

CSCE A405/A605 (Adv) Artificial Intelligence

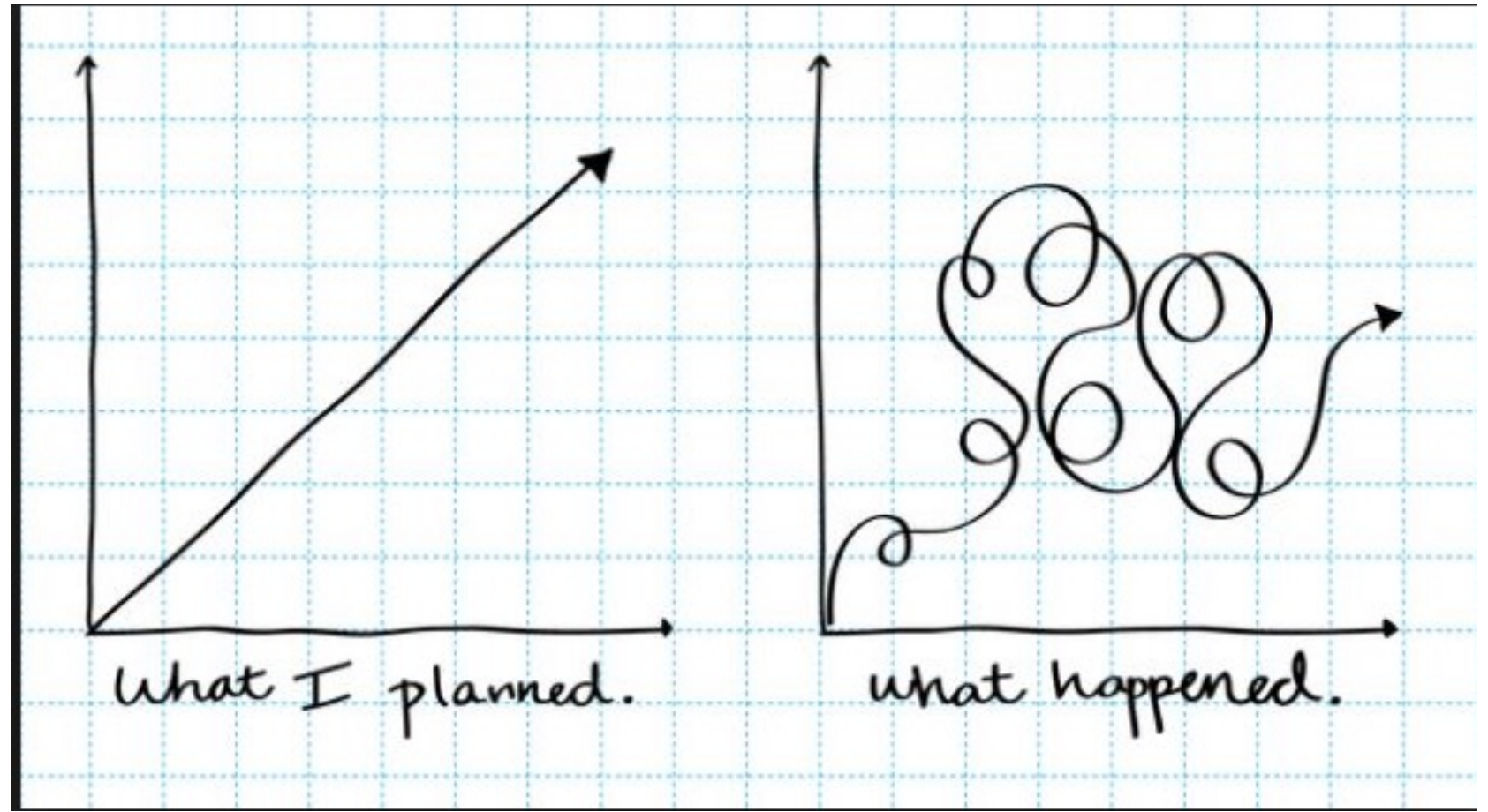
Making Complex Decisions

Ref: Artificial Intelligence: A Modern Approach, 4th ed by Stuart Russell and Peter Norvig, chapter 17

