CSE 573: Artificial Intelligence

Winter 2019

Hanna Hajishirzi

Markov Decision Processes

slides from

Dan Klein, Stuart Russell, Andrew Moore, Dan Weld, Pieter Abbeel, Luke Zettelmoyer

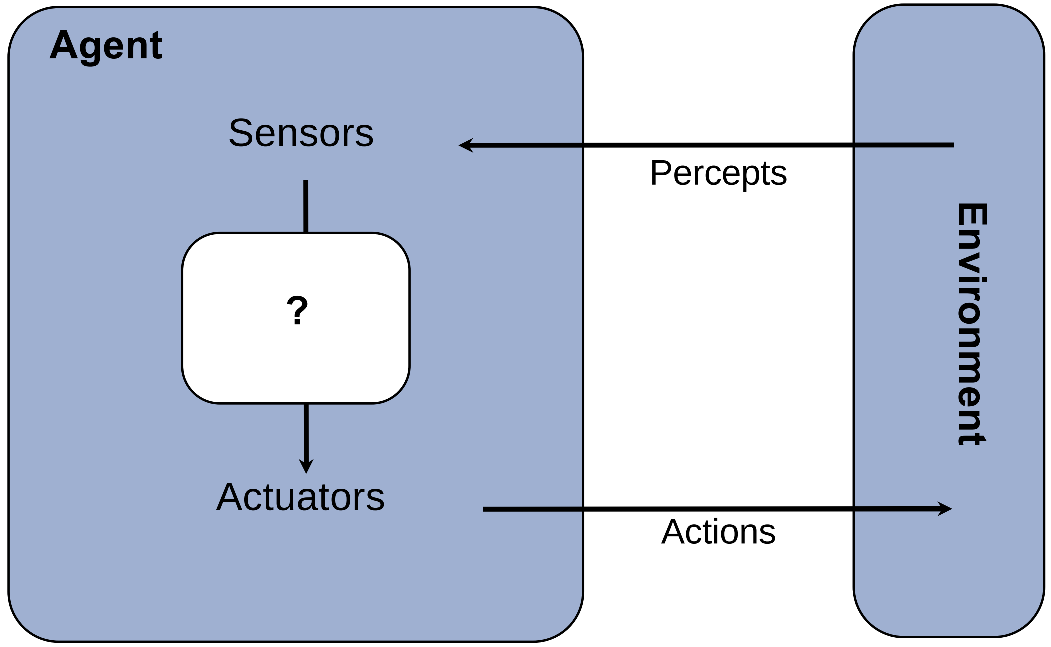
Review and Outline

- Adversarial Games
  - Minimax search
  - α-β search
  - Evaluation functions
  - Multi-player, non-0-sum
- Stochastic Games
  - Expectimax

  - Markov Decision Processes
  - Reinforcement Learning

Agents vs. Environment

- An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that ***maximize its utility function.***

