AIFA: Stochastic Planning MDP

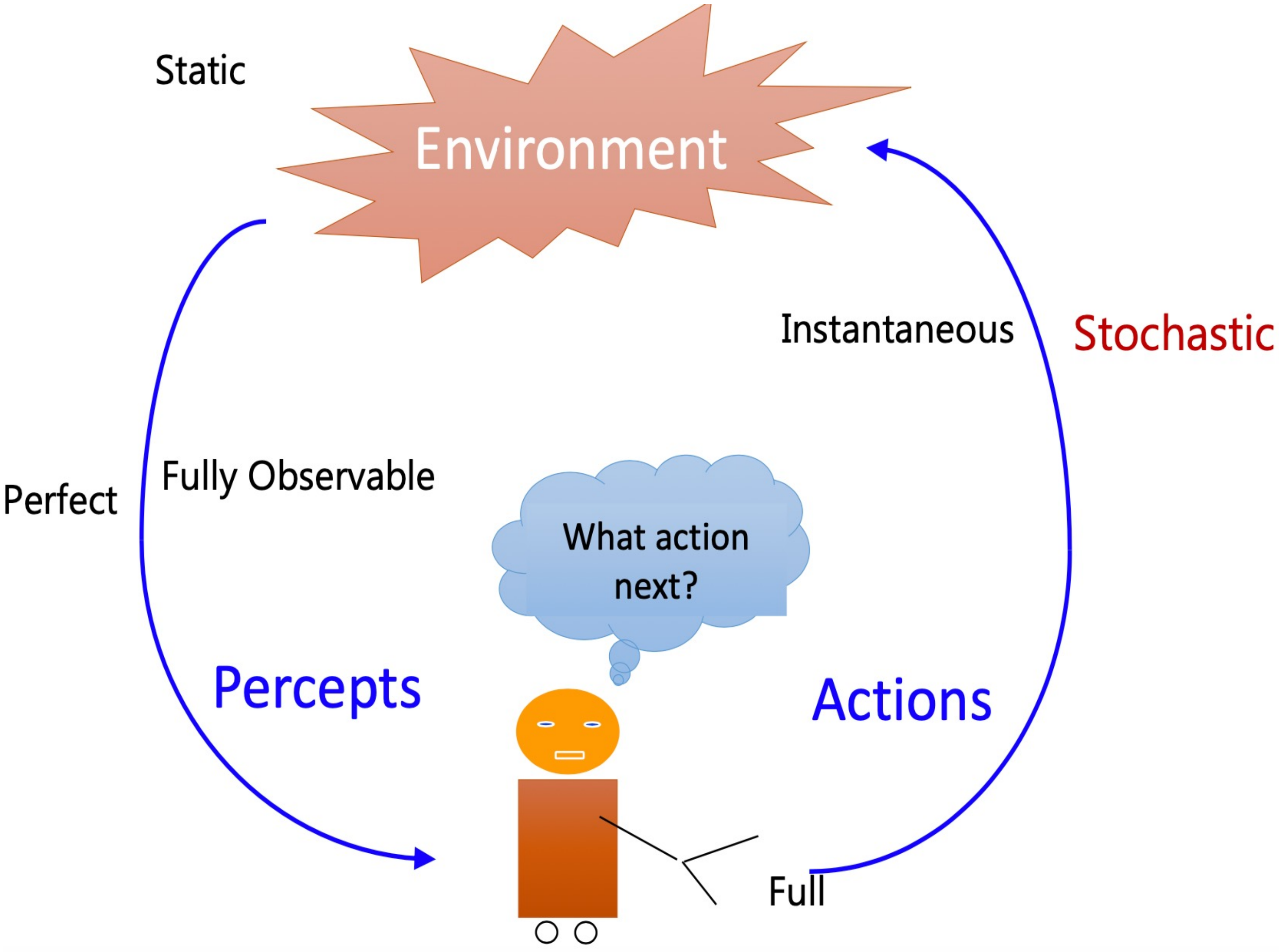
08/04/2024

Koustav Rudra

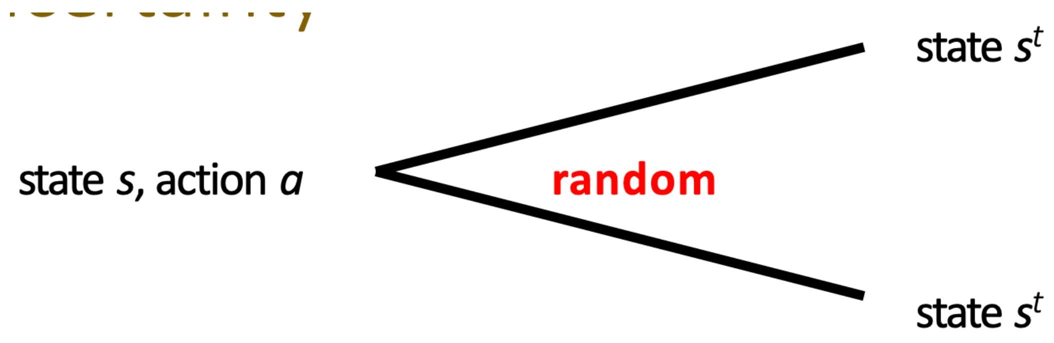
Markov Decision Processes

- Value Iteration
- Policy Iteration

Planning under Uncertainty



Uncertainty



Randomness:

- could be caused by limitations of the sensors and actuators of the robot
- could be caused by market forces or nature

…

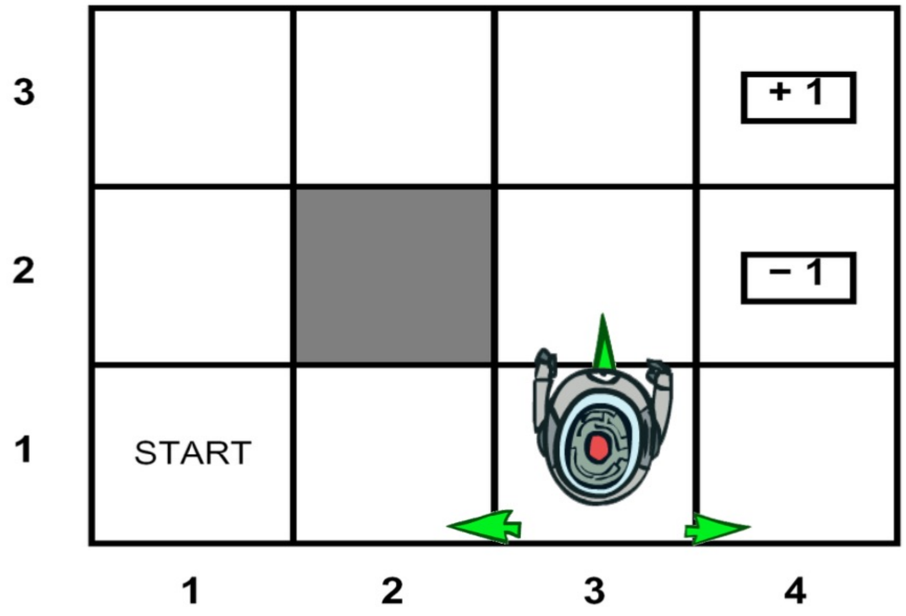
- Robotics: decide where to move, but actuators can fail, hit unseen obstacles, etc.
- Resource allocation: decide what to produce, don't know the customer demand for various products
- Agriculture: decide what to plant; don't know weather and thus crop yield