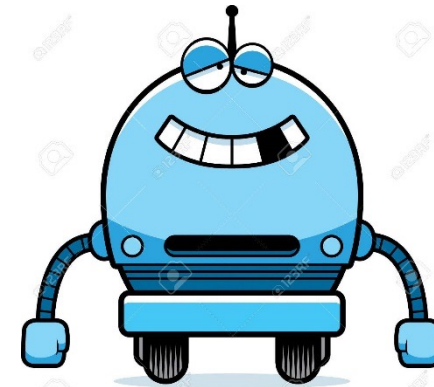
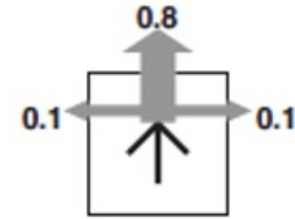
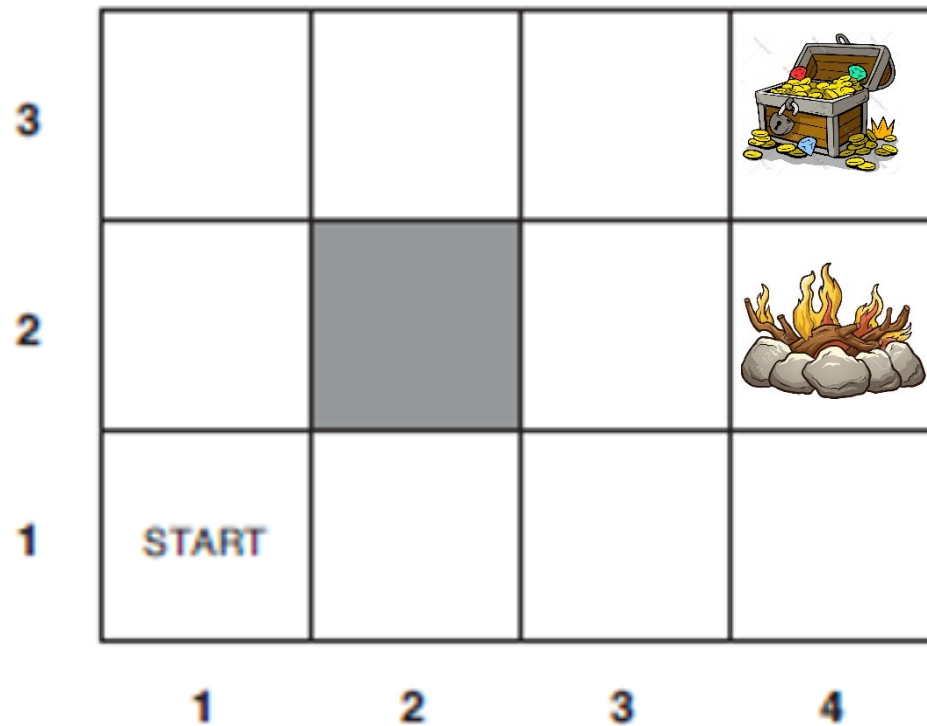
Instructor: Masoumeh Heidari (mheidari2@Alaska.edu)

Sequential decision problems

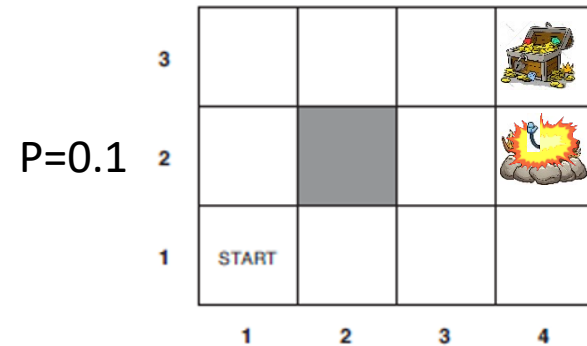
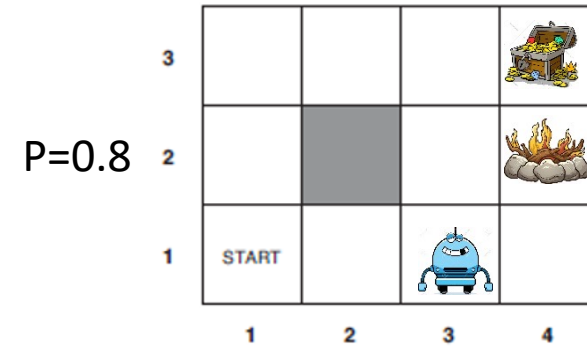
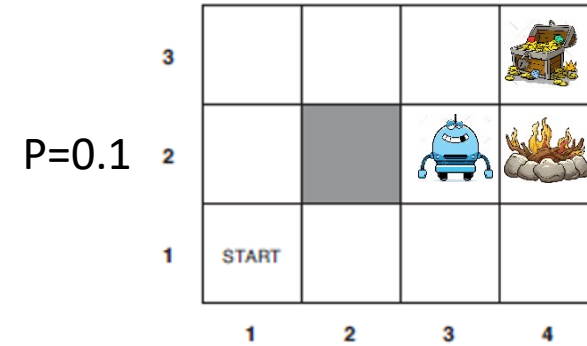
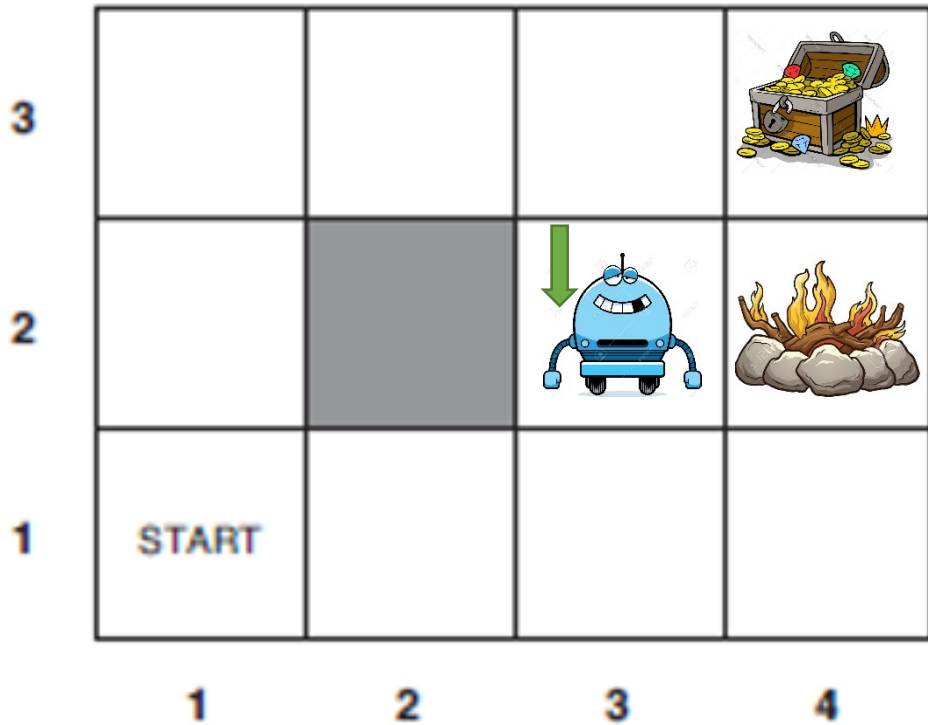
In this chapter, we address the computational issues involved in making decisions in a stochastic environment.



