Making Complex Decisions

Markov Decision Processes

CH 17

Section Overview

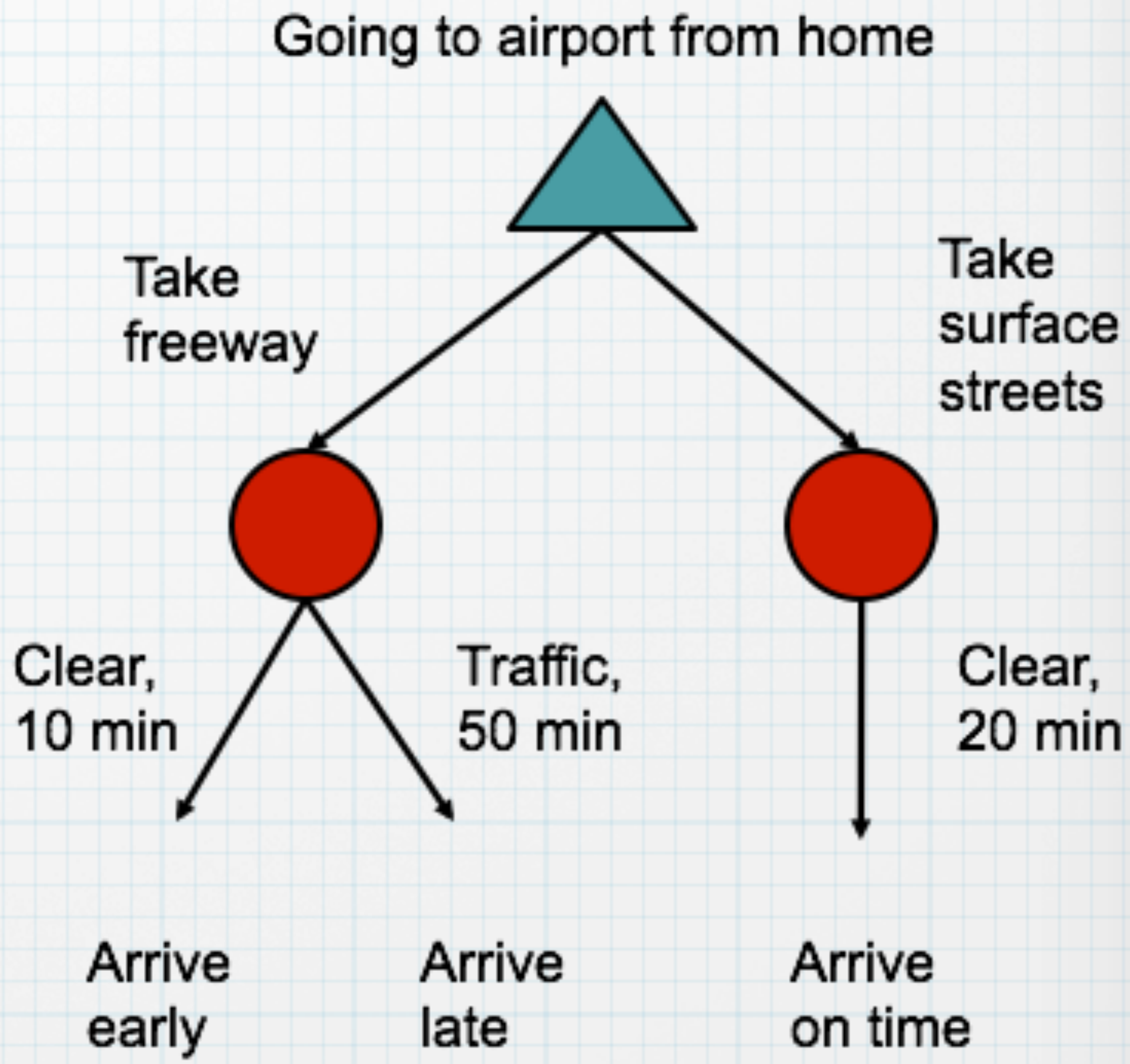
- Representing preferences
- Markov Decision Processes
- Solving MDPs
  - Value Iteration
  - Policy Iteration
- Partially Observable MDPs

Maximum Expected Utility

* Rational Agent = Maximizes its expected utility given its knowledge
* Questions:
  * Where do utilities come from?
  * Why expected utility?

Example

* One way has a chance to be better or worse
* How to decide?
* Which would you pick if you are catching a flight?
* Which if you are picking up a friend?



Assigning relative value to outcomes = Utilities

Agent Rational Decisions

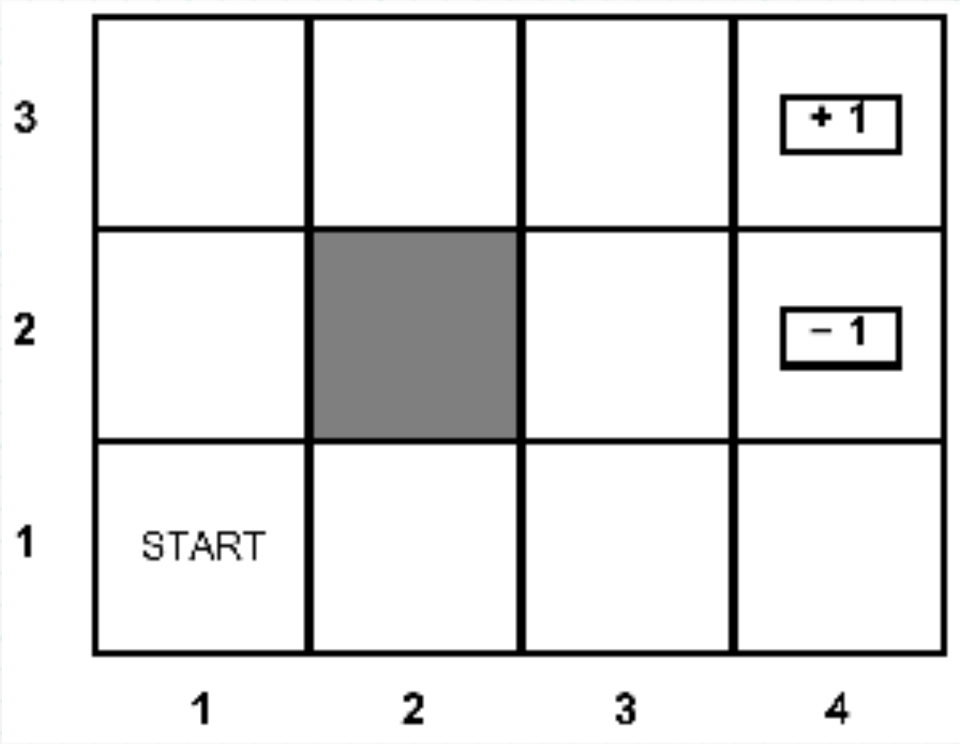
* Representing Decisions, Maximize Utility
  * Receive feedback = **rewards**
  * Utility defined by the **reward function**
  * Act to maximize expected rewards
  * Can learn to maximize rewards via Reinforcement Learning

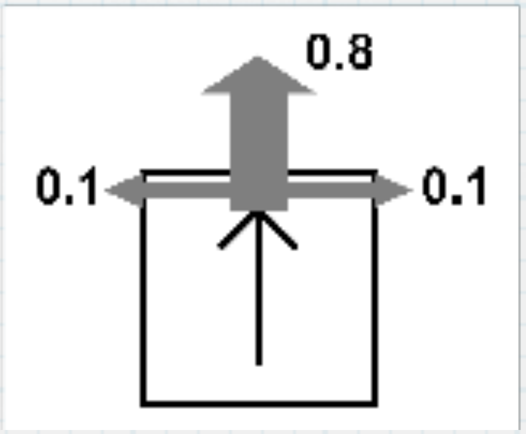
Reward Functions

* For example:
  * Playing a game, reward defined at the end for winning or losing
  * Vacuuming agent, reward for each piece of dirt
  * Autonomous taxi, reward for each passenger delivered

MDP Grid World

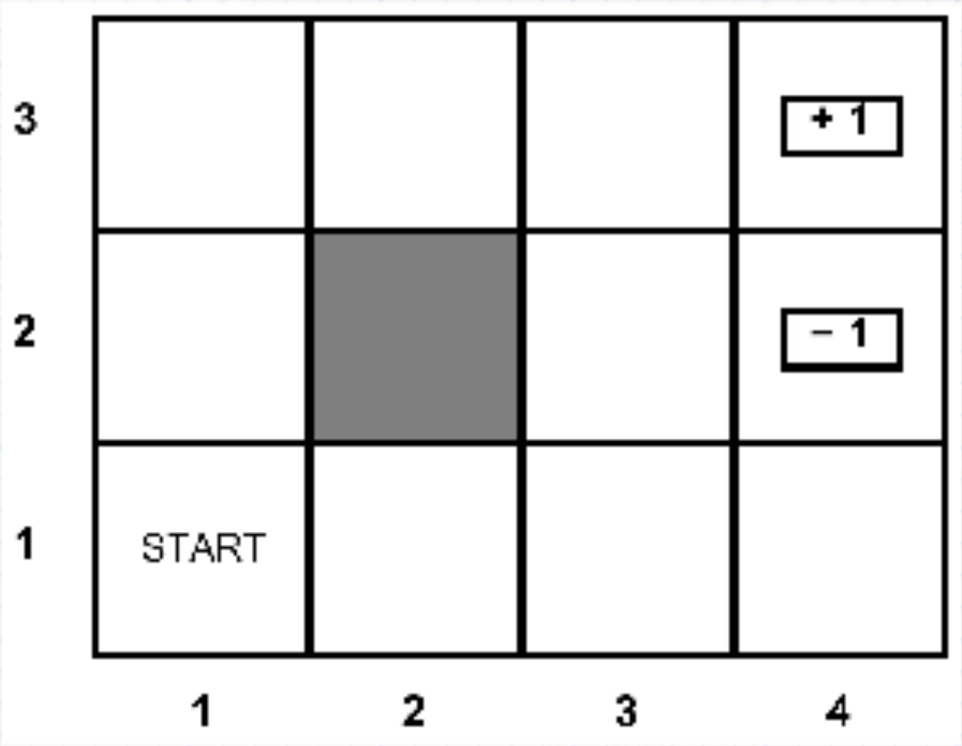
* Example agent for our MDP discussion
* Walls block the agent
* Actions only work 80% of the time
* Big rewards at the end

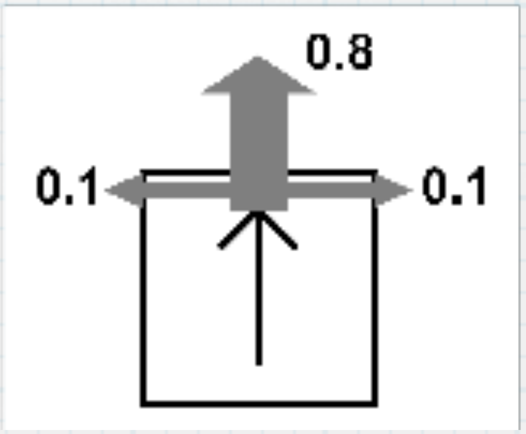




Markov Decision Processes

* MDP defined by
  * States $s \in S$
  * Actions $a \in A$
  * Transition function $T(s,a,s')$
  * Reward function $R(s,a,s')$





Just like our old search formulation but with non-det actions and rewards

What makes it Markov?