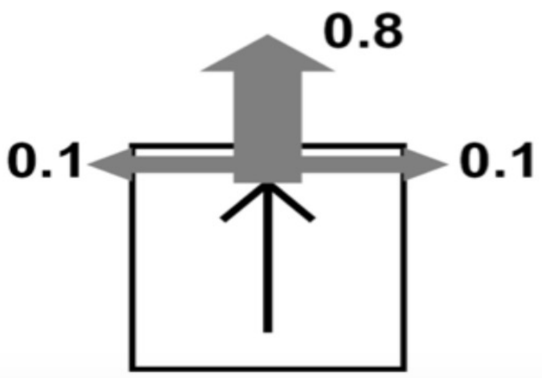
Example: Grid World

- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put

- The agent receives rewards
  - Small "living" reward each step
  - Big rewards come at the end

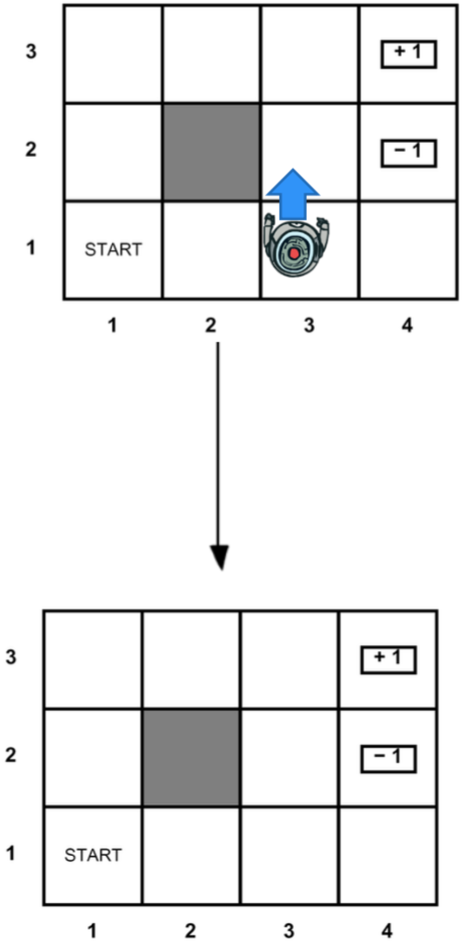
- Goal: maximize sum of rewards



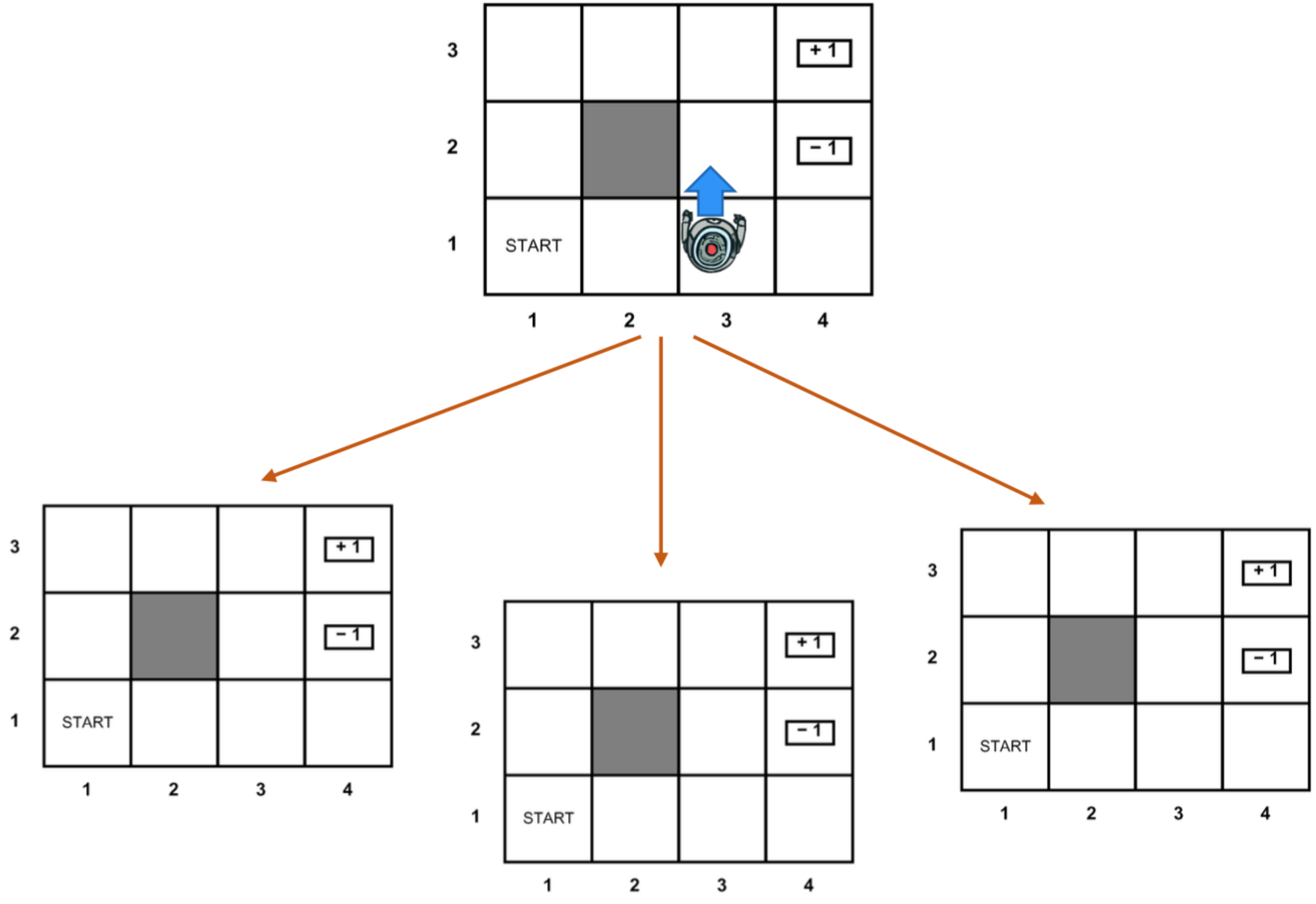


Grid World Actions

Deterministic Grid World

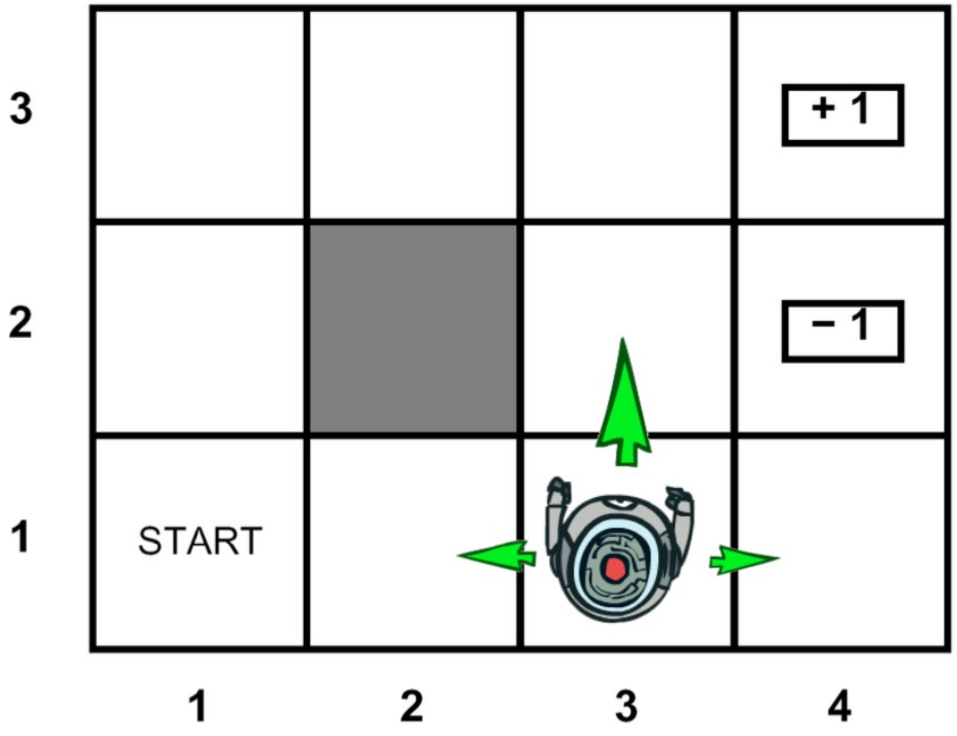


Stochastic Grid World



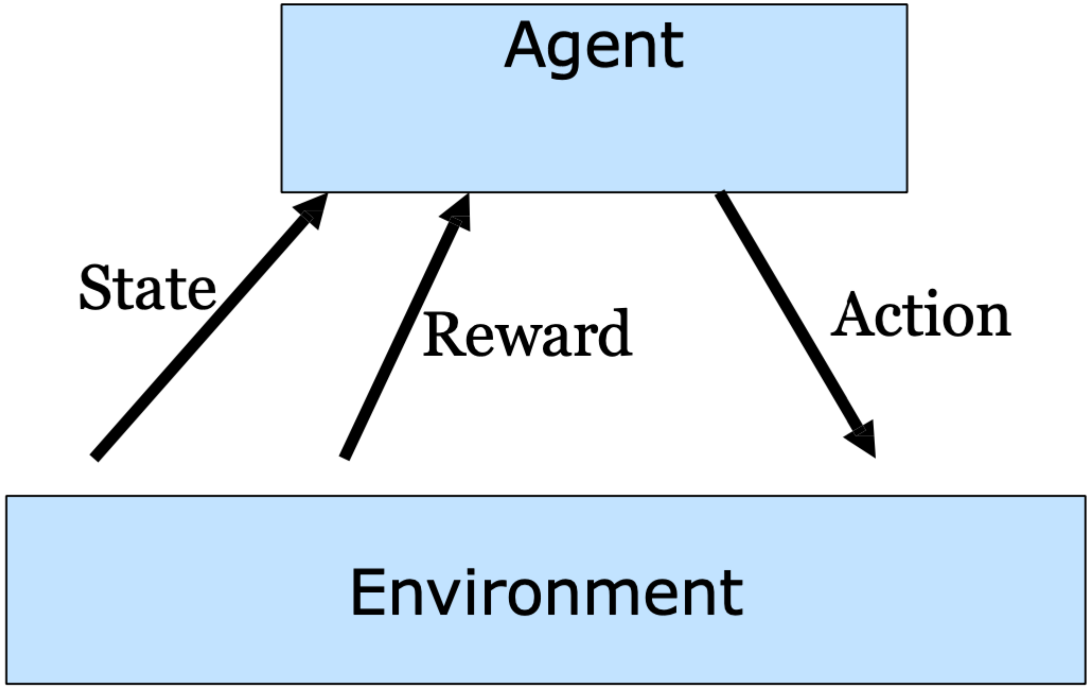
Markov Decision Process

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s, a, s')
