policy

- Starting from $s_0$, applying the policy π, generates a sequence of states $s_0, s_1, \ldots s_t$, and of rewards $r_0, r_1, \ldots r_t$
- For the (rational) decision-maker each sequence has an **utility** based on the *preferences* of the DM
- "Utility is an additive combination of the rewards"
- The utility, or *value* of a policy π starting in state $s_0$ is the expected utility over all the state sequences generated by the applying π

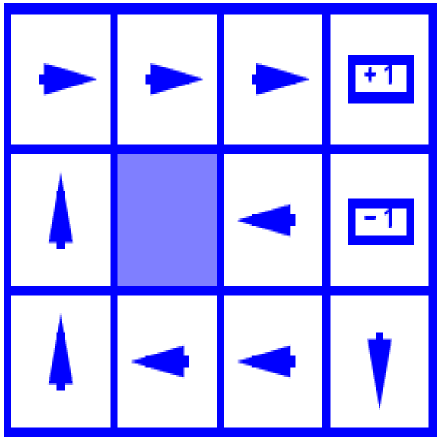
$$\sum_{\substack{\forall \text{ state sequences} \\ \text{starting from } s_0}} P^{\pi}(\text{sequence}) U(\text{sequence})$$

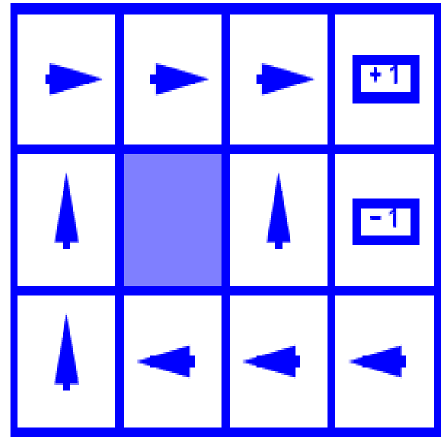
# Optimal Policies

- An optimal policy $\pi^*$ yields the maximal utility
- The maximal expected sum of rewards from following it starting from the initial state
- **Principle of maximum expected utility**: a rational agent should choose the action that maximizes its expected utility

# Optimal Policies

R(s) = -0.01

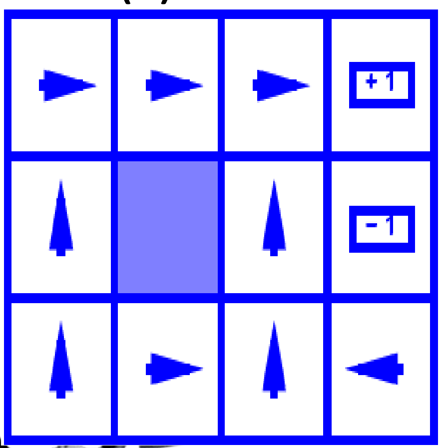


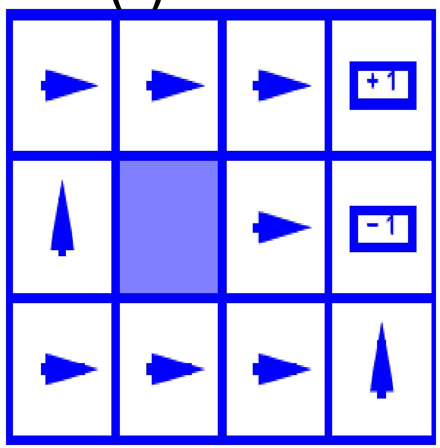

R(s) = -0.04

Balance between **risk** and **reward** changes depending on the value of R(s)