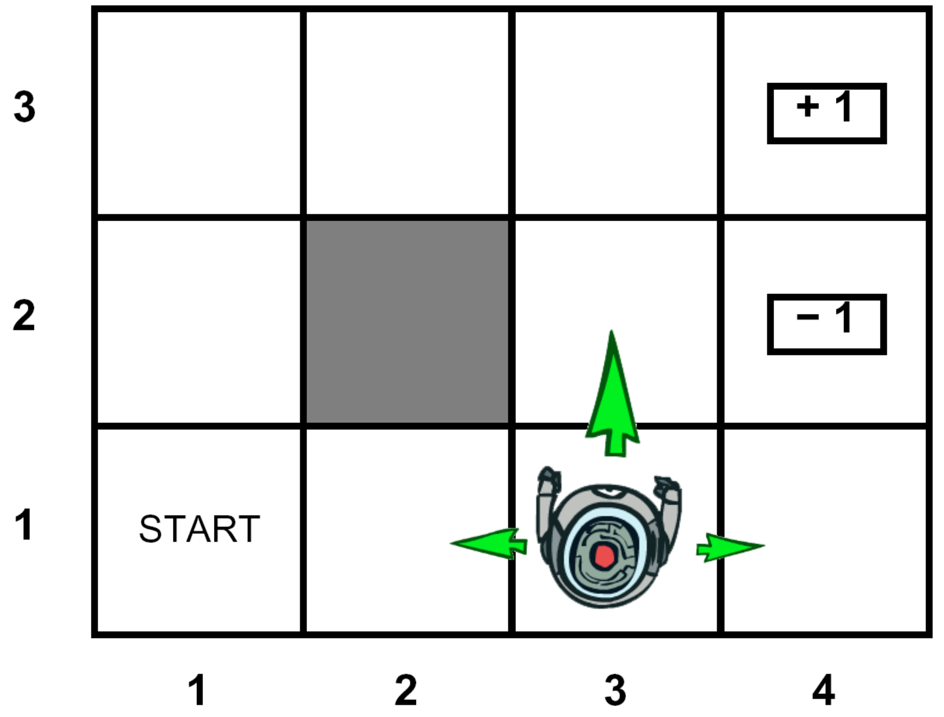


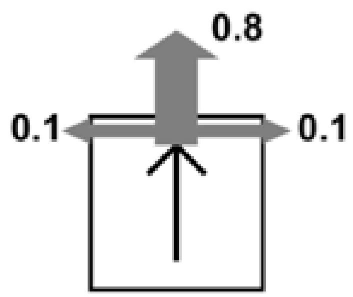
Deterministic *vs.* ***stochastic***

Fully observable *vs.* partially observable

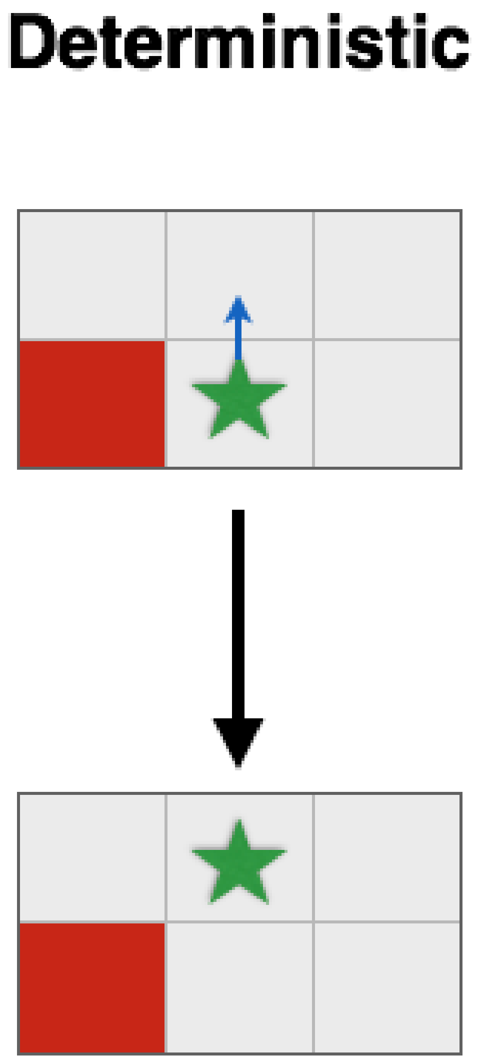
Example: Grid World