    - Probability that a from s leads to s', i.e., P(s'|s, a)
    - Also called the model or the dynamics
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state
  - Maybe a terminal state



Markov Decision Process

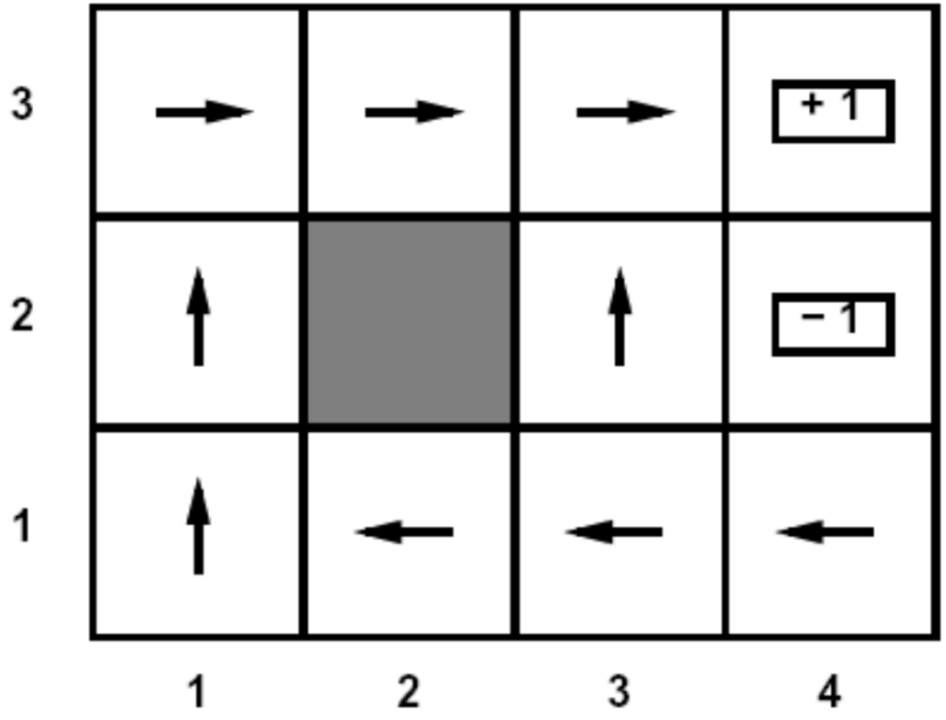
- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s, a, s')
    - Probability that a from s leads to s', i.e., P(s'|s, a)
    - Also called the model or the dynamics
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state
  - Maybe a terminal state