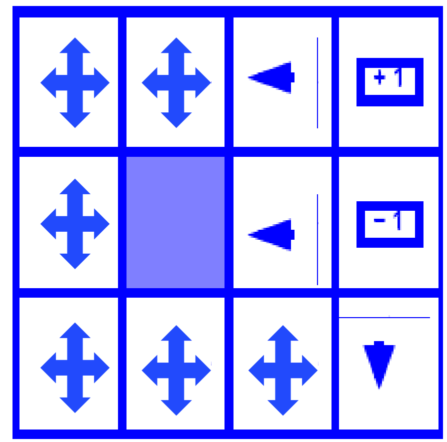
R(s) = -0.4




R(s) = -2.0

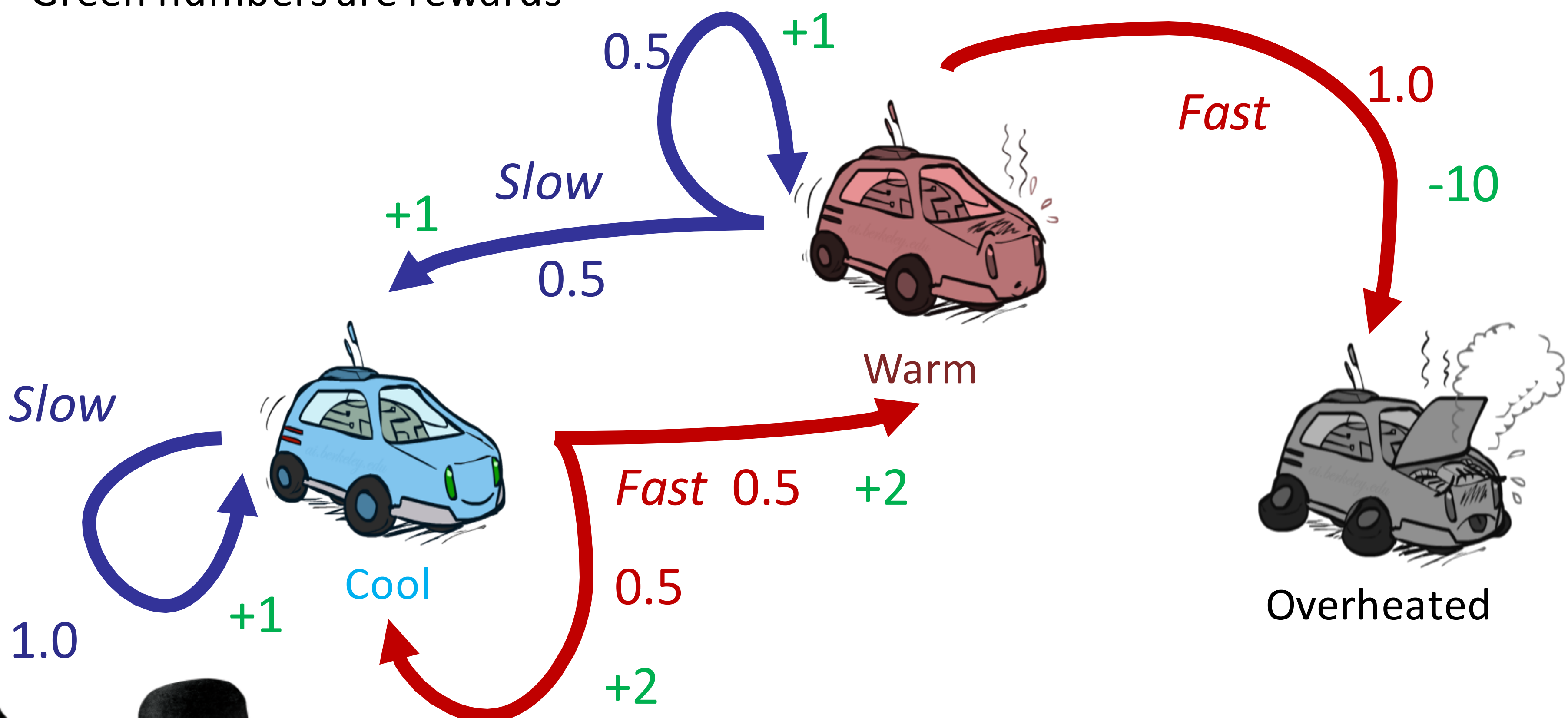



R(s) > 0

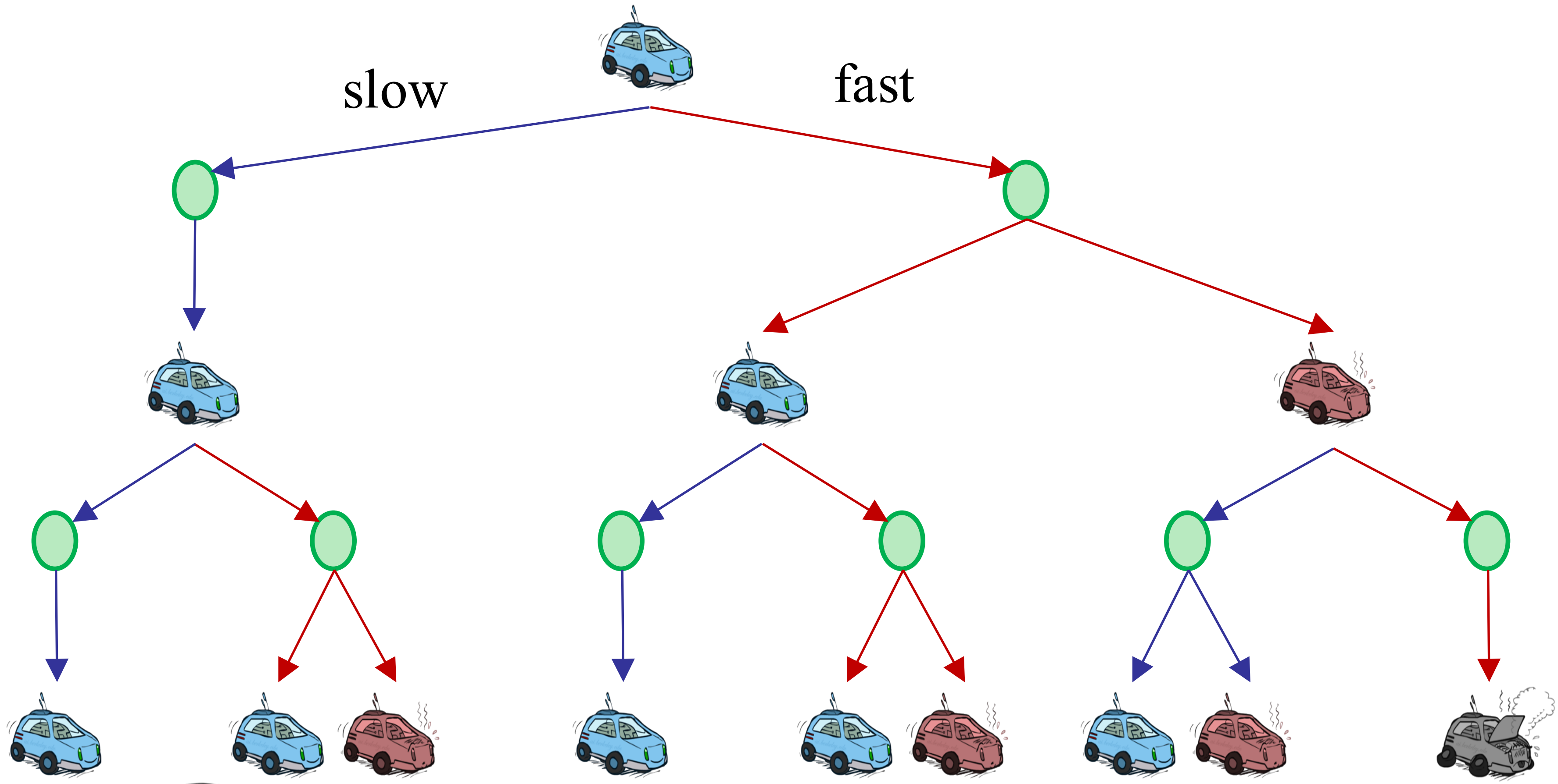


# Example: Racing

- A robot car wants to travel far, quickly
- Three states: Cool, Warm, Overheated
- Two actions: *Slow*, *Fast*
- Going faster gets double reward
- Green numbers are rewards