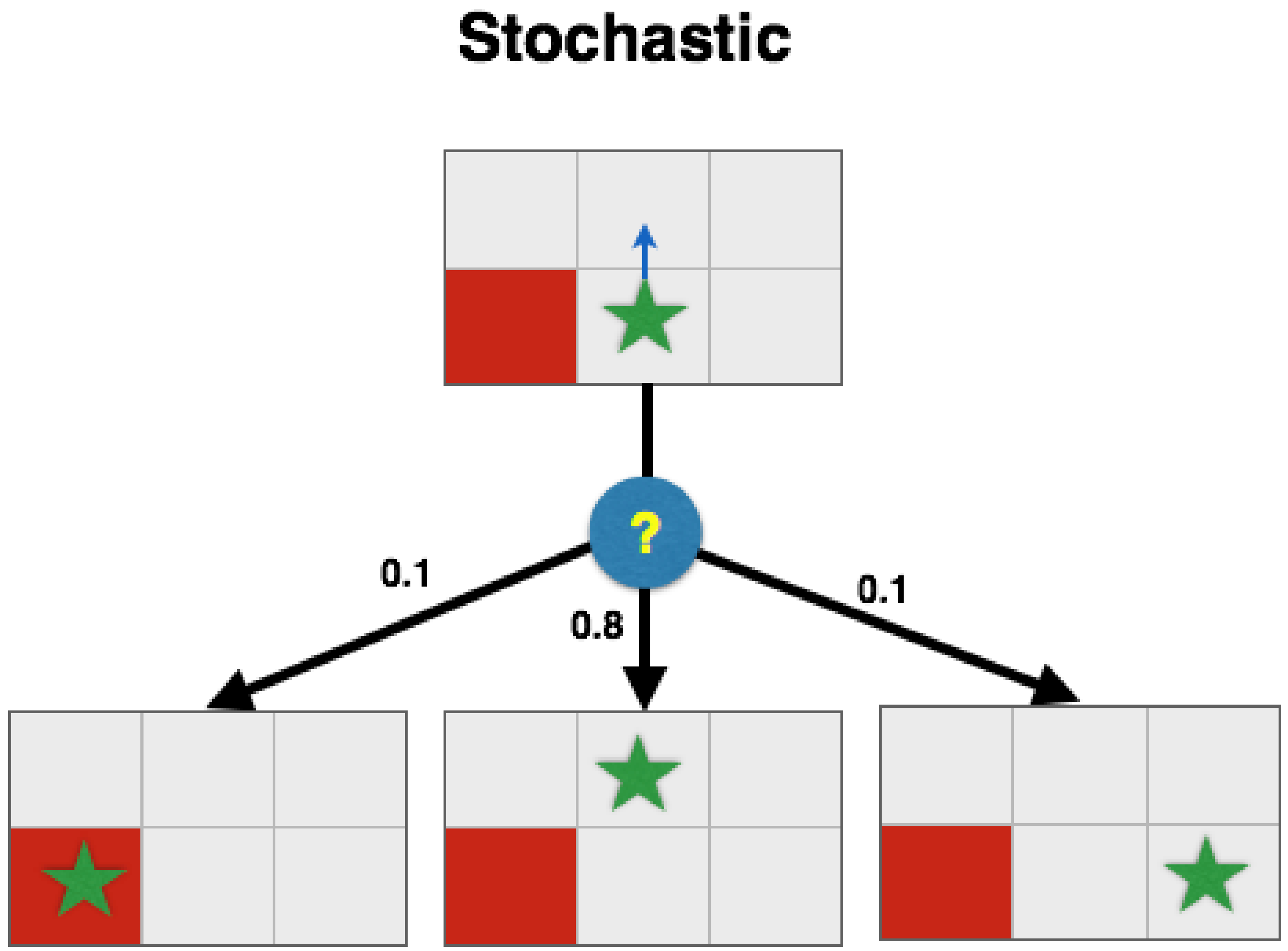
- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent’s path
- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards 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