* Markov means: given the present state future and past are independent
* Specifically for MDPs Markov means

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

$$=$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

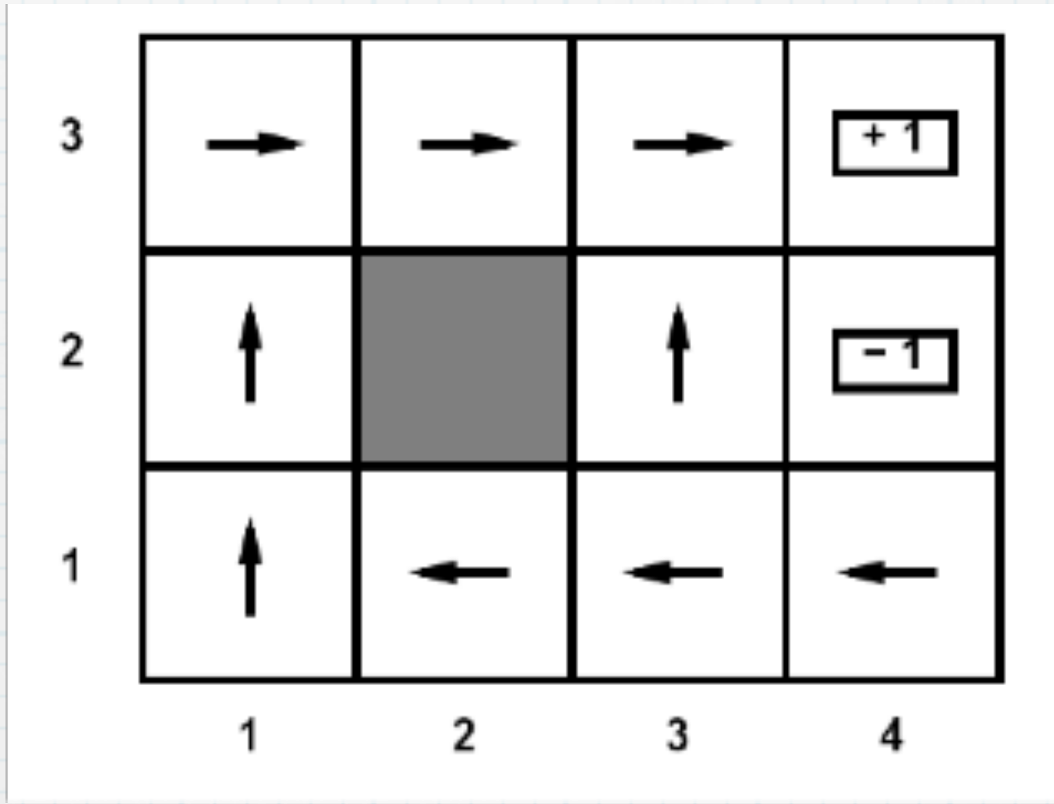
Solving MDPs

* In deterministic single-agent search problem we solved for an optimal plan, sequence of actions
* Now an optimal policy $\pi^* = S \rightarrow A$
  * policy, π, gives an action for every state
  * optimal policy, π^*, maximizes expected utility
  * defines a reflex agent

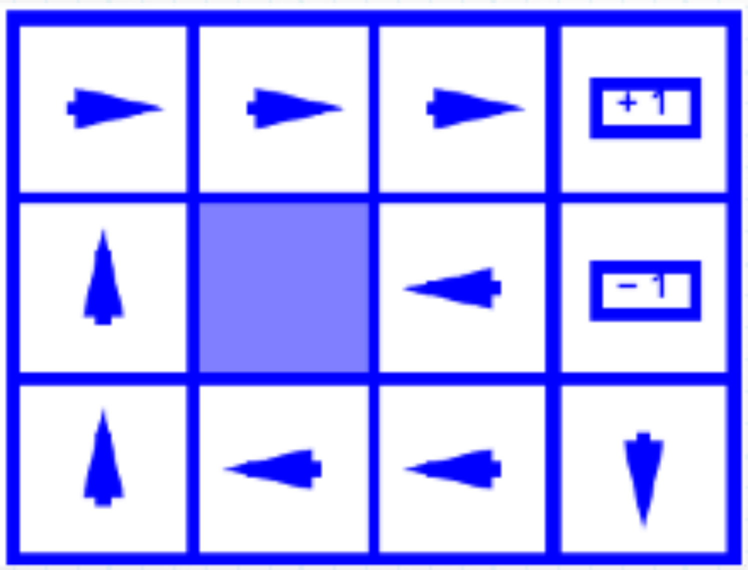
Solving MDPs

* In deterministic single-agent search problem we solved for an optimal plan, sequence of actions
* Now an optimal policy $\pi^* = S \rightarrow A$

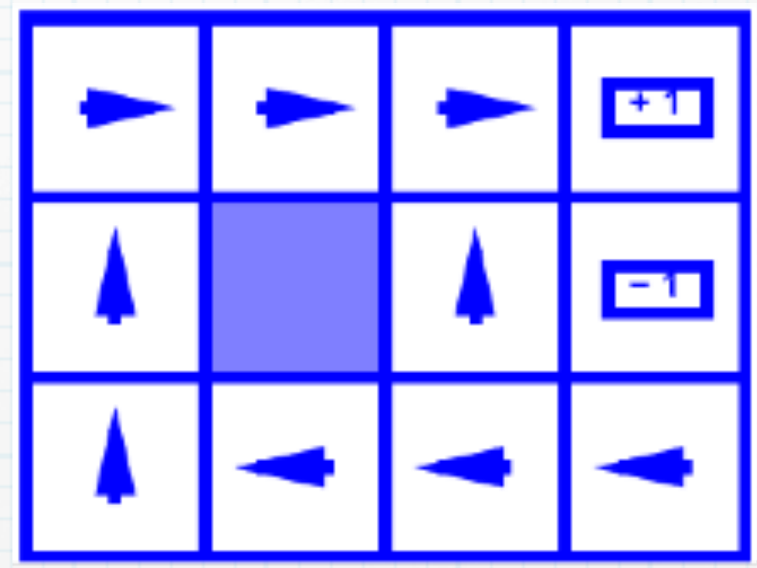
Optimal policy when R(s, a, s') = -0.03 for all non-terminals s