Goal: Learn to choose actions that maximize reward

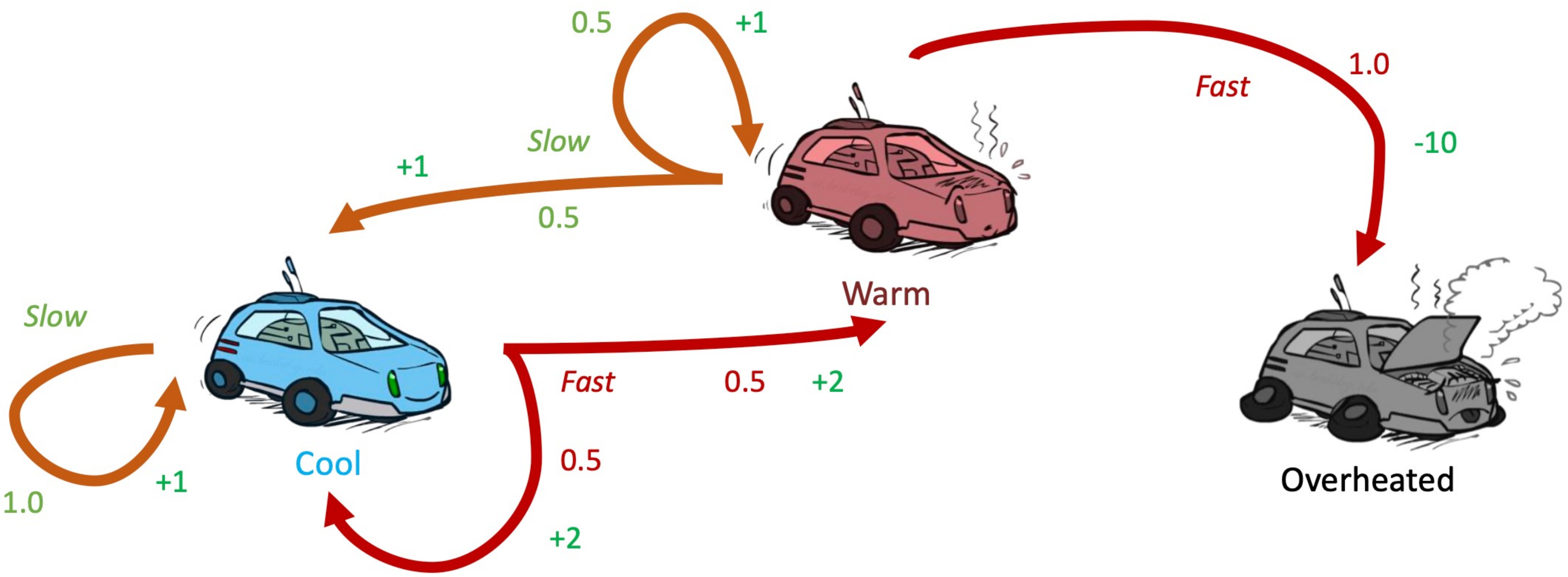
Solution to MDP: Policies

- For MDPs, we want an optimal policy $\pi^*: S \to A$
- A policy π gives an action for each state
- An optimal policy is one that maximizes expected utility if followed