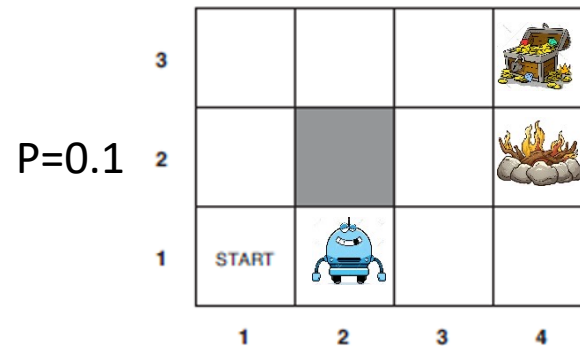
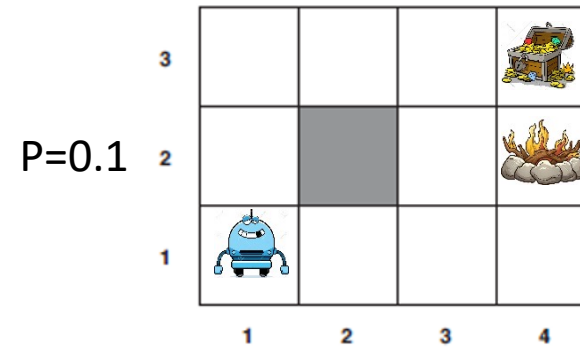
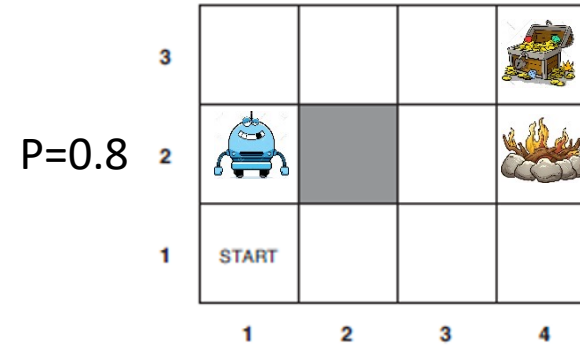
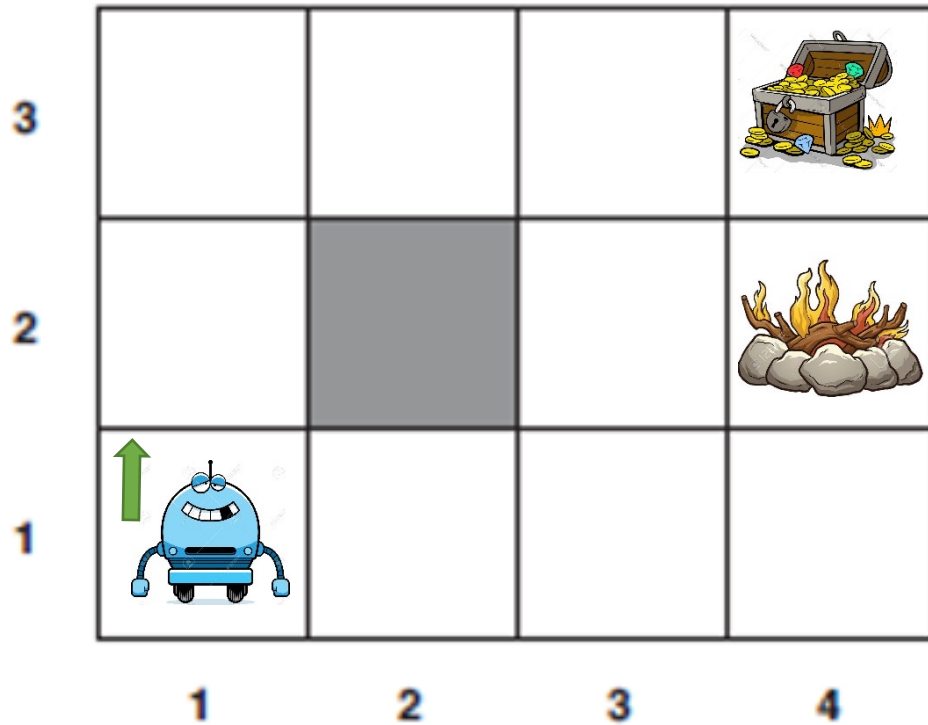
Sequential decision problems



Sequential decision problems

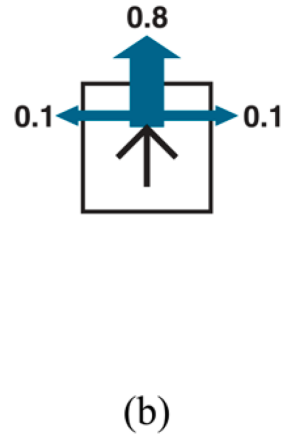
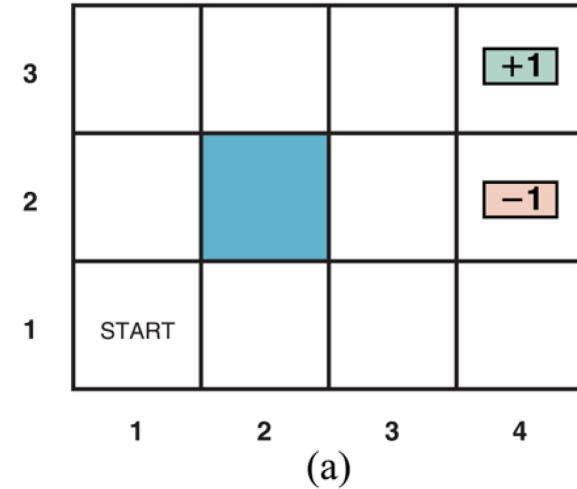