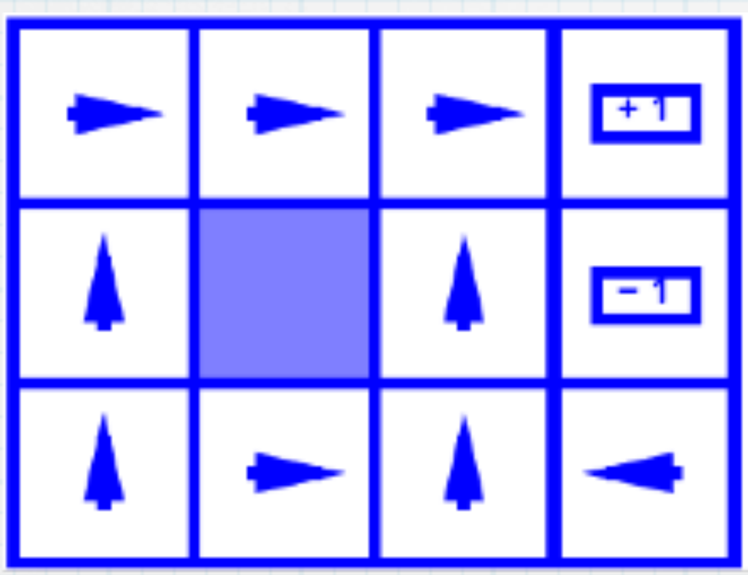
Example Optimal Policies



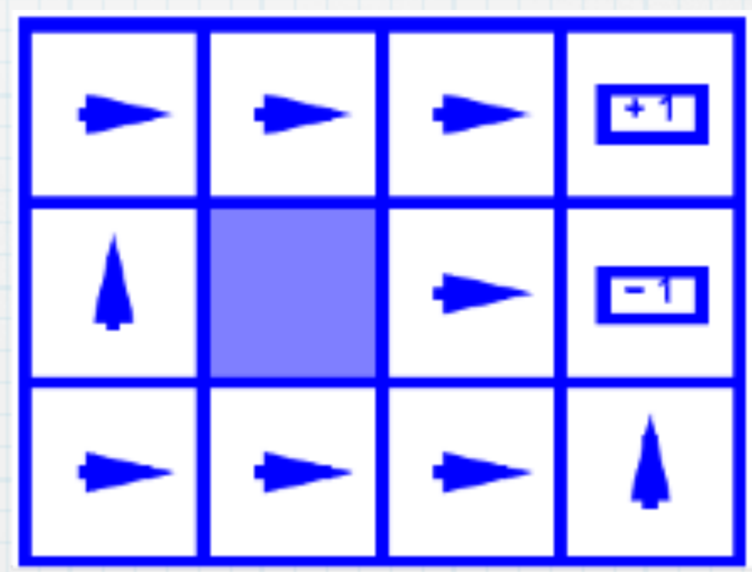
R(s) = -0.01



R(s) = -0.03



R(s) = -0.4



R(s) = -2.0

Summary + Preview

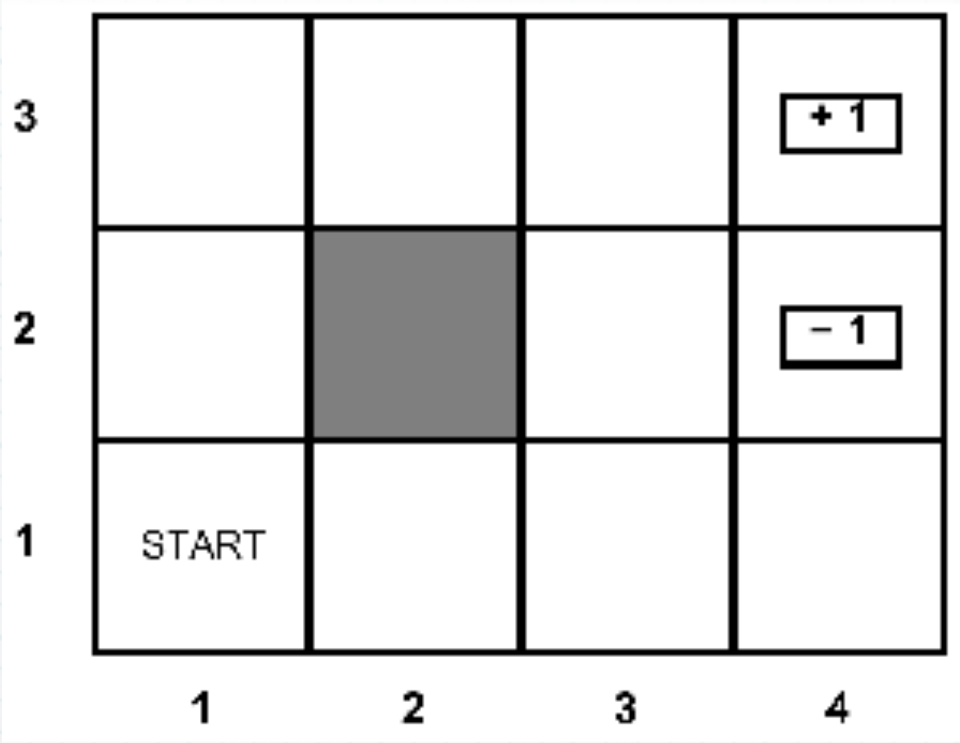
* Preferences & Utilities & Rewards
* Markov Decision Processes
* Next...
  * Two algorithms for solving an MDP
  * Partially observable states

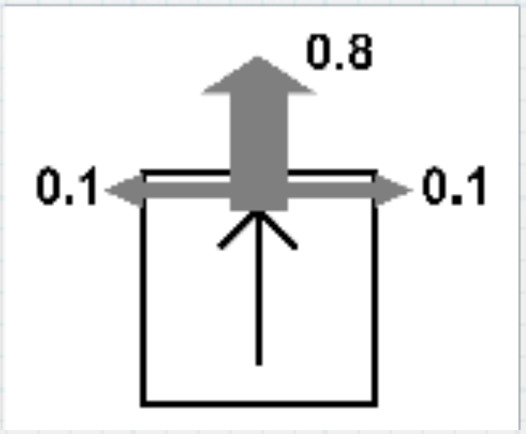
Making Decisions

* When agent will need to act in the same environment over and over, to achieve the same goal, with nondeterministic actions
* "Policy" of action instead of "Plans"

Markov Decision Processes

* MDP defined by
  * States $s \in S$
  * Actions $a \in A$
  * Transition function $T(s,a,s')$
  * Reward function $R(s,a,s')$





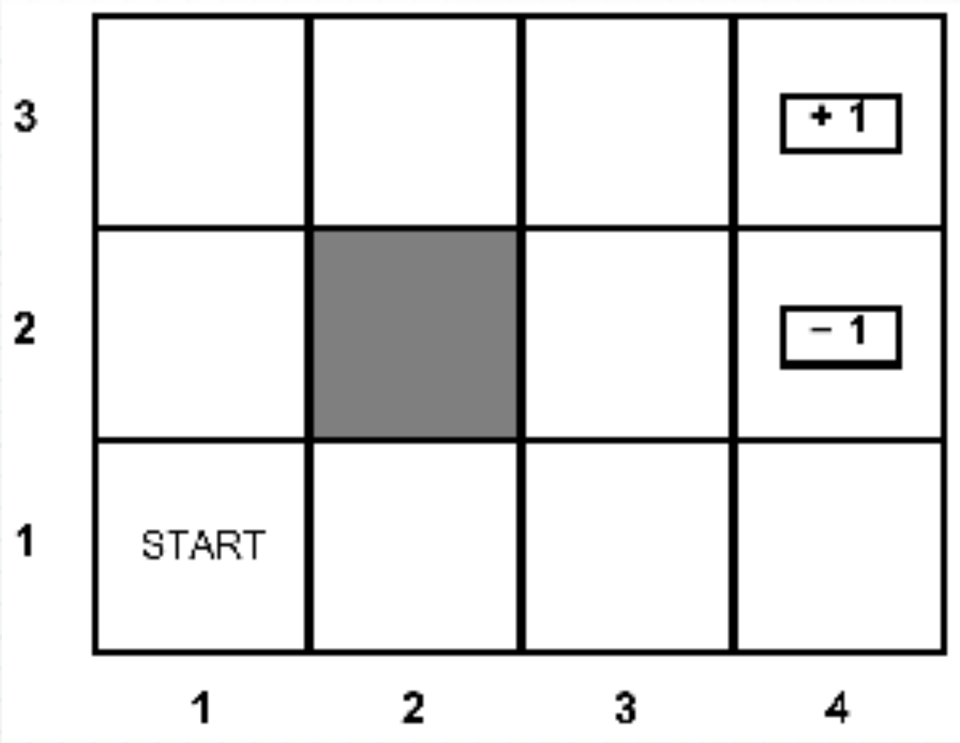
Trashcan Robot MDP

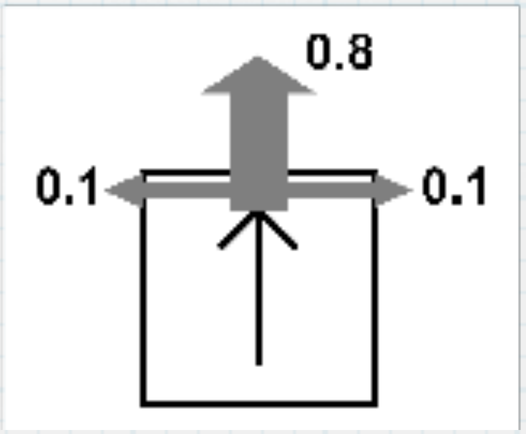
You are building a robot to move around an office building collecting trash (from tables, floors, people, etc.). Define how you would set up this task as a Markov Decision Process

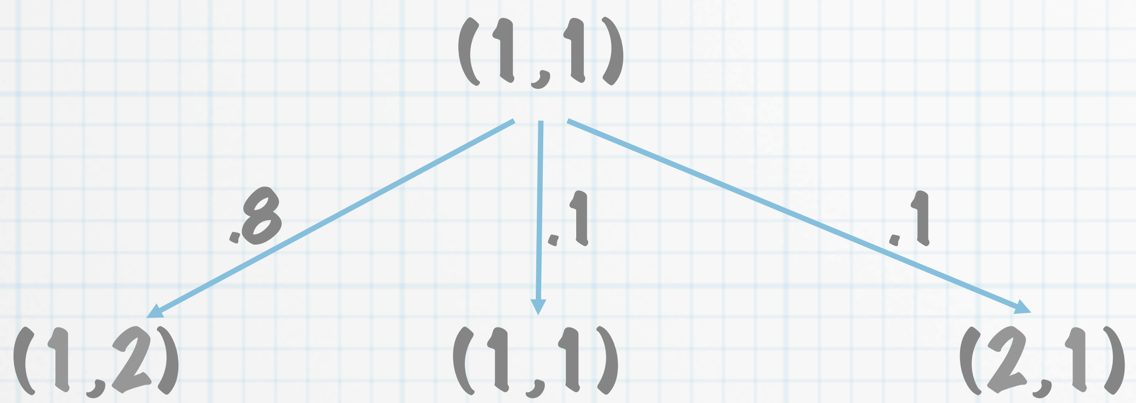
* What is the state space?
* What are the actions needed?
* What should the rewards be?