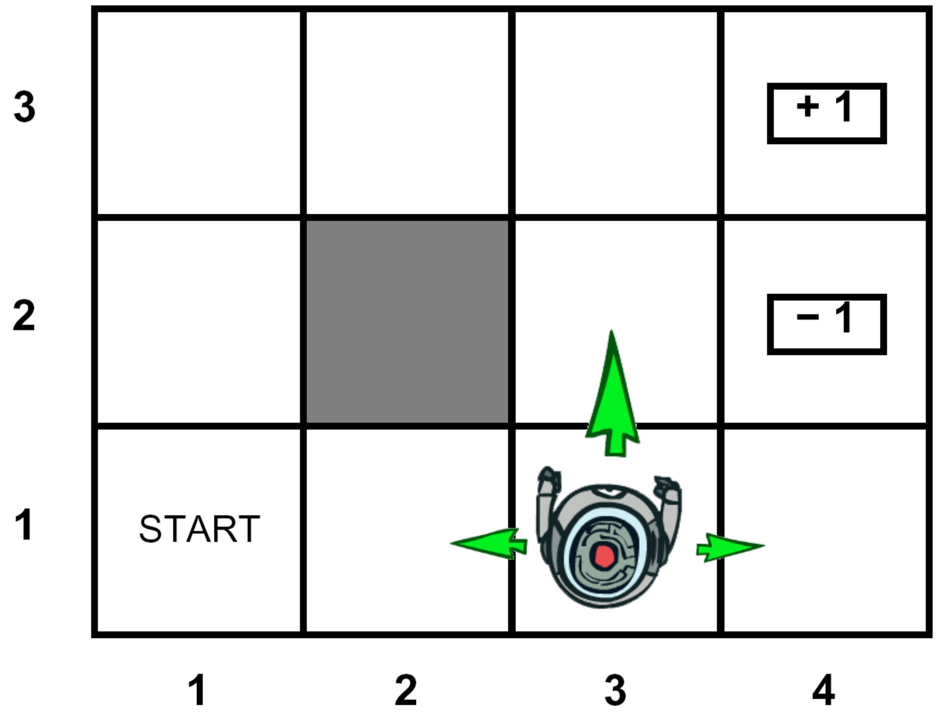
Grid World Actions





Markov Decision Processes

- 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



$T(s_{11}, E, \ldots$

…

$T(s_{31}, N, s_{11}) = 0$

…

$T(s_{31}, N, s_{32}) = 0.8$

$T(s_{31}, N, s_{21}) = 0.1$

$T(s_{31}, N, s_{41}) = 0.1$

…

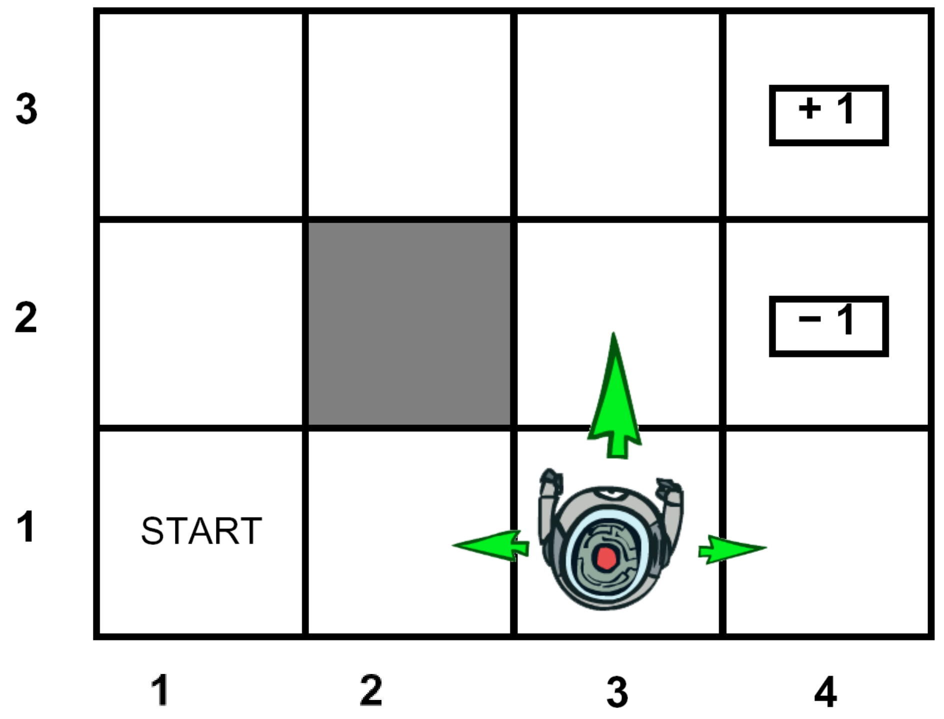
This is a Big Table!