Sequential decision problems



Sequential decision problems

- Suppose that an agent is situated in the 4×3 environment shown in the right.
- Beginning in the start state, it must choose an action at each time step. The interaction with the environment terminates when the agent reaches one of the goal states, marked $+1$ or -1 .
- Just as for search problems, the actions available to the agent in each state are given by $Actions(s)$, sometimes abbreviated to $A(s)$. In this example, $A(s) = \{Up, Down, Left, Right\}$.

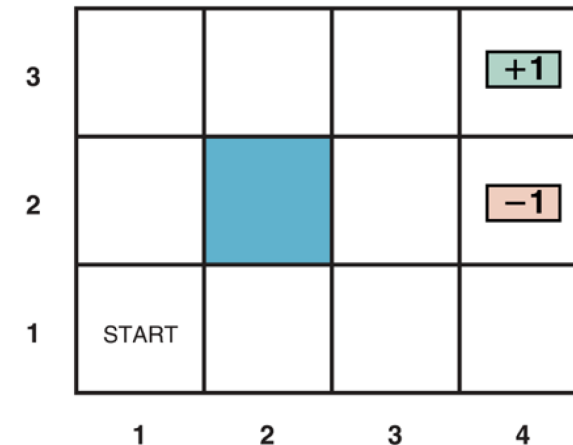


Sequential decision problems

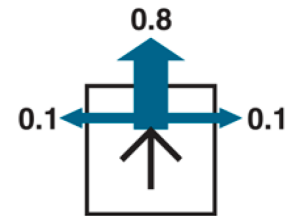
- We assume for now that the environment is fully observable, so that the agent always knows where it is.

- If the environment were deterministic, a solution would be easy:
[Up, Up, Right, Right, Right].

Unfortunately, the environment won't always go along with this solution, because the actions are unreliable.



(a)



(b)











Sequential decision problems

- The transition model (or just “model,” whenever no confusion can arise) describes the outcome of each action in each state.
- $p(s' \mid s, a)$ denotes the probability of reaching state s' if action a is done in state s .
- We will assume that transitions are **Markovian**, that is, the probability of reaching s' from s depends only on s and not on the history of earlier states.
- The agent’s utility function will depend on a **sequence of states**—an environment history—rather than on a single state.

Sequential decision problems

- In each state s , the agent receives a reward $R(s)$, which may be positive or negative, but must be bounded.
- For our particular example, the reward is -0.04 in all states except the terminal states (which have rewards $+1$ and -1).
- The utility of an environment history is just the sum of the rewards received.

Total utility

3	7 	8 	9 	
2	6 			
1	1 	5 	2 	4 
	1	2	3	4

-0.04 -0.24

-0.08 -0.28

-0.12 -0.32

-0.16 0.68

Markov decision process (MDP)

- A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards is called a Markov decision process, or MDP.

- An MDP is defined by:

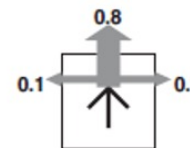
- A set of states $s \in S$

- A set of actions $a \in A$

- A transition function $T(s, a, s')$

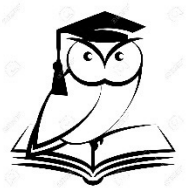
- $P(s' | s, a)$

- A reward function $R(s)$



3	-0.04	-0.04	-0.04	1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

What does a solution to the problem look like?



You may have a good plan but the outcome is not guaranteed!

- We have seen that any fixed action sequence won't solve the problem, because the agent might end up in a state other than the goal.
- Therefore, a solution must specify what the agent should do for any state that the agent might reach.
- A solution of this kind is called a policy. denoted by π , and $\pi(s)$ is the action recommended by the policy π for state s .

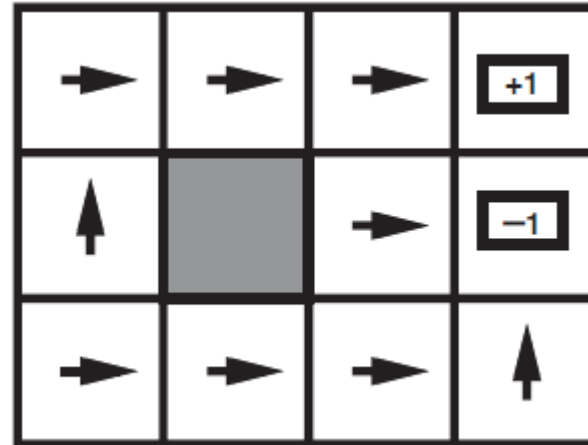
What does a solution to the problem look like?

- If the agent has a complete policy, then no matter what the outcome of any action, the agent will always know what to do next.
- An optimal policy is a policy that yields the highest expected utility, denoted by π^* .
- Given π^* , the agent decides what to do by consulting its current percept, which tells it the current state s , and then executing the action $\pi^*(s)$.