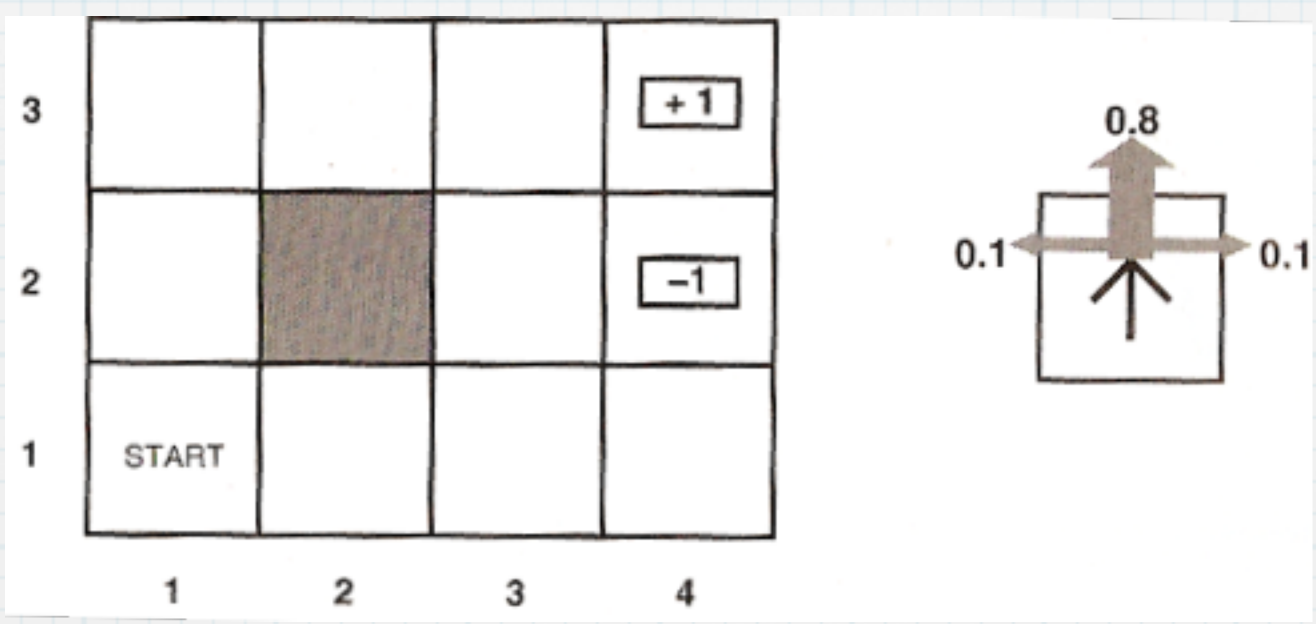
MDP Practice Exercise

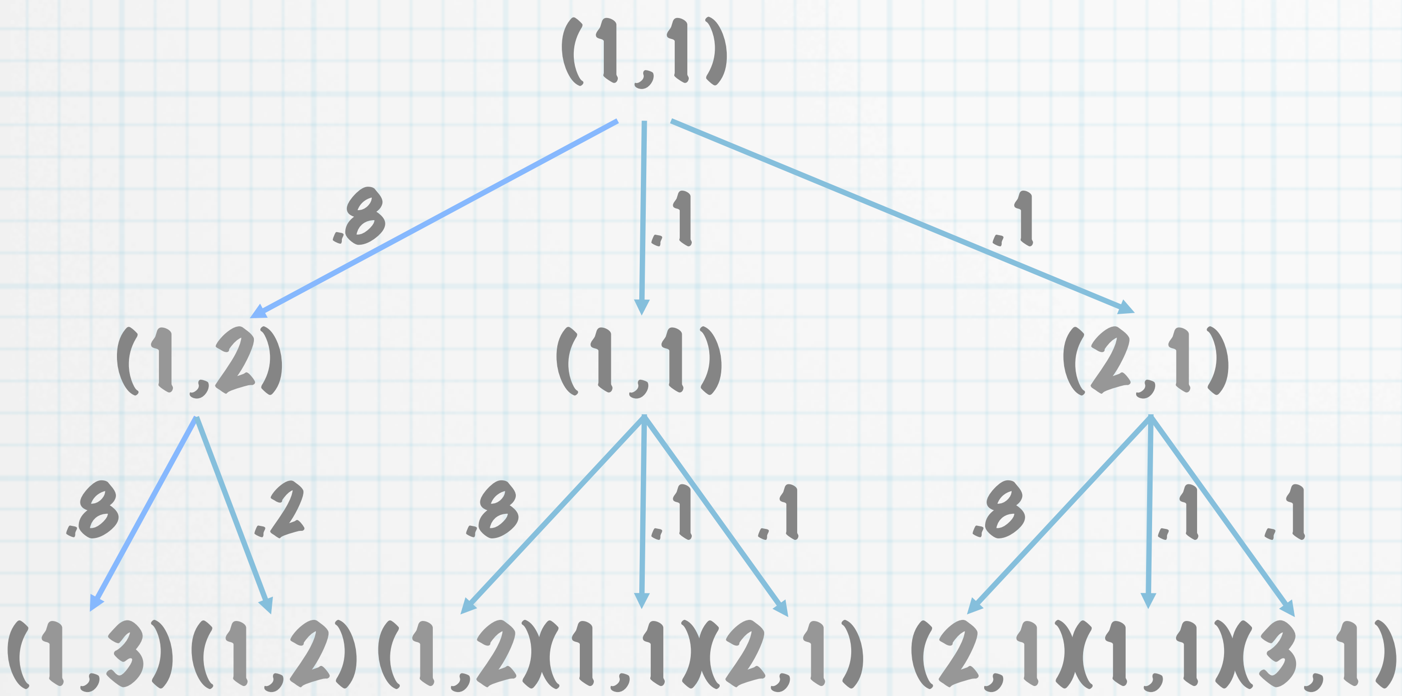
- What are the possible states the agent could be in after moving: a_1=North, a_2=North
- With what probabilities?



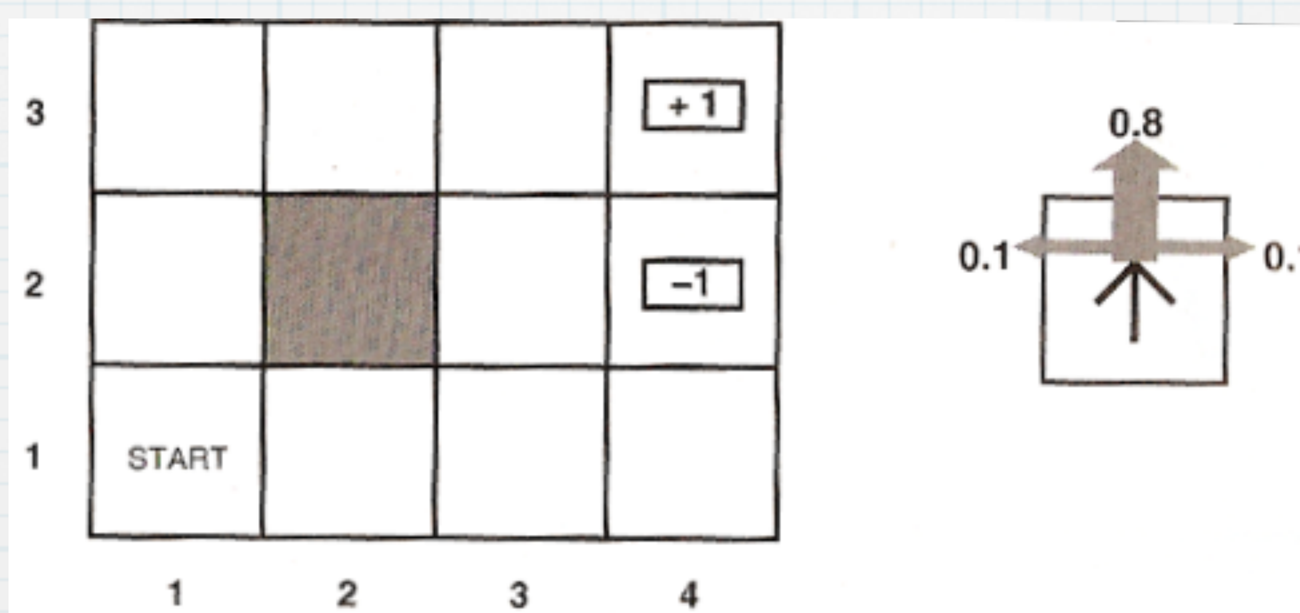



(1,1)
.8
.1
.1
(1,2)
(1,1)
(2,1)