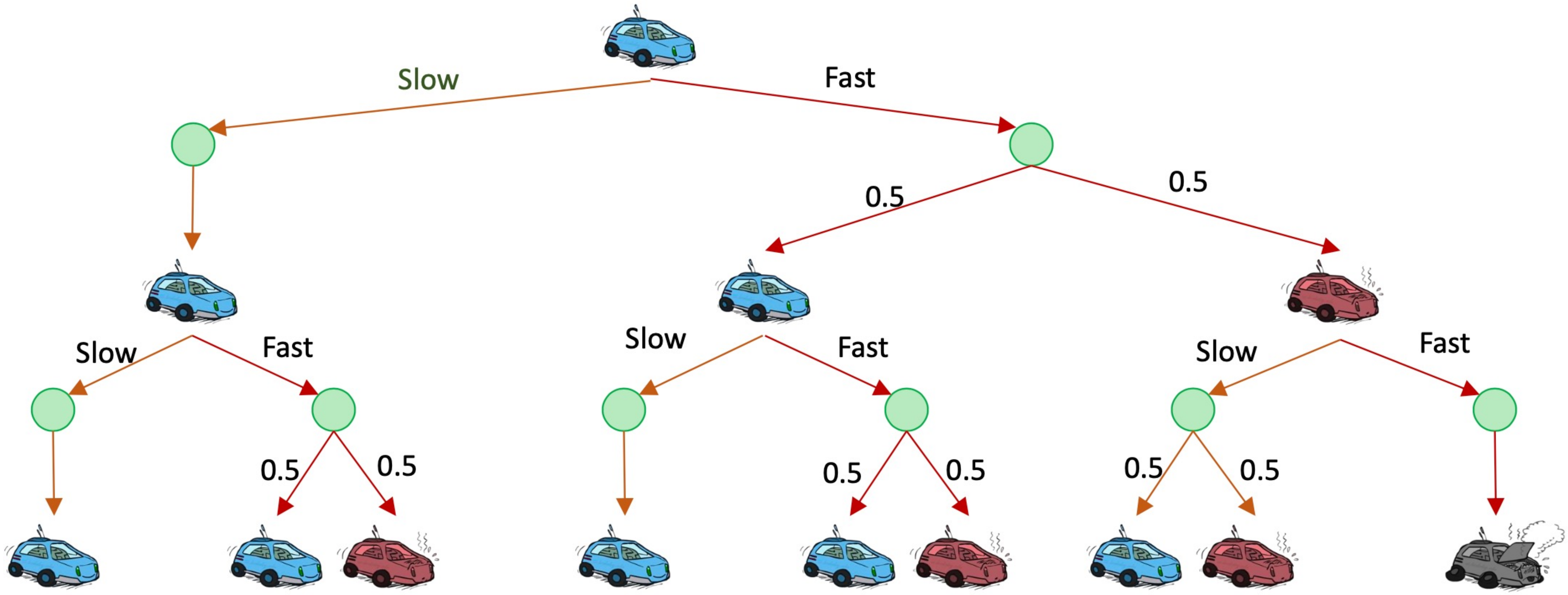


Example: Racing

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

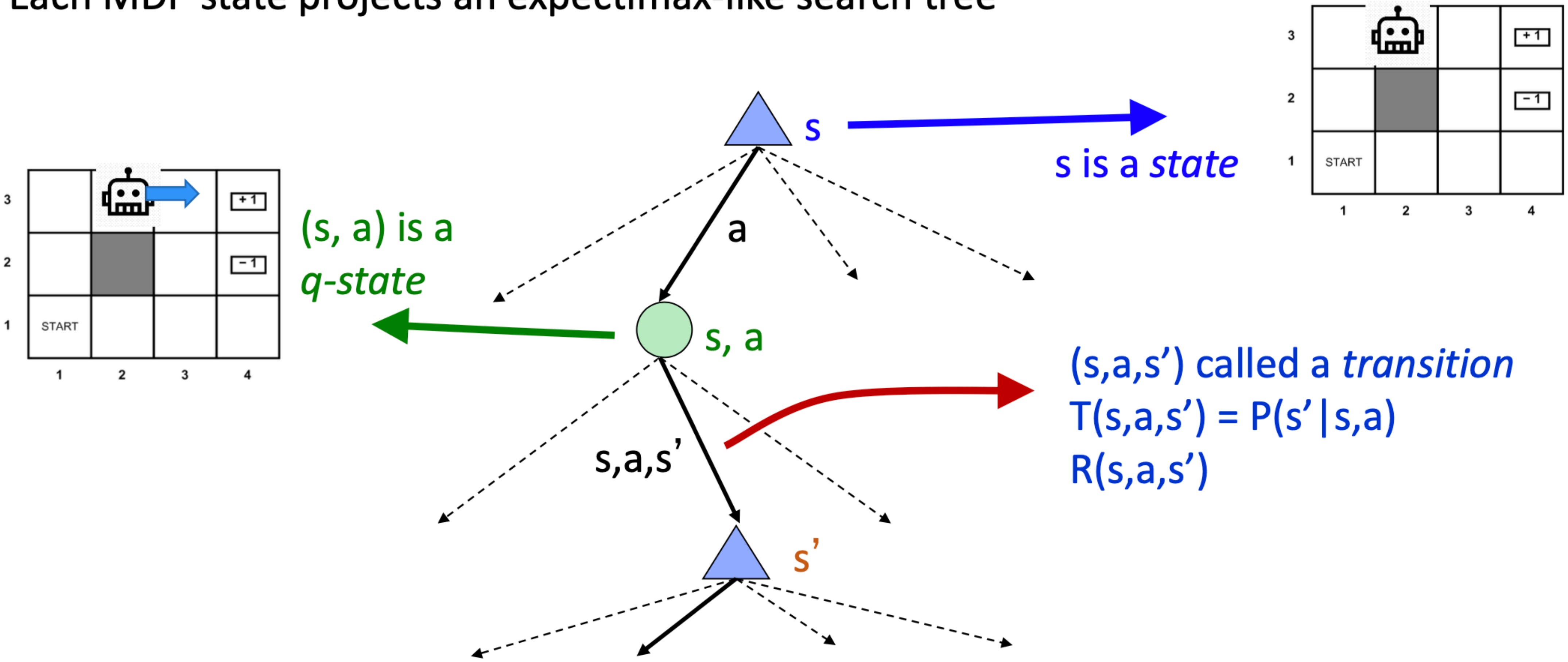


Racing car search tree



MDP search trees

Each MDP state projects an expectimax-like search tree



Utilities

Two ways to define utilities

- Additive utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$

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

$0 \leq \gamma \leq 1$: discount factor

What is a solution?

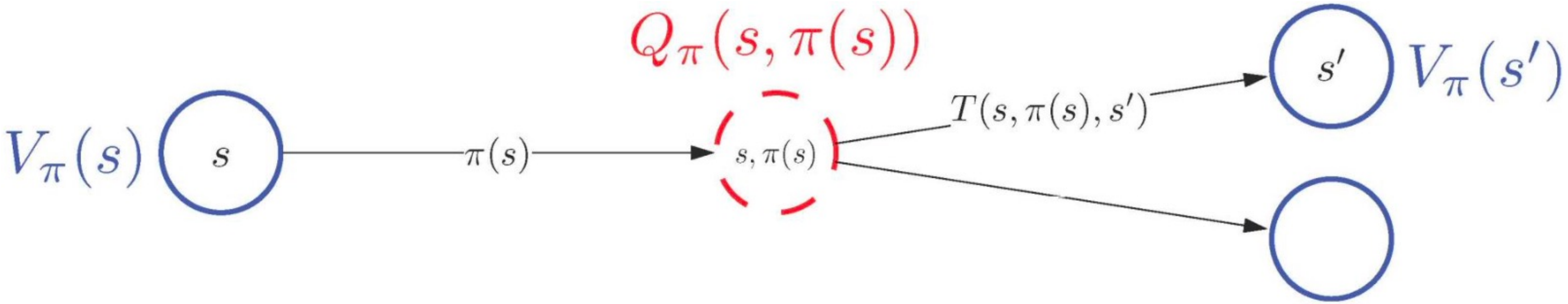
- A policy π is a mapping from each state s ∈ States to an action a ∈ Actions(s)

- Evaluating a policy
  - Following a policy yields a random path
  - The utility of a policy is the (discounted) sum of the rewards on the path (this is a random variable).
  - The value of a policy at a state is the expected utility.

Policy Evaluation

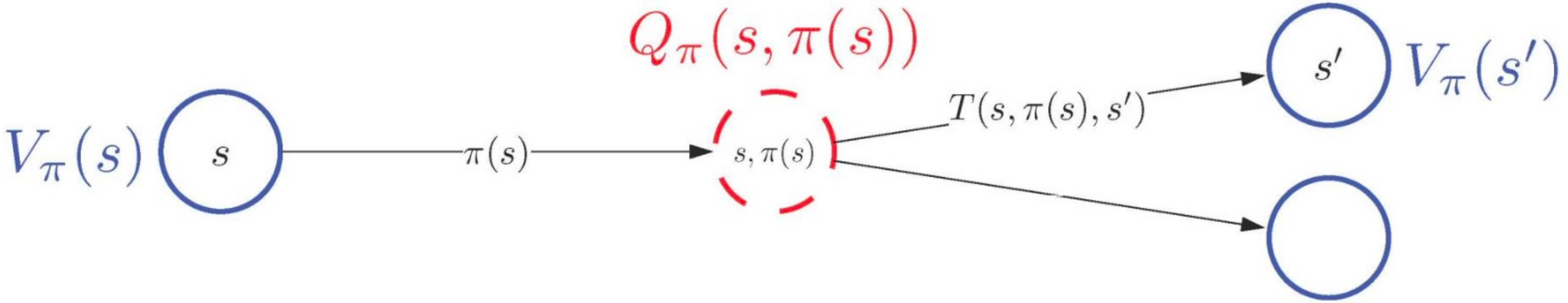
- Definition: value of a policy
  - Let $V_\pi(s)$ be the expected utility received by following policy π from state s

- Definition: Q-value of a policy
  - Let $Q_\pi(s, a)$ be the expected utility of taking action a from state s, and then following policy π



Policy Evaluation

- Plan: define recurrences relating value and Q-value



$$V_\pi(s,a) = \begin{cases} 0 \text{ } if \text{ } isEnd(s) \\ Q_\pi(s, \pi(s)) \text{ } otherwise \end{cases}$$