Optimal policies for different reward functions

3				1
2				-1
1				
	1	2	3	4

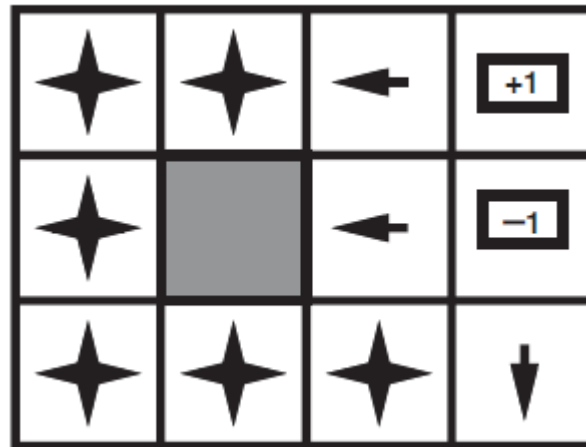
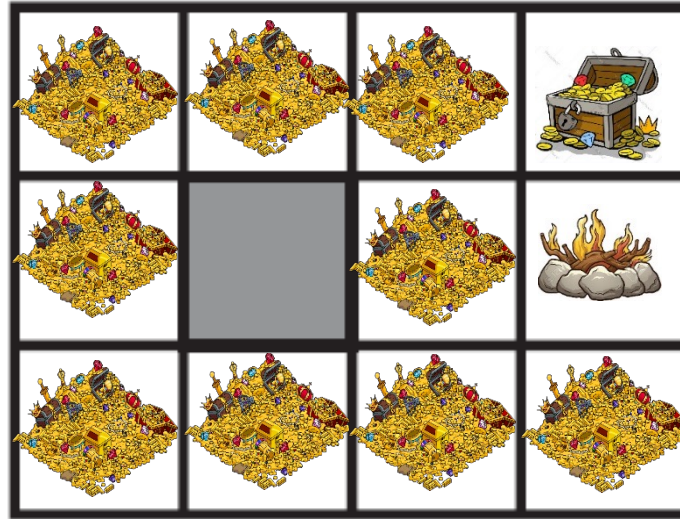
$R(s) = -2$ for nonterminal states.



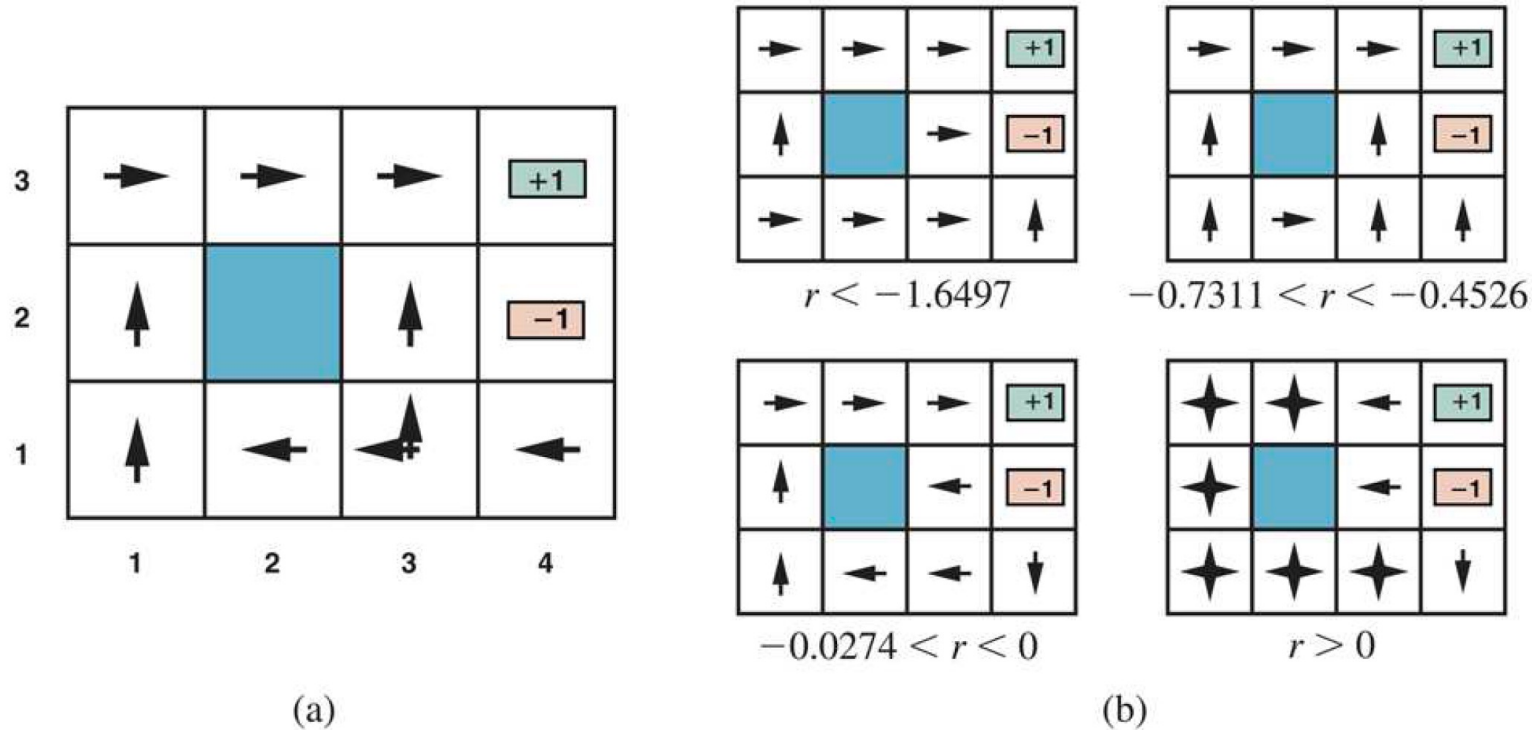
Optimal policies for different reward functions

3				1
2				-1
1				
	1	2	3	4

$R(s) = +2$ for nonterminal states.