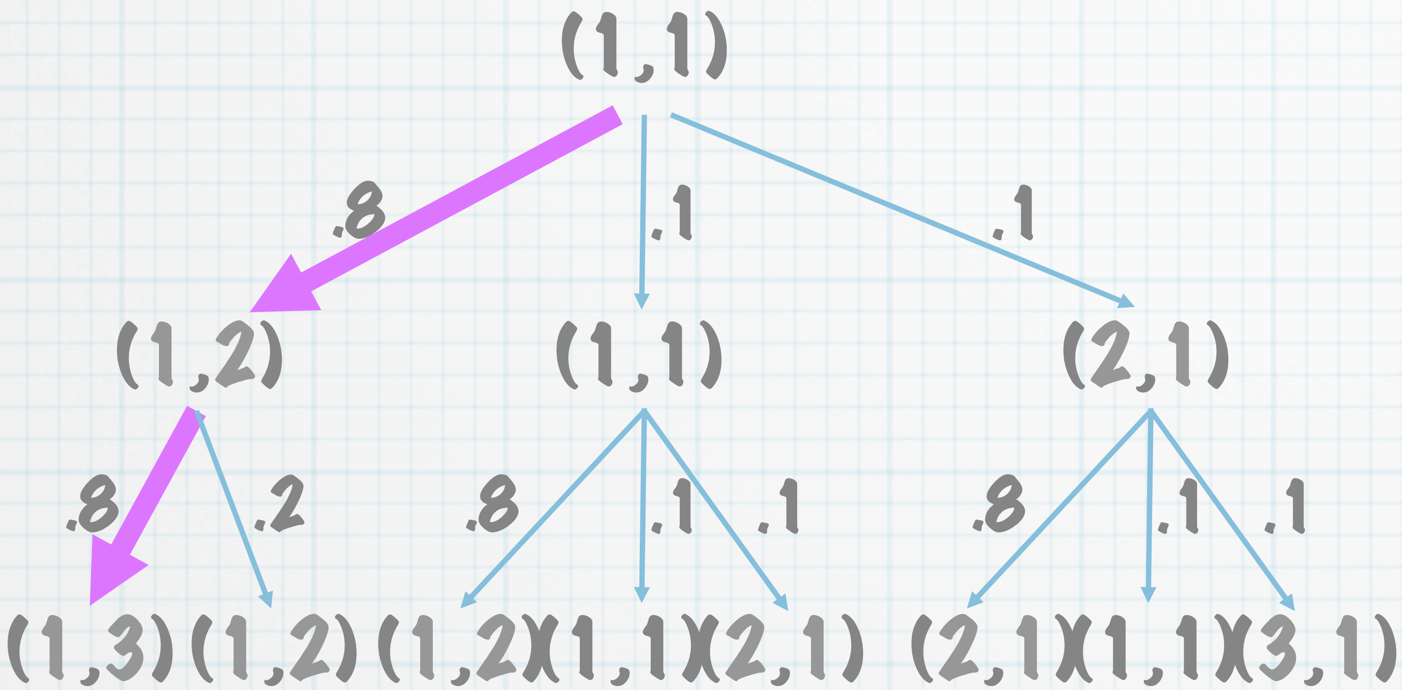

3
2
1
+1
–1
START
1
2
3
4
0.8
0.1
0.1



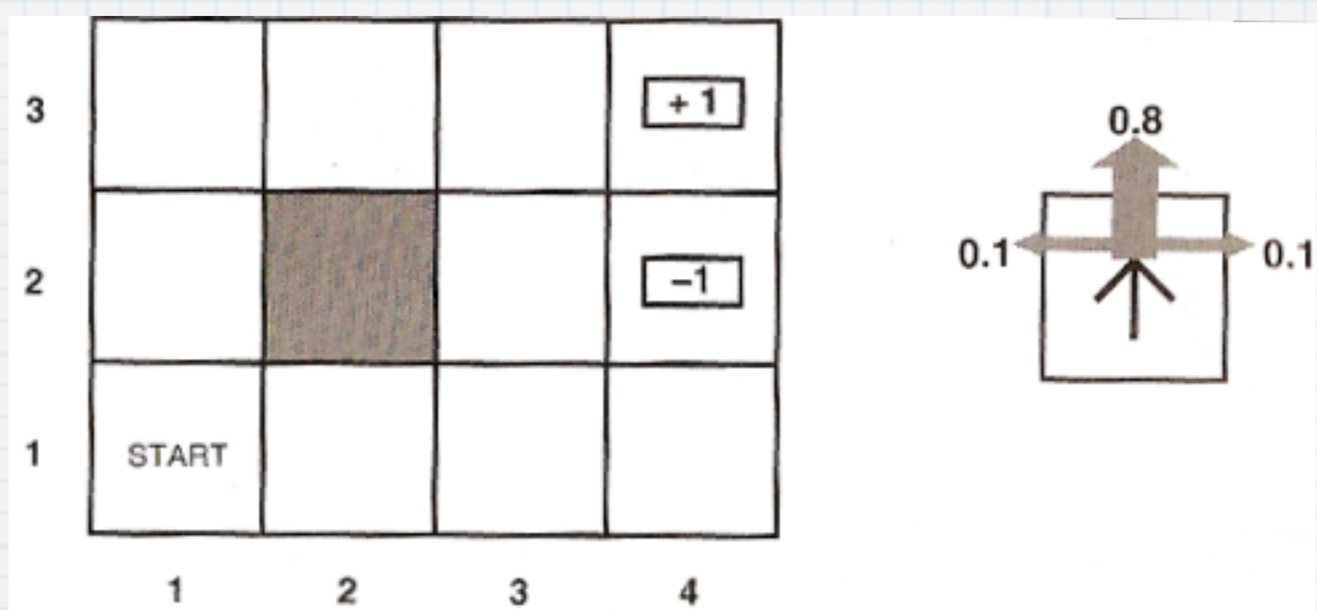
(1,1)
.8
.1
.1
(1,2)
(1,1)
(2,1)
.8
.2
.8
.1
.1
.8
.1
.1
(1,3)
(1,2)
(1,2)
(1,1)
(2,1)
(2,1)
(1,1)
(3,1)

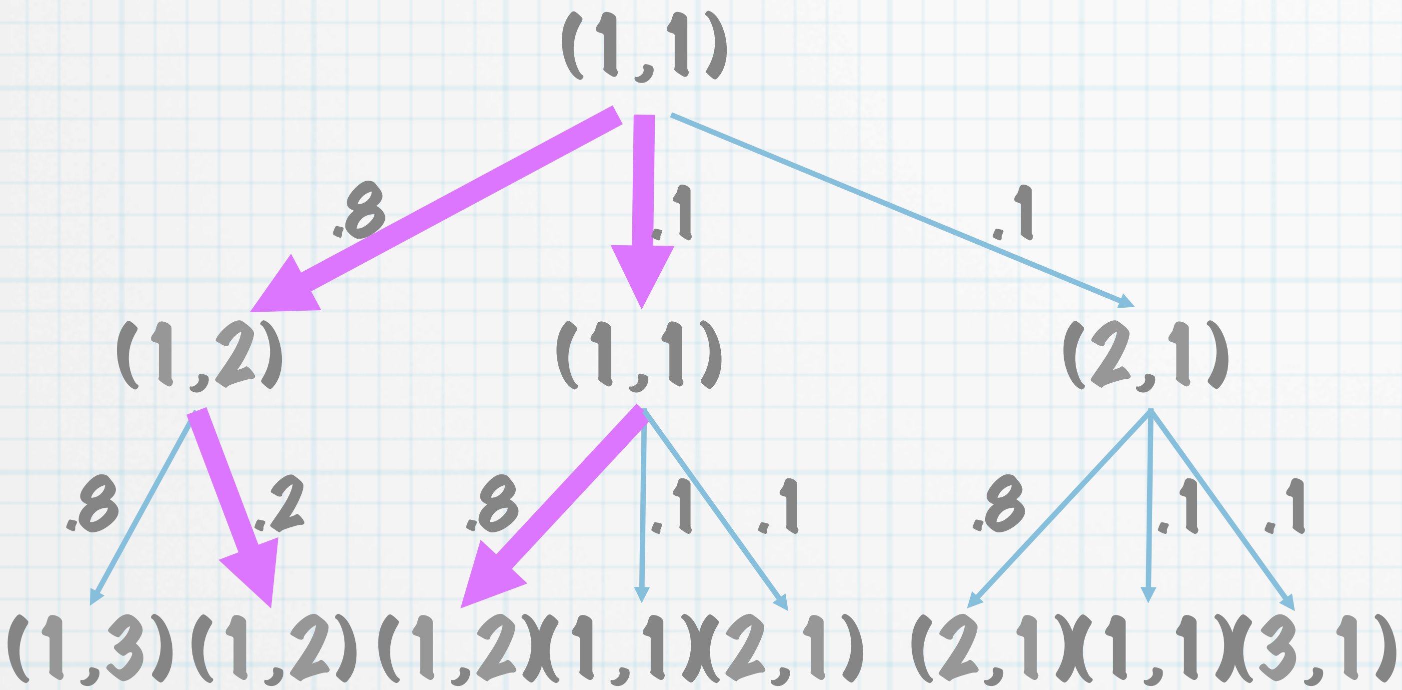


3
2
1
START
+1
−1
1
2
3
4
0.8
0.1
0.1


(1,1)
.8
.1
.1
(1,2)
(1,1)
(2,1)
.8
.2
.8
.1
.1
.8
.1
.1
(1,3) (1,2) (1,2)(1,1)(2,1) (2,1)(1,1)(3,1)