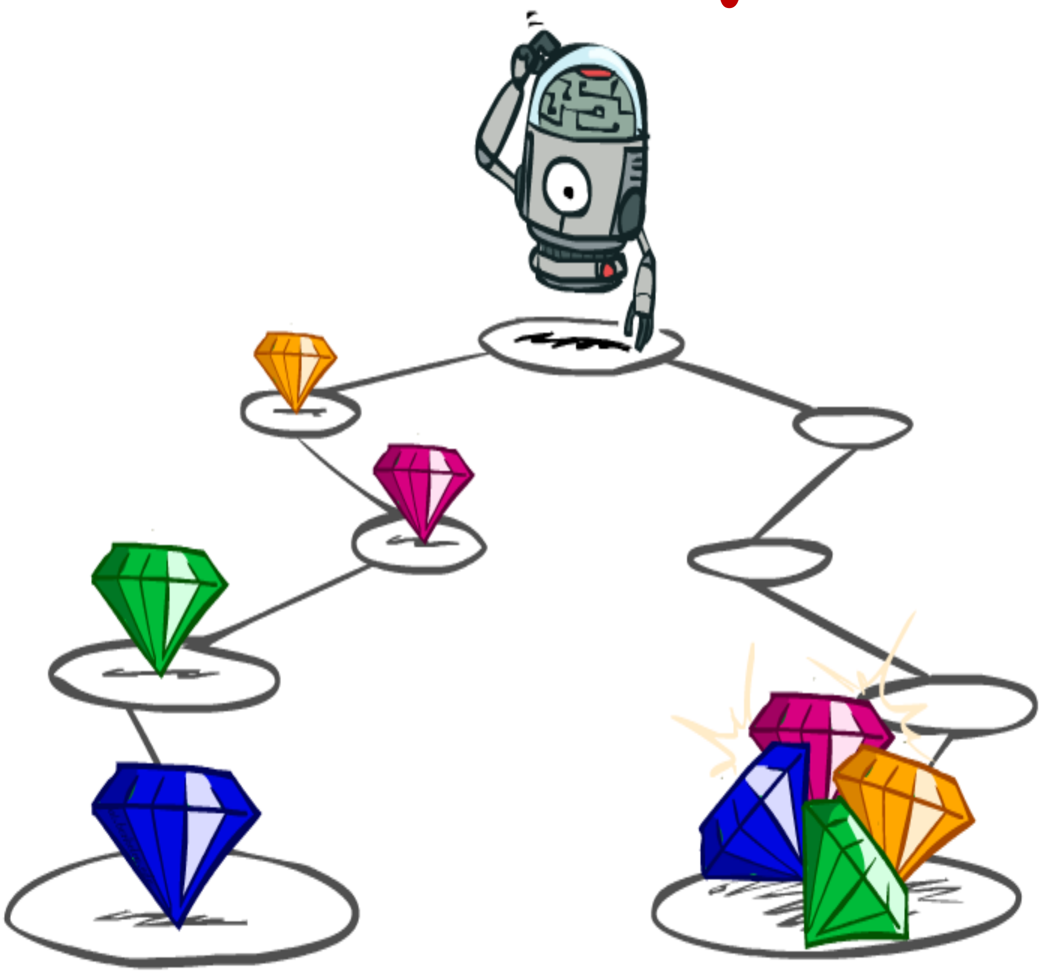


# Racing Search Tree




Slide adapted from Klein and Abbeel

# Utilities of Sequences



# Utilities of Sequences

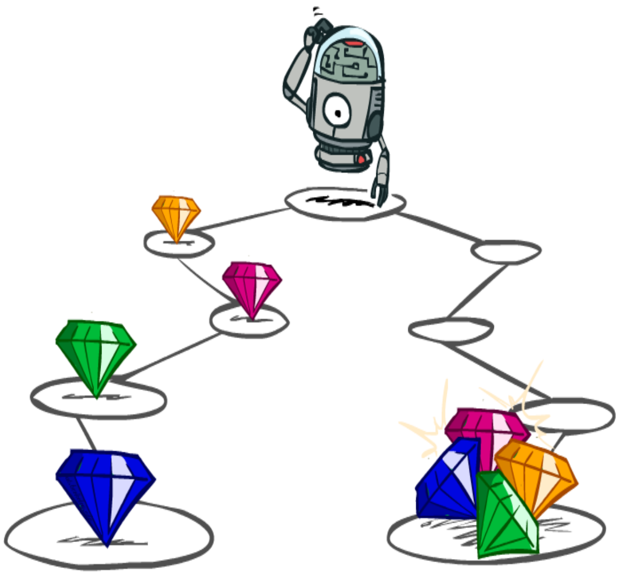
- What preferences should an agent have over reward sequences?

- More or less?

  [1, 2, 2] or [2, 3, 4]

- Now or later?

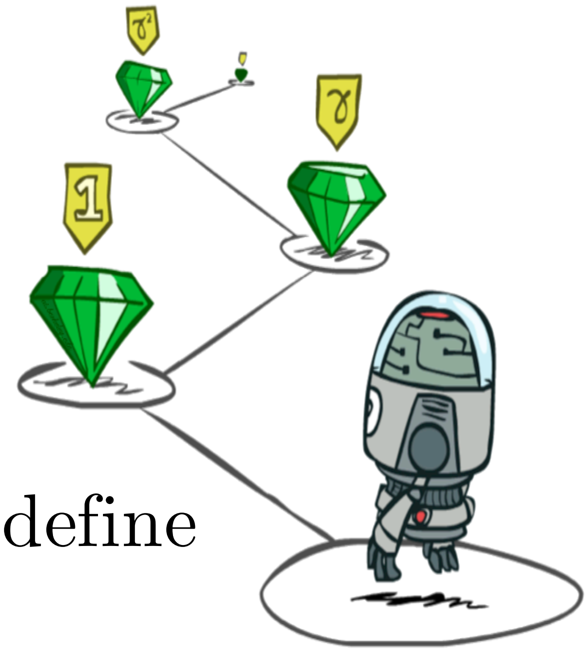
  [0, 0, 1] or [1, 0, 0]