11 X 4 x 11 = 484 entries

For now, we give this as input to the agent

Markov Decision Processes

- 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')



…

$R(s_{32}, N, s_{33}) = -0.01$

…

$R(s_{32}, N, s_{42}) = -1.01$

…

$R(s_{33}, E, s_{43}) = 0.99$

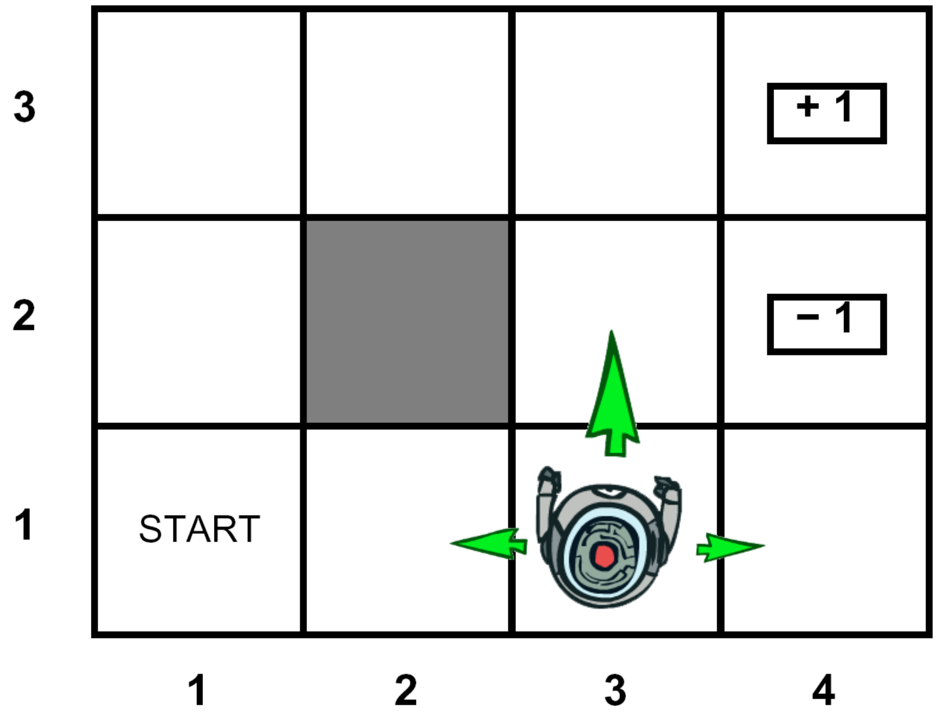
Cost of breathing

R is also a Big Table!

For now, we also give this to the agent

Markov Decision Processes

- An MDP is defined by:
  - A set of states s ∈ S
  - A set of actions a ∈ 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



- MDPs are non-deterministic search problems
  - One way to solve them is with expectimax search
  - We'll have a new tool soon

What is Markov about MDPs?

- "Markov" generally means that given the present state, the future and the past are independent

- For Markov decision processes, "Markov" means action outcomes depend only on the current state

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \dots S_0 = s_0)$$
$$=$$
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- This is just like search, where the successor function could only depend on the current state (not the history)

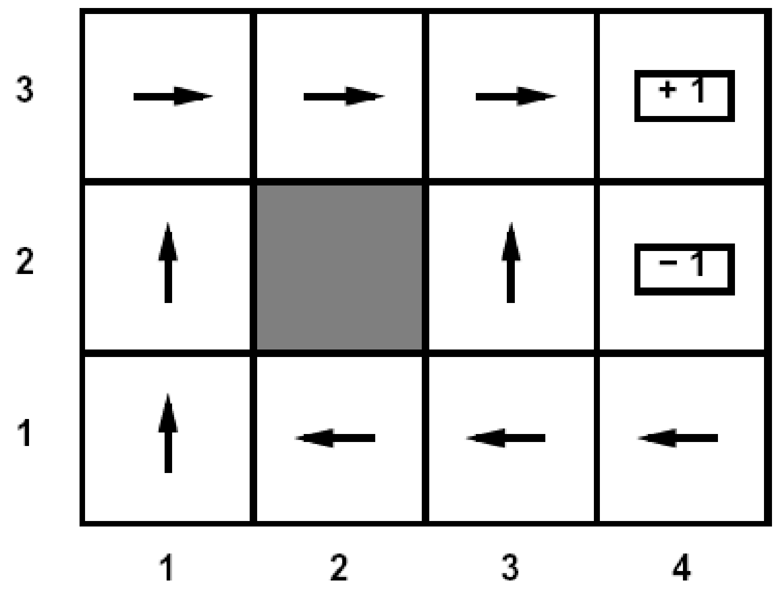
Andrey Markov
(1856-1922)