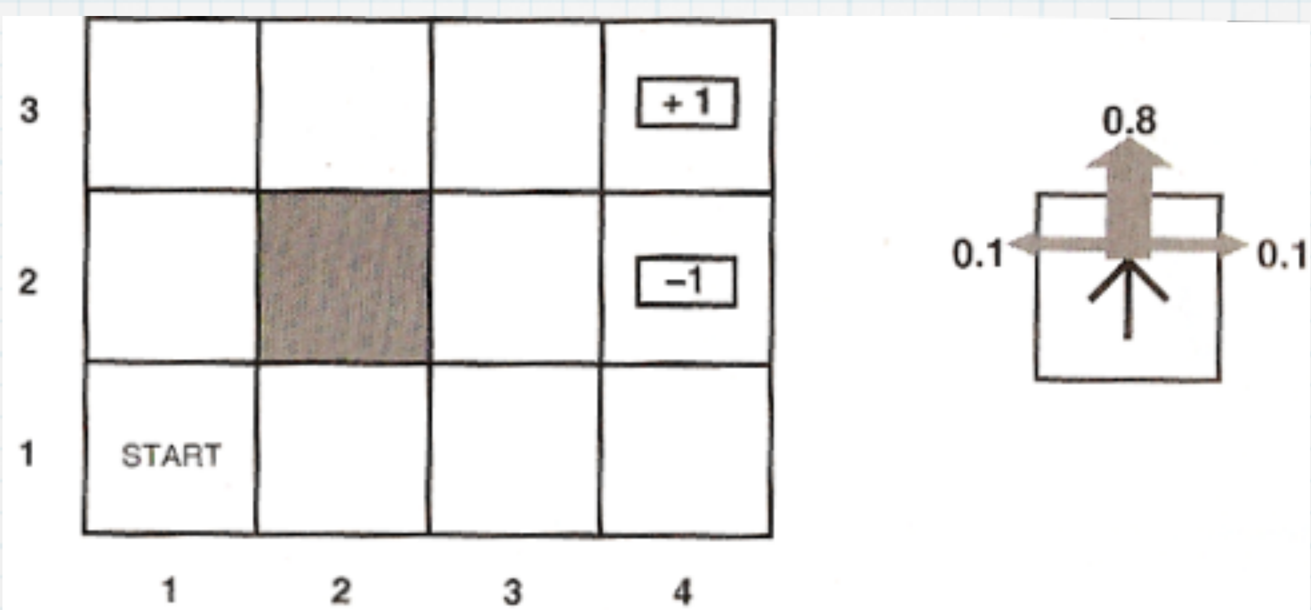
(1,3) = .8*.8 = .64

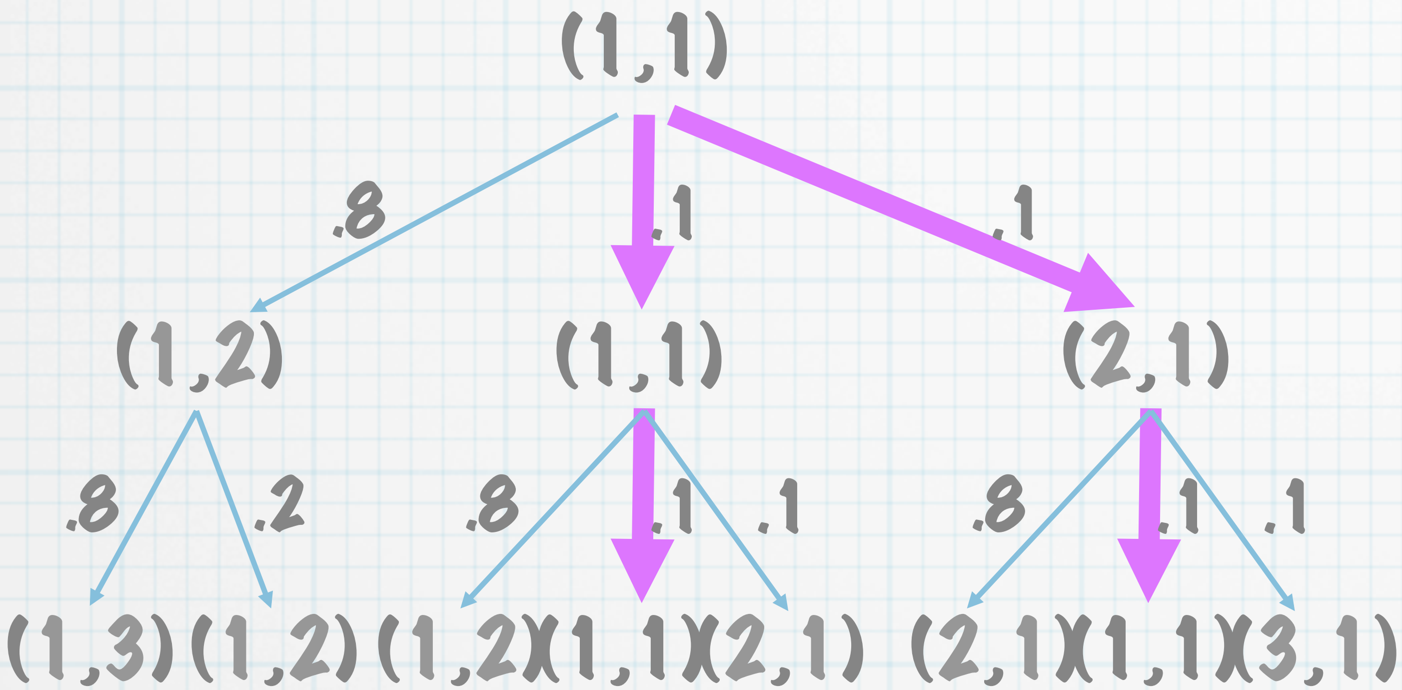

3
2
1
+1
−1
START
1
2
3
4
0.8
0.1
0.1



$$(1,3) = .8 * .8 = .64$$

$$(1,2) = .8 * .2 + .1 * .8 = .24$$

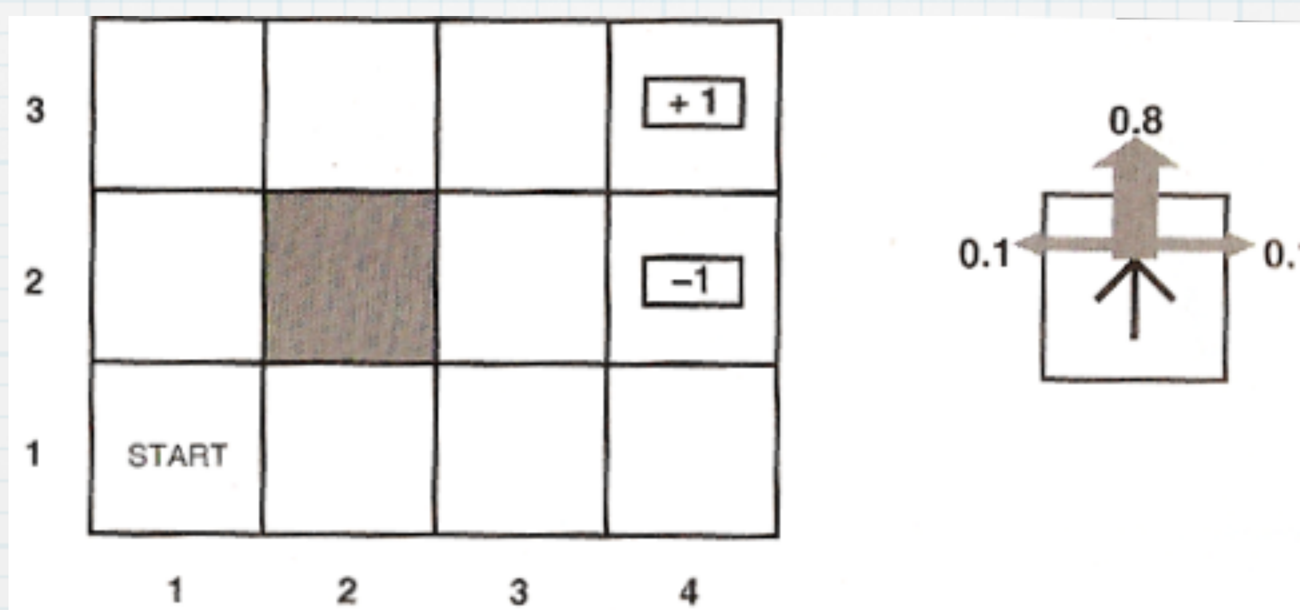


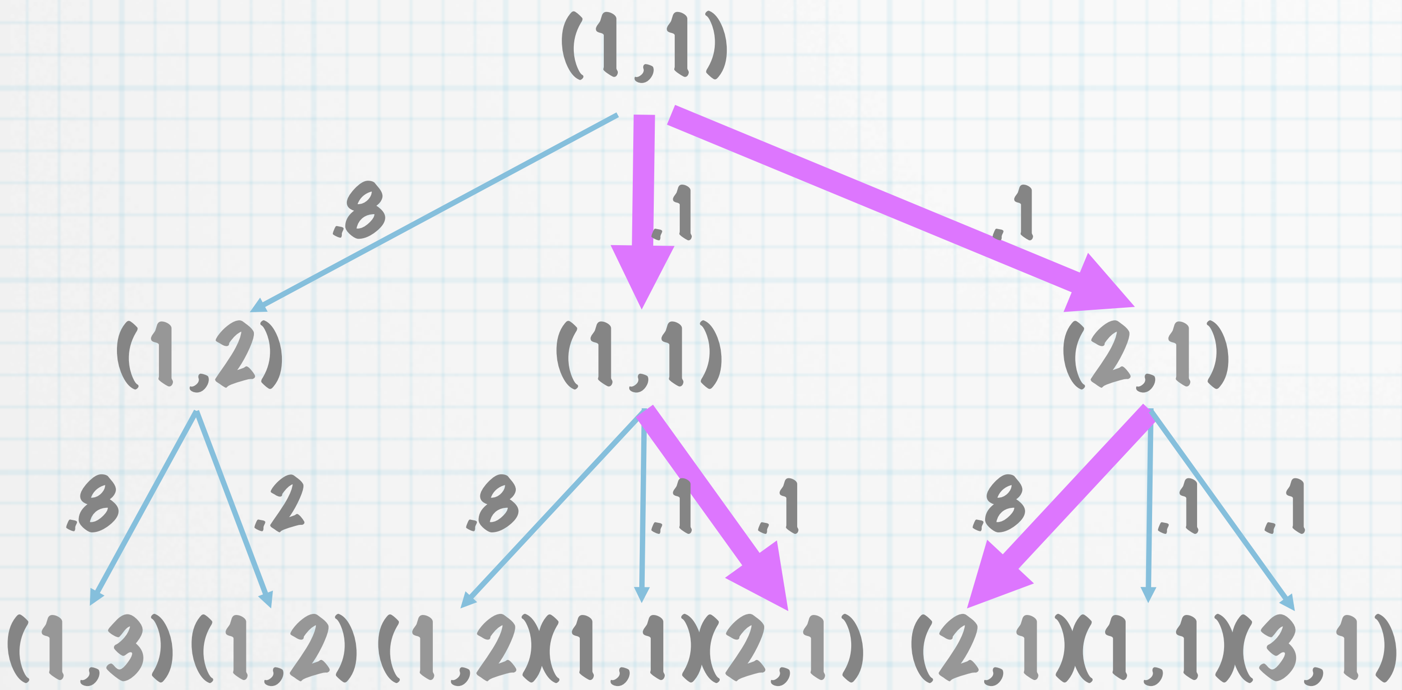


$(1,3) = .8 * .8 = .64$

$(1,2) = .8 * .2 + .1 * .8 = .24$

$(1,1) = .1 * .1 + .1 * .1 = .02$



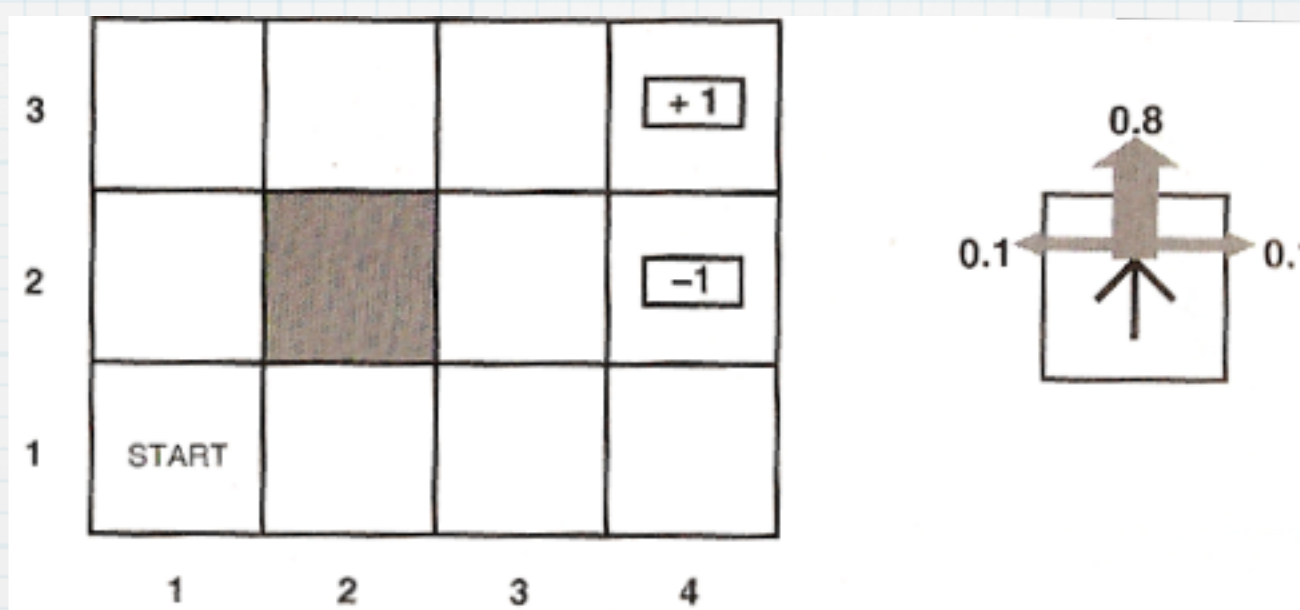


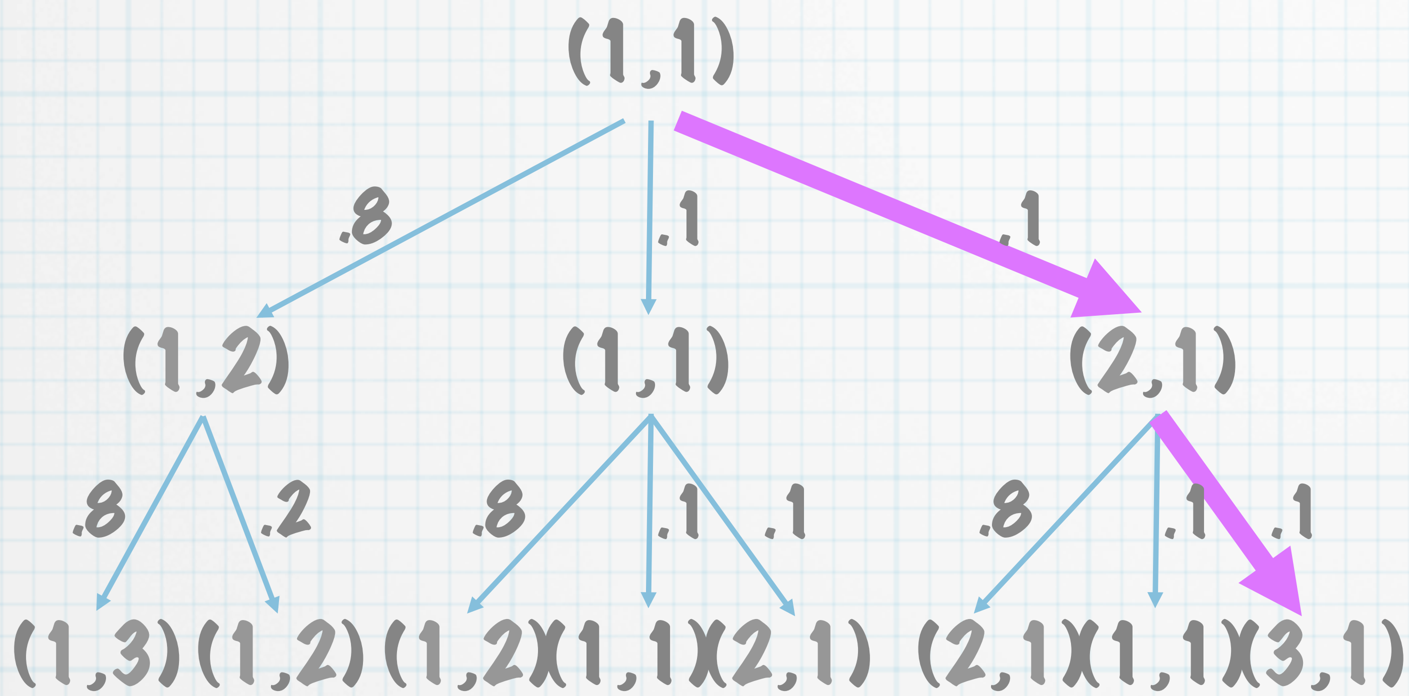
$(1,3) = .8 * .8 = .64$

$(1,2) = .8 * .2 + .1 * .8 = .24$