Optimal policies for different reward functions



(a) The optimal policies for the stochastic environment with $r = -0.04$ for transitions between nonterminal states. There are two policies because in state (3,1) both *Left* and *Up* are optimal. (b) Optimal policies for four different ranges of r .

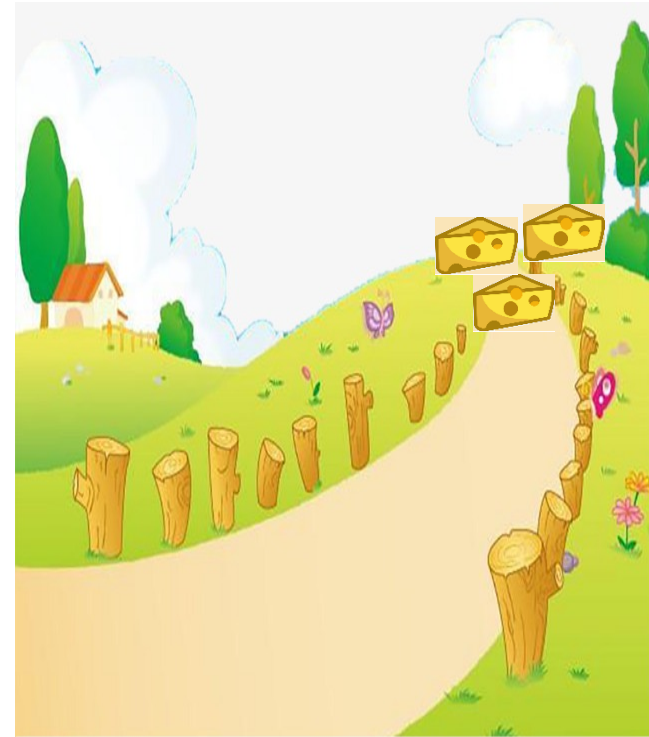
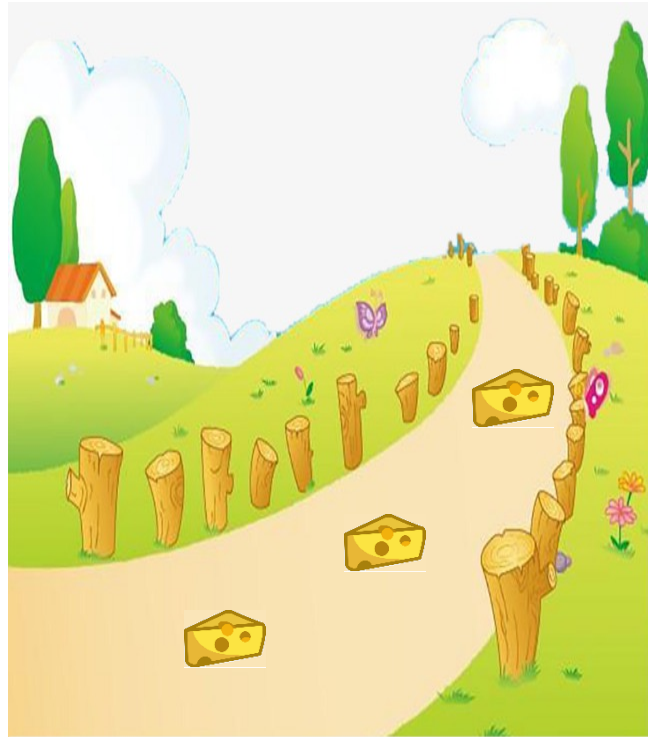
Optimal policies for different reward functions

- The balance of risk and reward changes depending on the value of $r = R(s, a, s')$ for transitions between nonterminal states.
- The introduction of uncertainty brings MDPs closer to the real world than deterministic search problems.
- For this reason, MDPs have been studied in several fields, including AI, operations research, economics, and control theory.