Policies

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

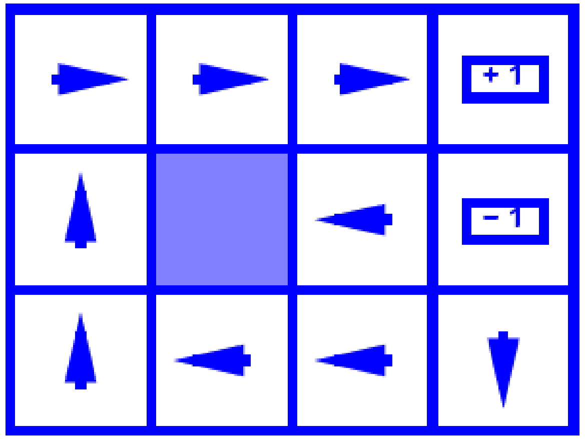
- For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy is one that maximizes expected utility if followed
  - An explicit policy defines a reflex agent

- Expectimax didn't compute entire policies
  - It computed the action for a single state only

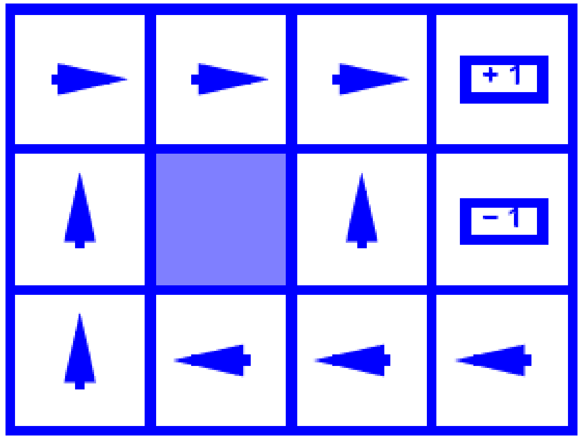


Optimal policy when R(s, a, s') = -0.03 for all non-terminals s

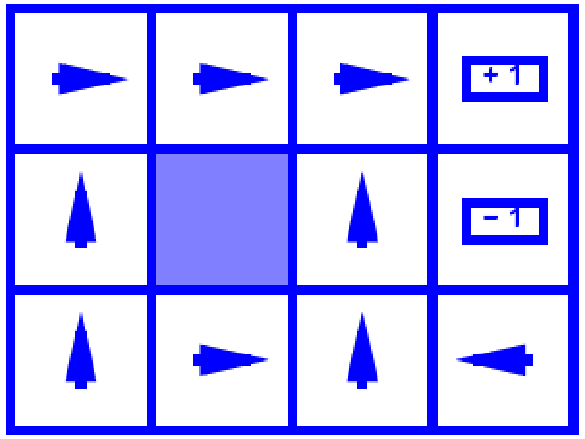
Optimal Policies



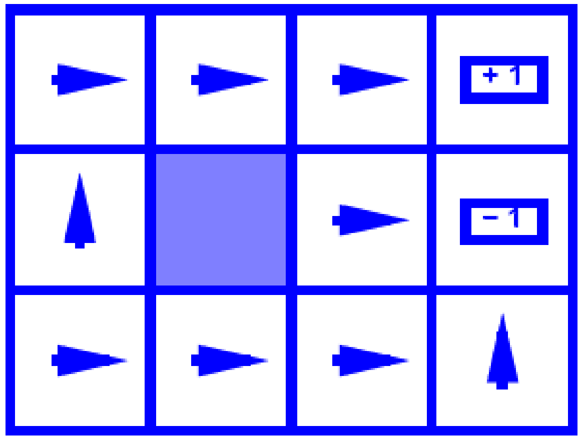
R(s) = -0.01



R(s) = -0.03



R(s) = -0.4



R(s) = -2.0