$(1,1) = .1 * .1 + .1 * .1 = .02$

$(2,1) = .1 * .1 + .1 * .8 = .09$





$P(s_2)$

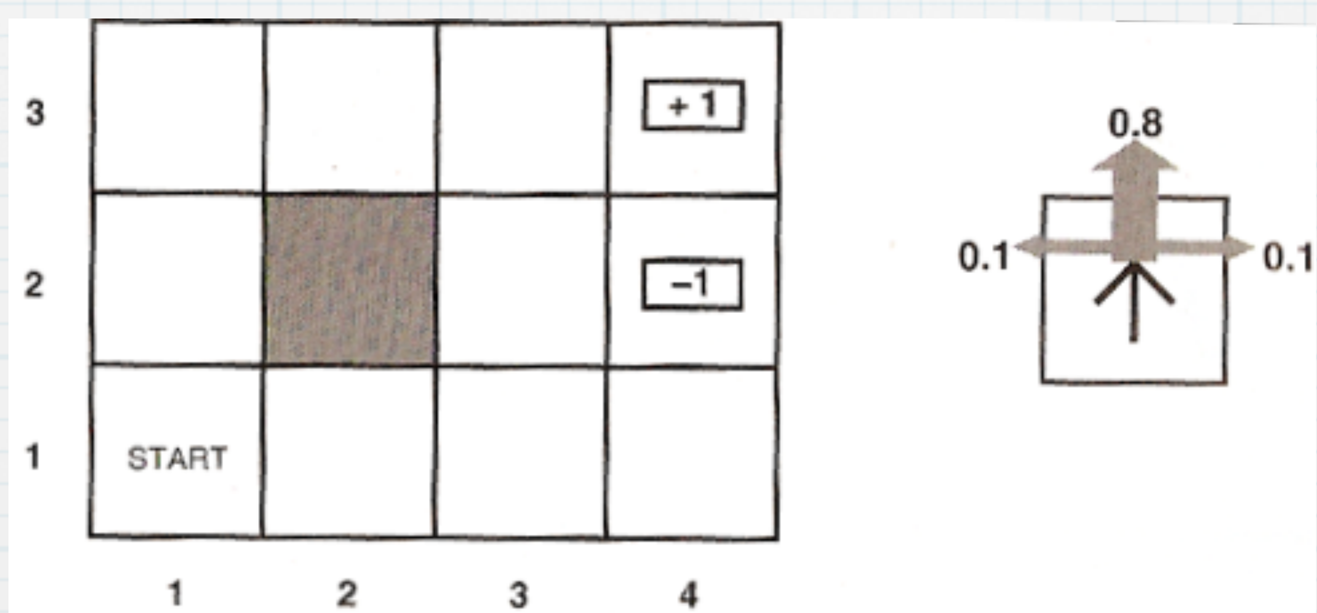
$(1,3) = .8 * .8 = .64$

$(1,2) = .8 * .2 + .1 * .8 = .24$

$(1,1) = .1 * .1 + .1 * .1 = .02$

$(2,1) = .1 * .1 + .1 * .8 = .09$

$(3,1) = .1 * .1 = .01$



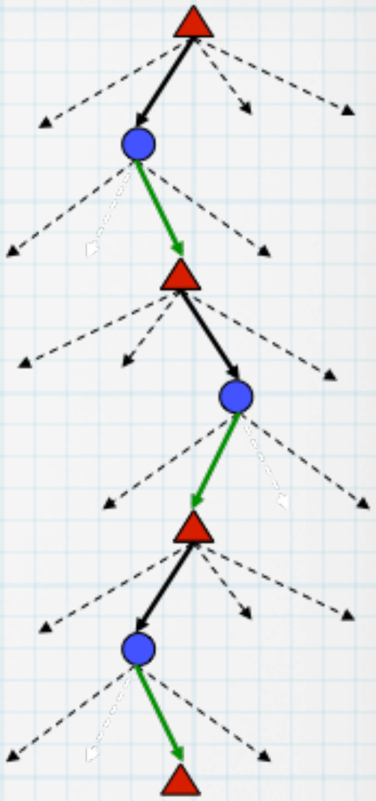
Markov Decision Process (MDP)

MDP Solution = policy: what action to do in every state

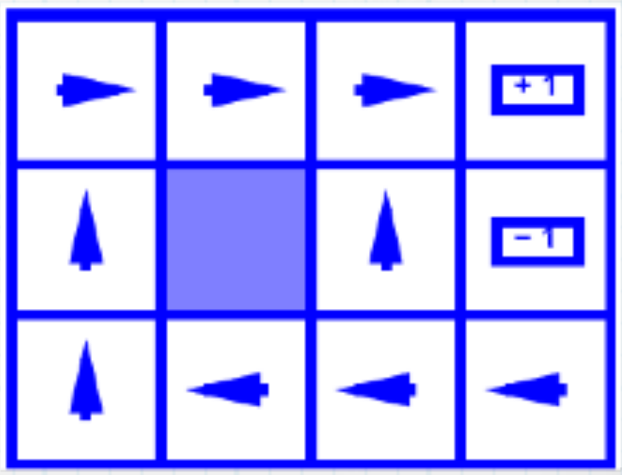
Expected Utility: the expected reward from executing a particular policy