Utilities over time

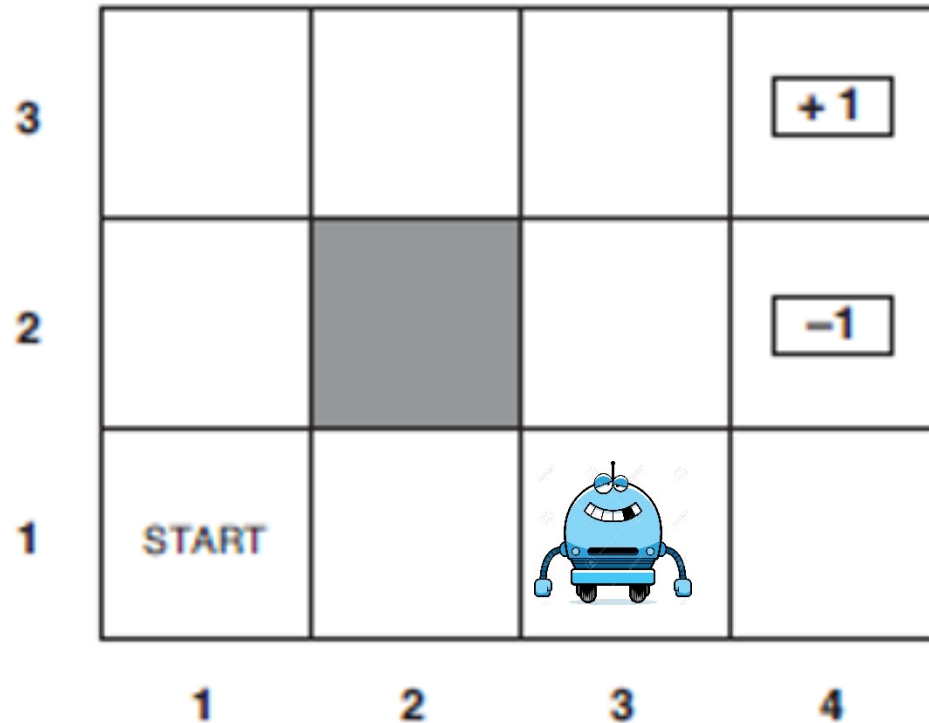


Is sooner better?



Finite Horizon or Infinite Horizon

$R(s) = -0.04$ for nonterminal states.



Maximum number of steps = 3

To have any chance of reaching the +1 state, the agent must head directly for it, and the optimal action is to go *Up*.

Maximum number of steps = 100












Then there is plenty of time to take the safe route by going *Left*.

So, with a finite horizon, the optimal action in a given state could change over time.



How to calculate the utility of state sequences?

Total utility

3	7 	8 	9 	
2	6 			
1	1 	5 	2 	4 
	1	2	3	4

-0.04	-0.24
-0.08	-0.28
-0.12	-0.32
-0.16	0.68

$$U_h([s_0, s_1, s_2, \dots]) =$$

$$R(s_0) + R(s_1) + R(s_2) + \dots =$$

$$R(s_0, a_0, s_1) + R(s_1, a_1, s_2) + R(s_2, a_2, s_3) + \dots$$

How to calculate the utility of state sequences?

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) = R(s_0, a_0, s_1) + R(s_1, a_1, s_2) + R(s_2, a_2, s_3) + \dots$$



1



$0 < \gamma < 1$

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots$$



γ^2

Additive discounted rewards

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots$$

- The discount factor describes the preference of an agent for current rewards over future rewards.
- When γ is close to 0, rewards in the distant future are viewed as insignificant. When γ is close to 1, an agent is more willing to wait for long-term rewards.
- When γ is exactly 1, discounted rewards reduce to the special case of purely additive rewards.

Why additive discounted rewards?

- There are several reasons why additive discounted rewards make sense.
 - **Empirical**: both humans and animals appear to value near-term rewards more highly than rewards in the distant future.
 - **Economic**: if the rewards are monetary, then it really is better to get them sooner rather than later because early rewards can be invested and produce returns while you're waiting for the later rewards.
 - **Uncertainty about the true rewards**: they may never arrive for all sorts of reasons that are not taken into account in the transition model.
 - It conveniently makes some **nasty infinities go away**. With infinite horizons there is a potential difficulty: if the environment does not contain a terminal state, or if the agent never reaches one, then all environment histories will be infinitely long, and utilities with additive undiscounted rewards will generally be infinite.

Infinite Utilities?

- With discounted rewards, the utility of an infinite sequence is finite. In fact if $\gamma < 1$ and rewards are bounded by $\pm R_{max}$, we have

$$U_h([s_0, a_0, s_1, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \sum_{t=0}^{\infty} \gamma^t R_{max} = \frac{R_{max}}{1-\gamma}$$

using the standard formula for the sum of an infinite geometric series.

- If the environment contains terminal states and if the agent is guaranteed to get to one eventually, then we will never need to compare infinite sequences. A policy that is guaranteed to reach a terminal state is called a proper policy. With proper policies, we can use $\gamma = 1$ (i.e., additive undiscounted rewards).

Optimal policies and the utilities of states

- Having decided that the utility of a given history is the sum of discounted rewards, we can compare policies by comparing the **expected utilities** obtained when executing them.
- We assume the agent is in some initial state s and define S_t (a random variable) to be the state the agent reaches at time t when executing a particular policy π .

$$U^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

Optimal policies and the utilities of states

$$U^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, \pi(S_t), S_{t+1})\right]$$

$$U_h([s_0, a_0, s_1, a_1, s_2, \dots]) =$$

$$R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \dots = \sum_{t=0}^{\infty} \gamma^t R(S_t, A_t, S_{t+1})$$

$$\pi_s^* = \operatorname{argmax}_{\pi} U^\pi(s).$$

Note that $U(s)$ and $R(s)$ are quite different quantities; $R(s)$ is the “short term” reward for being in s , whereas $U(s)$ is the “long term” total reward from s onward.

How to choose an action?

- The utility function $U(s)$ allows the agent to select actions by using the principle of maximum expected utility.
- Choose the action that maximizes the reward for the next step plus the expected discounted utility of the subsequent state:

$$\pi^*(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U(s')]$$

Bellman equation

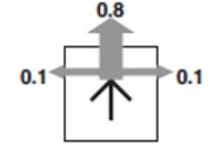
- There is a direct relationship between the utility of a state and the utility of its neighbors:

The utility of a state is the expected reward for the next transition plus the discounted utility of the next state, assuming that the agent chooses the optimal action.