$$Q_\pi(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_\pi(s')]$$

Policy Evaluation

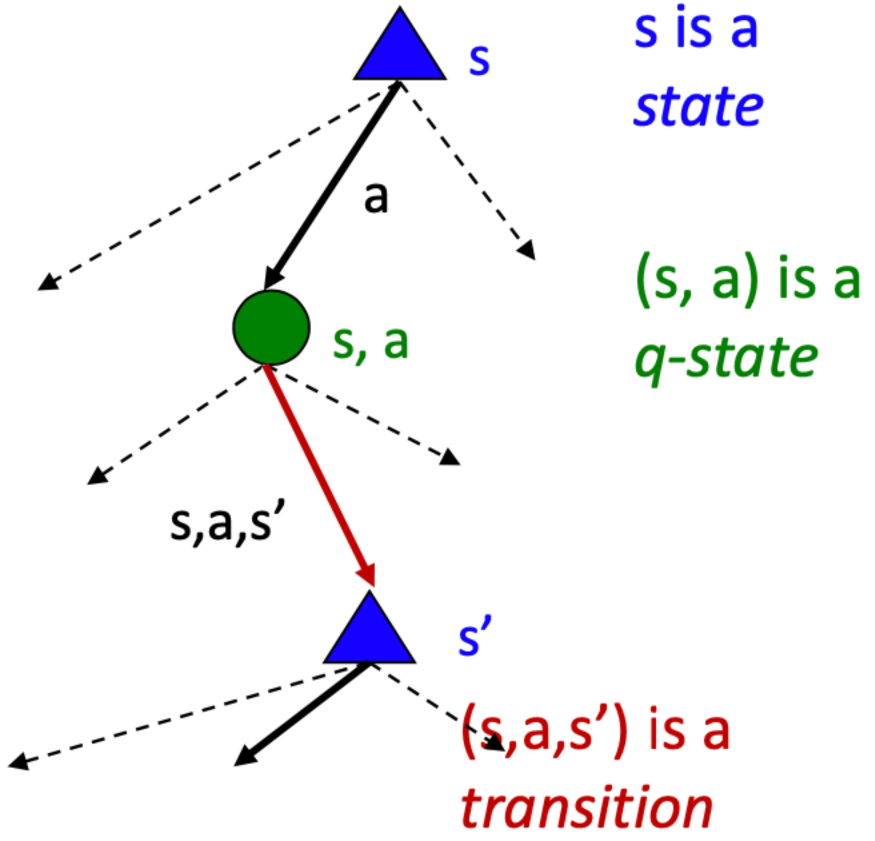
- Iterative algorithm: Start with arbitrary policy values and repeatedly apply recurrences to converge to true values
- Initialize $V_\pi^0(s) \leftarrow 0$ for all states s
- For iteration t = 1, … , t_{PE}
    - For each state s
        - $V_\pi^{(t)}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_\pi^{(t-1)}(s')]$
- How many iterations t_{PE}?
    - Repeat until values do not change much
    - $\max_s \left| V_\pi^{(t)}(s) - V_\pi^{(t-1)}(s) \right| \leq \epsilon$

Optimal Quantities

- The value (utility) of a state s:
  - $V^*(s) =$
    $expected\ utility\ starting\ in\ s\ and\ acting\ optimally$

- The value (utility) of a q-state (s,a):
  - $Q^*(s,a) =$
    $expected\ utility\ starting\ out\ having\ taken\ action\ a$
    $from\ state\ s\ and\ thereafter\ acting\ optimally$

- The optimal policy:
  - $\pi^*(s)$ = optimal action from state s



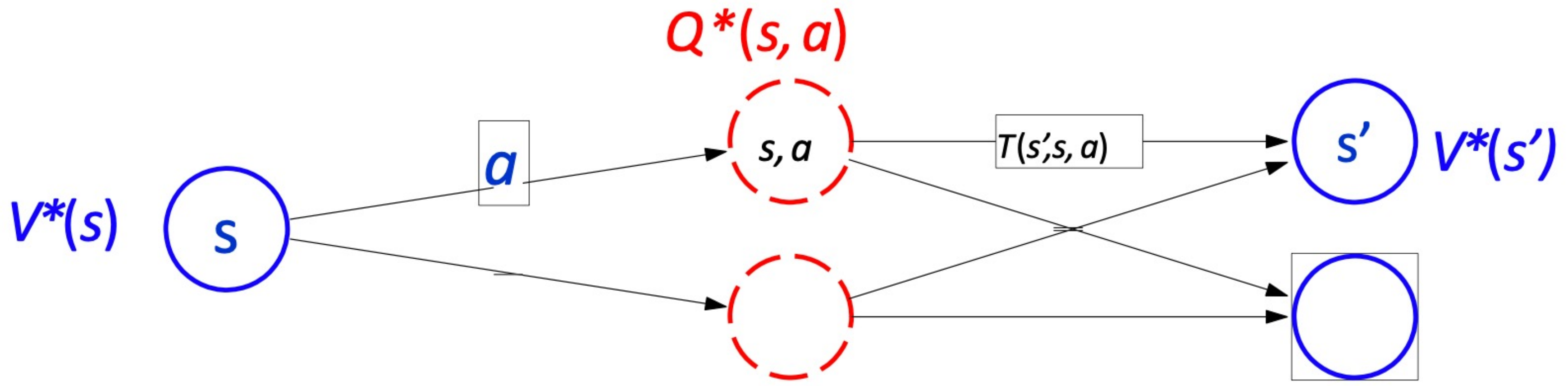
Value Function

- Value function for a policy $\pi: S \rightarrow A$
  - $V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s')$

- Optimum value function
  - $V^{*}(s) = \max_{\pi} V^{\pi}(s)$

- $Q^{\pi}(s, a)$: the expected utility of taking action a from state s, and then following policy π.

Optimal Values and Q-values

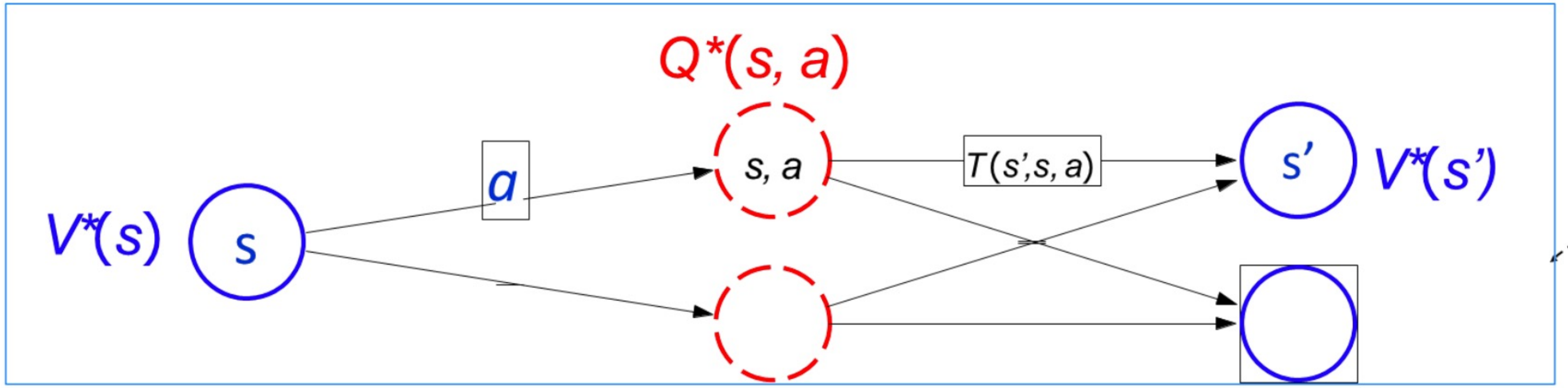


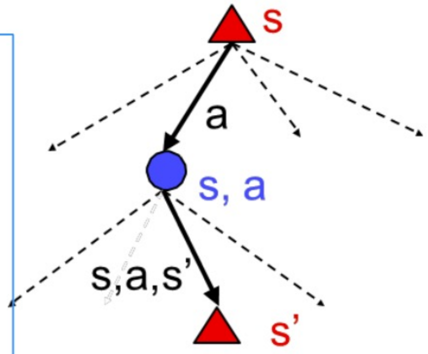
$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a Q^*(s,a)$$

The Bellman Equations

- Definition of "optimal utility" leads to a simple one-step lookahead relationship amongst optimal utility values:





$$V^*(s) = \max_a Q^*(s,a)$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')\left[R(s,a,s') + \gamma V^*(s')\right]$$