Optimal Policy: has the highest expected utility

Utility of Sequences

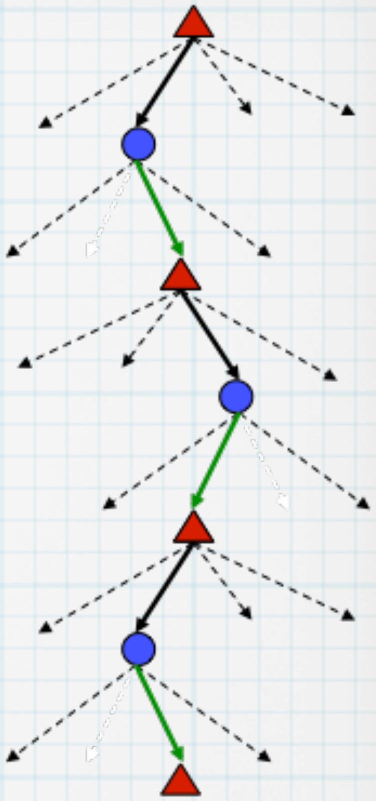


- Finite
  - fixed time the agent is around
  - the right action in a state depends on how much time left
- Infinite
  - no time limits, optimal action only depends on the state



R(s) = -0.03

Utility of Sequences



* **Finite**
  * fixed time the agent is around
  * the right action in a state depends on how much time left
* **Infinite**
  * no time limits, optimal action only depends on the state

Non-stationary

Stationary

Utilities of Sequences

* Problem: infinite sequences have infinite rewards

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

* Solution?
    * cut-off sequences, to make finite
    * ...non-stationary policies

Utilities of Sequences

* Problem: infinite sequences have infinite rewards

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

* Solution: discount future rewards

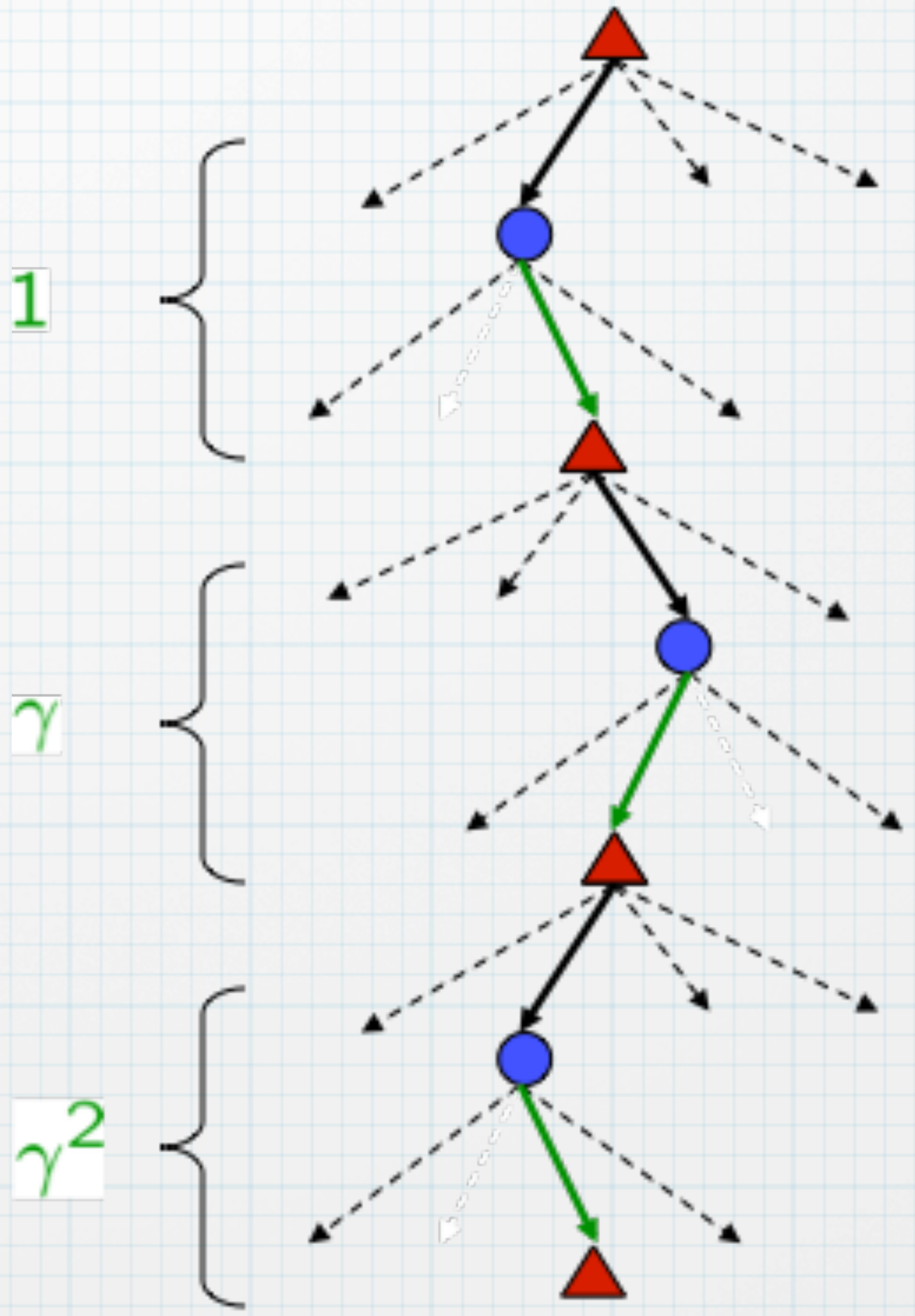
$$U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$$

Discount Future

* Discount factor for each time step $\gamma<1$

* Earlier rewards have higher utility than later rewards

* With discounted rewards, the utility of an infinite sequence is finite

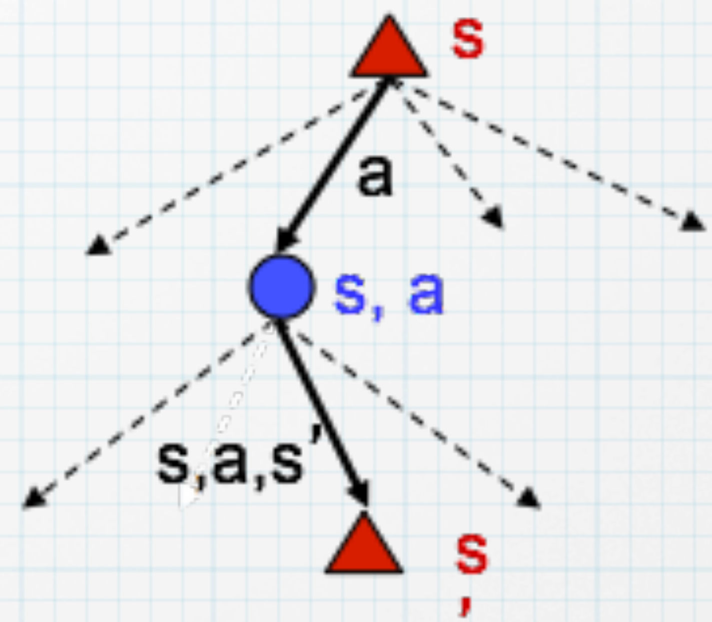


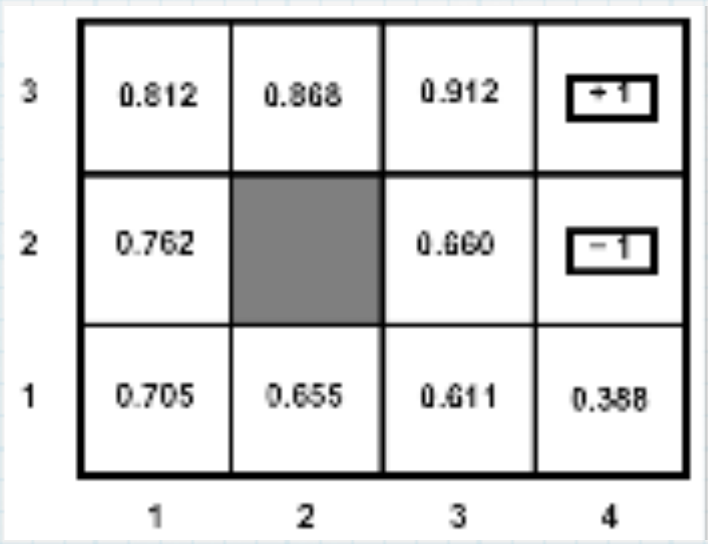
Optimal Values for States

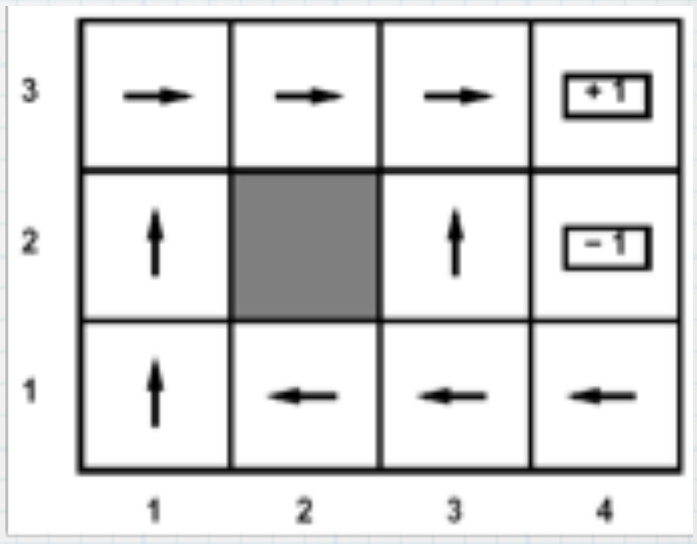
* Optimal values define optimal policies

* U*(s) = expected utility starting in s and acting optimally

* Q*(s,a) = expected utility taking a in s, and then acting optimally







Calculating Policies

MDPs end up being a good representation for many real world problems.

Given an MDP description, several algorithms for solving for the optimal policy

Two general classes of algorithms:

Value Iteration and Policy Iteration

Optimal Utility

* Utility or Value of a state
  * Expected utility of the sequence to follow that state, with particular policy of action
  * $U^*(s)$ is the expected utility of following the optimal policy from s
* $U(s)$ vs. $R(s)$

Optimal Policy

$$U^{\pi}(S) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right]$$

$$\pi_S^* = \arg\max_{\pi} U^{\pi}(S)$$

$$\pi_{s_t}^* = \arg\max_{a \in A(s_t)} \sum_{s_{t+1}} P(s_{t+1} \mid s_t, a) U(s_{t+1})$$

Expected utility