# Stationary Preferences

- Theorem: if we assume *stationary preferences* between sequences:

$$[a_1, a_2, \ldots] \succ [b_1, b_2, \ldots]$$
$$\Updownarrow$$
$$[r, a_1, a_2, \ldots] \succ [r, b_1, b_2, \ldots]$$




- Then: there are only two ways to define utilities over sequences of rewards
  - Additive utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$
  - Discounted utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$

# What are Discounts?

- It's reasonable to prefer rewards now to rewards later
- Decay rewards exponentially

$1$

Worth Now

$\gamma$

Worth Next Step

$\gamma^2$

Worth In Two Steps

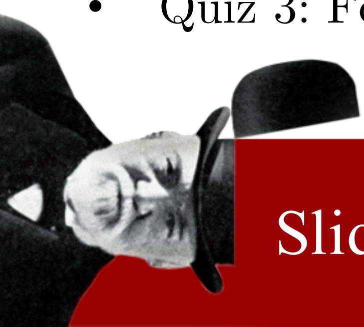
# DISCOUNTING

- Given:

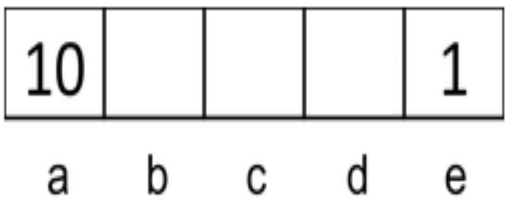
| | | | | |
|---|---|---|---|---|
| 10 | | | | 1 |
| a | b | c | d | e |

$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

  - Actions: East, West
  - Terminal states: a and e (end when reach one or the other)
  - Transitions: deterministic
  - Reward for reaching a is 10
  - reward for reaching e is 1, and the reward for reaching all other states is 0
- Quiz 1: For $\gamma = 1$, what is the optimal policy?
- Quiz 2: For $\gamma = 0.1$, what is the optimal policy for states b, c and d?
- Quiz 3: For which $\gamma$ are West and East equally good when in state d?

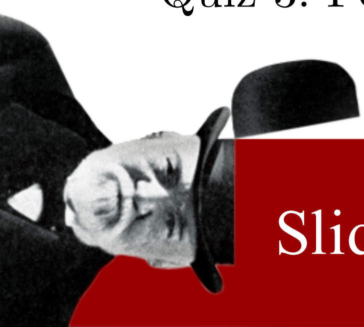


# DISCOUNTING

- Given: 

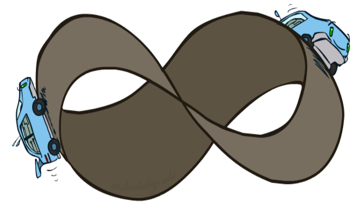
$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$
  - Actions: East, West
  - Terminal states: a and e (end when reach one or the other)
  - Transitions: deterministic
  - Reward for reaching a is 10
  - reward for reaching e is 1, and the reward for reaching all other states is 0
- Quiz 1: For $\gamma = 1$, what is the optimal policy?
  - In all states, Go West (towards a)
- Quiz 2: For $\gamma = 0.1$, what is the optimal policy for states b, c and d?
  - b=W, c=W, d=E
- Quiz 3: For which $\gamma$ are West and East equally good when in state d?

$$\gamma = \sqrt{(1/10)}$$



# Infinite Utilities?!

- Problem: What if the process lasts forever? Do we get infinite rewards?



- Solutions:
  - Finite horizon: (similar to depth-limited search)
    - Terminate episodes after a fixed T steps (e.g. life)
    - Gives nonstationary policies ($\pi$ depends on time left)
  - Discounting: use $0 < \gamma < 1$

$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

    - Smaller $\gamma$ means smaller "horizon" – shorter term focus
  - Absorbing state: guarantee that for every policy, a terminal state will eventually be reached (like "overheated" for racing)

# Recap: Defining MDPs

- Markov decision processes:
  - Set of states $S$
  - Start state $s_0$
  - Set of actions $A$
  - Transitions $\mathbf{P}(s'|s,a)$ (or $\mathbf{T}(s,a,s')$)
  - Rewards $R(s,a,s')$ (and discount $\gamma$)
- MDP quantities so far:
  - Policy $\pi$ = Choice of action for each state
  - Utility/Value = sum of (discounted) rewards
  - Optimal policy $\pi^*$ = Best choice, that max Utility