Value Iteration

1. For each state *s*, initialize *V*(*s*) := 0.
2. **for** until convergence **do**
3. For every state, update

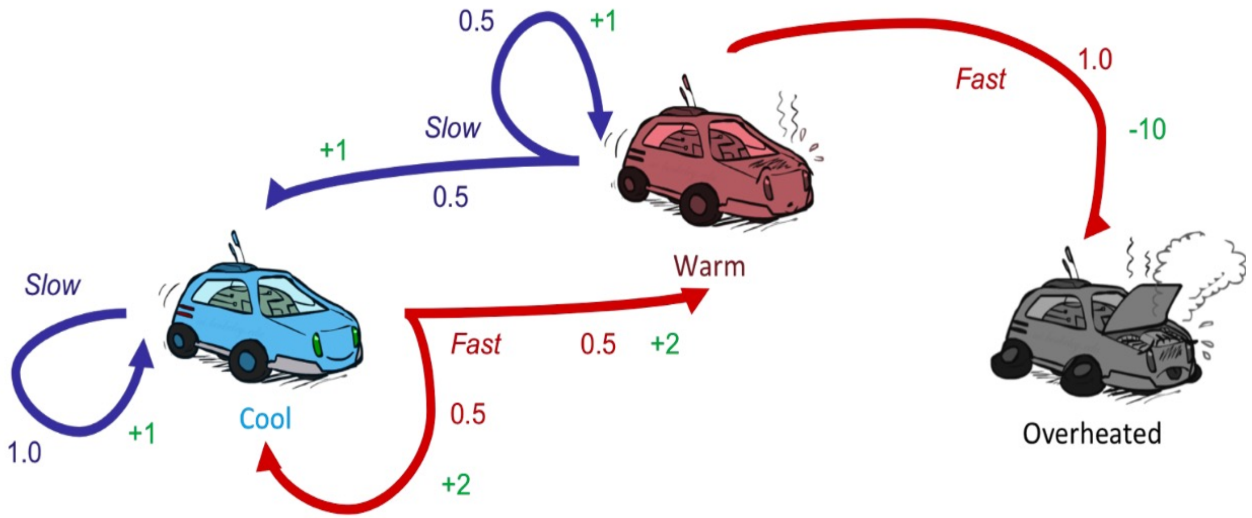
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

Complexity of each iteration: $O(S^2A)$

- Theorem: will converge to unique optimal values
    - Basic idea: approximations get refined towards optimal values
    - Policy may converge long before values do

Example: Value Iteration

V_2			
V_1	S: 1 F: .5*2+.5*2=2		
V_0	0	0	0

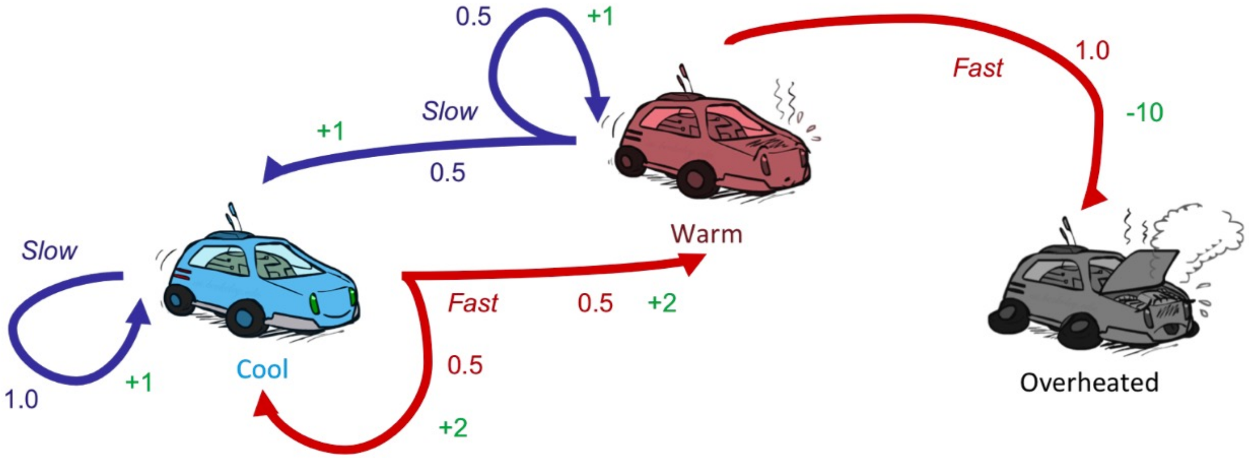


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

Example: Value Iteration

V_2			
V_1	2	S: .5*1+.5*1=1 F: -10	
V_0	0	0	0

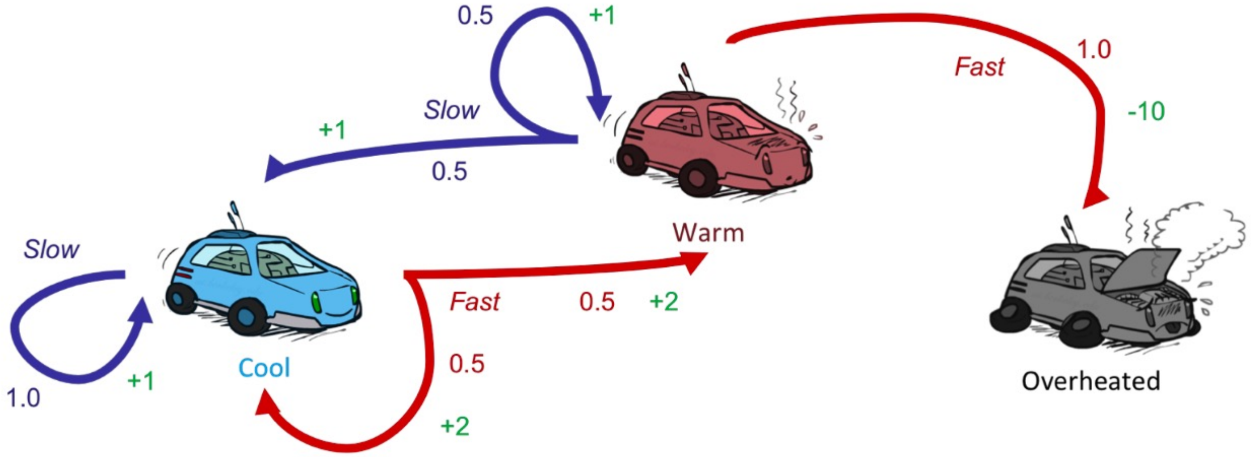


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

Example: Value Iteration

V_2			
V_1	2	1	0
V_0	0	0	0

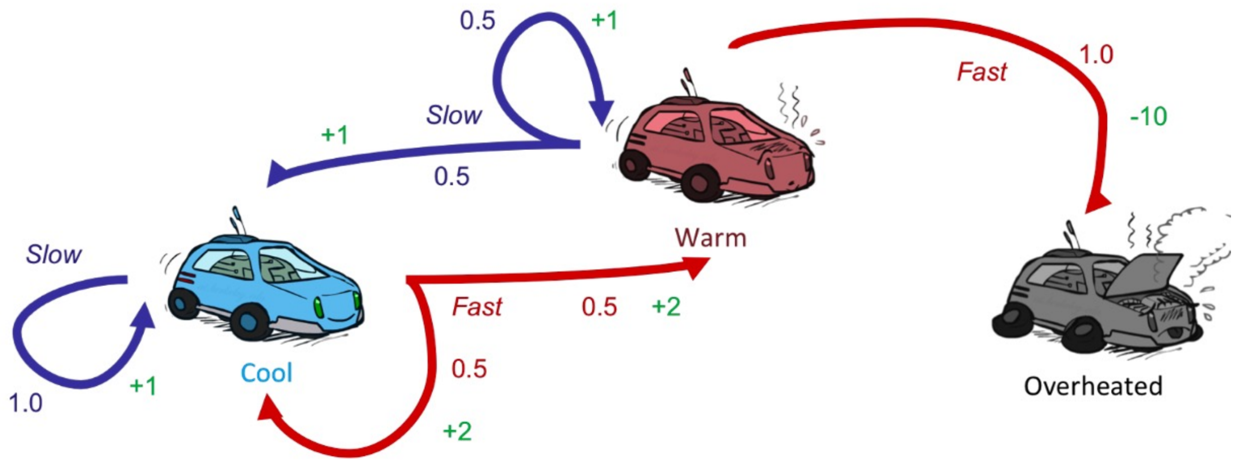


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

Example: Value Iteration

V_2	S: 1+2=3 F: .5*(2+2)+.5*(2+1)=3.5		
V_1	2	1	0
V_0	0	0	0

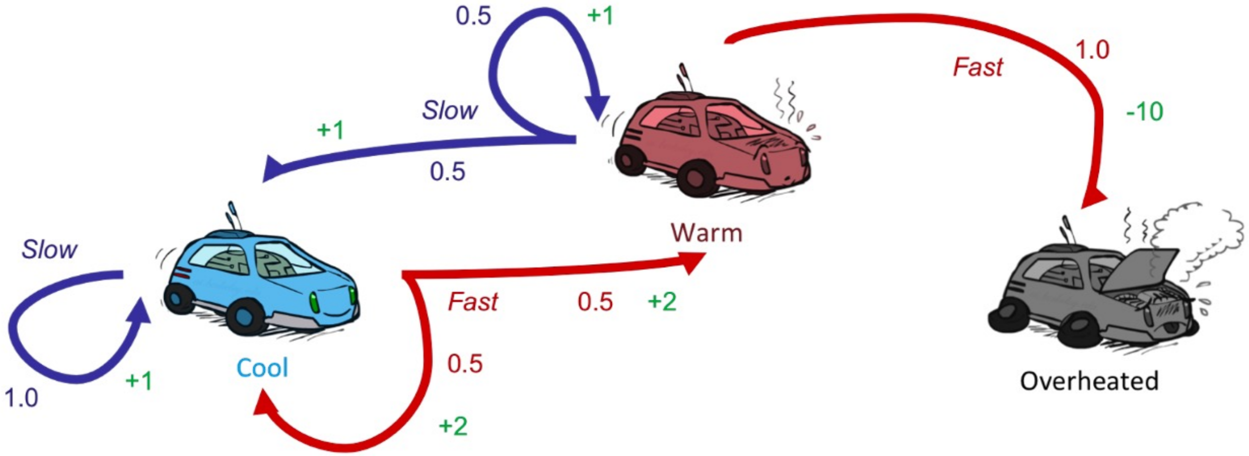


Assume no discount!

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s') \right]$$

Example: Value Iteration

V_2	3.5	2.5	0
V_1	2	1	0
V_0	0	0	0



Assume no discount!

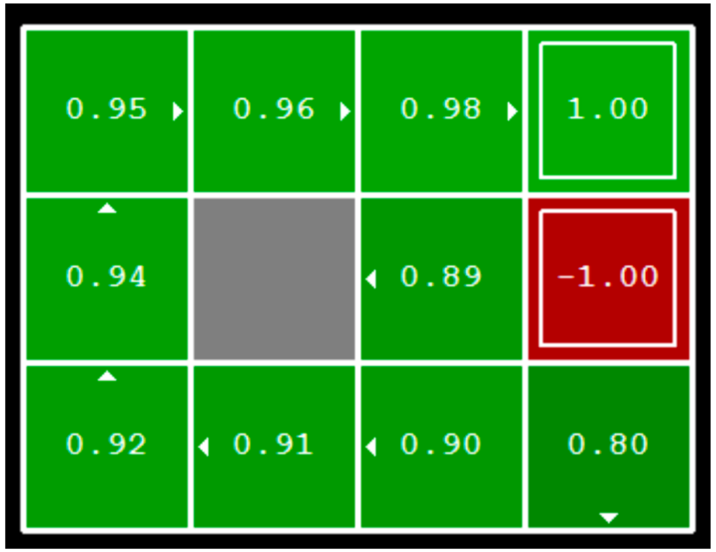
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

Computing Actions from Values

- Let's imagine we have the optimal values V*(s)

- How should we act?

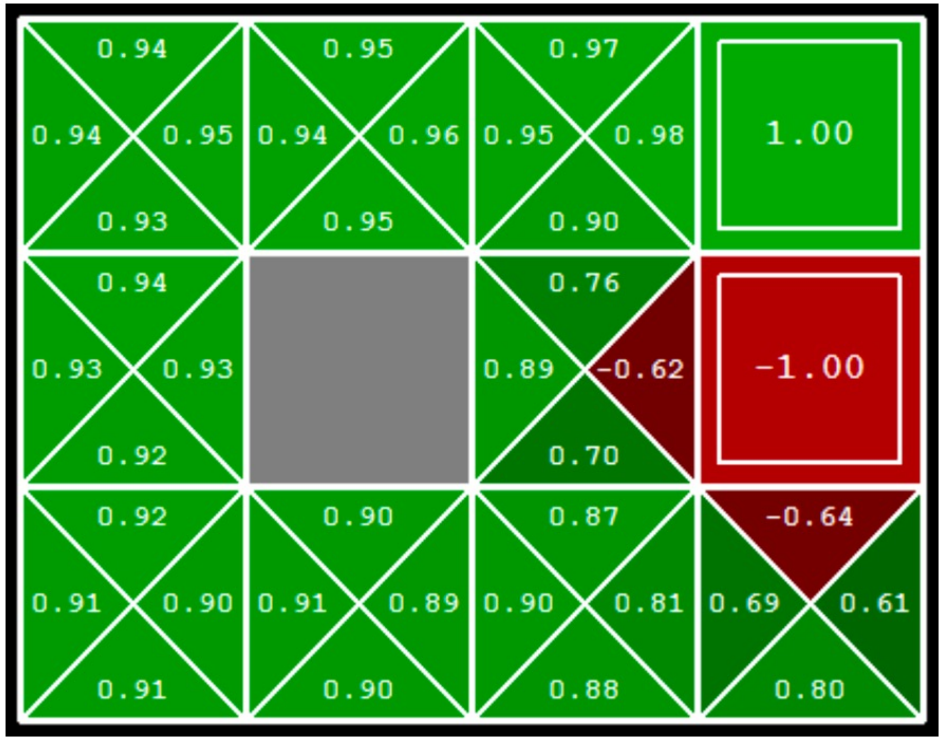
$$\pi^*(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$



- This is called policy extraction, since it gets the policy implied by the values

Computing actions from Q-values

- Let's imagine we have the optimal q-values:



- How should we act?
    - Completely trivial to decide!

$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

- Important lesson: actions are easier to select from q-values than values!

Thank You