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

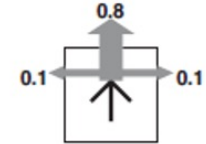
3	-0.04	-0.04	-0.04	+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$


3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

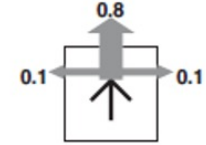
$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4




Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

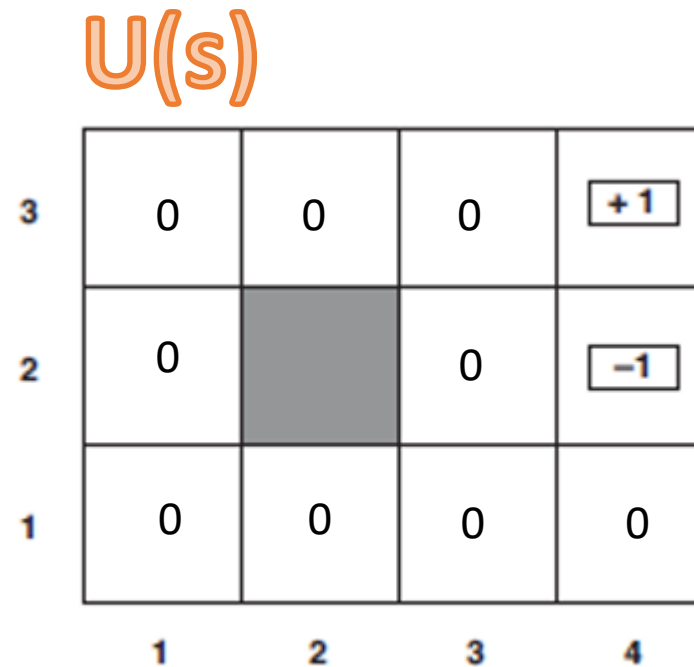
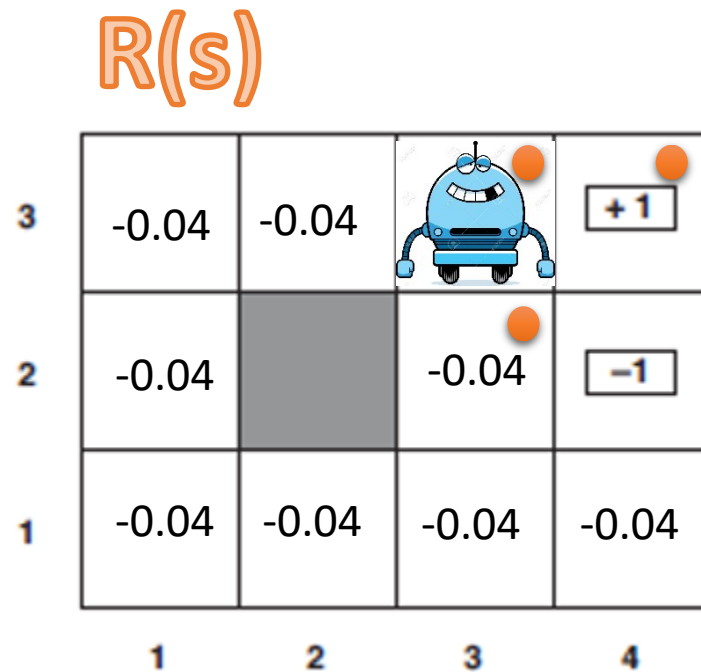
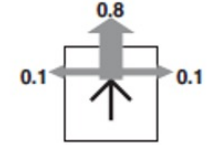
$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4



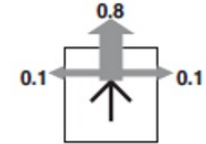
Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

















Example



$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		           
2	-0.04		-0.04	
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	
2	0		0	
1	0	0	0	0
	1	2	3	4

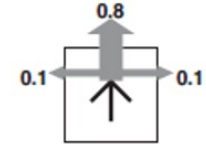


$$0.8 * 1 + 0.1 * 0 + 0.1 * 0 = 0.8$$




Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4



$$0.8 * 1 + 0.1 * 0 + 0.1 * 0 = 0.8$$



$$0.8 * 0 + 0.1 * 1 + 0.1 * 0 = 0.1$$



$$0.8 * 0 + 0.1 * 0 + 0.1 * 1 = 0.1$$

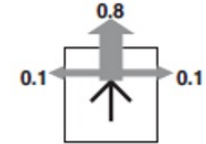


$$0.8 * 0 + 0.1 * 0 + 0.1 * 0 = 0$$




Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4



$$0.8 * 1 + 0.1 * 0 + 0.1 * 0 = 0.8$$



$$0.8 * 0 + 0.1 * 1 + 0.1 * 0 = 0.1$$



$$0.8 * 0 + 0.1 * 0 + 0.1 * 1 = 0.1$$

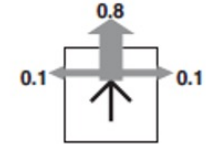


$$0.8 * 0 + 0.1 * 0 + 0.1 * 0 = 0$$




Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04		+1
2	-0.04		-0.04	-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4



$$0.8 * 1 + 0.1 * 0 + 0.1 * 0 = 0.8$$



$$0.8 * 0 + 0.1 * 1 + 0.1 * 0 = 0.1$$



$$0.8 * 0 + 0.1 * 0 + 0.1 * 1 = 0.1$$

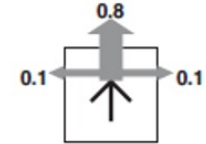


$$0.8 * 0 + 0.1 * 0 + 0.1 * 0 = 0$$


$$U(3,3) = -0.04 + 0.8 = 0.76$$

Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04	-0.04	+1
2	-0.04		-0.04	-1
1		-0.04	-0.04	-0.04
	1	2	3	4

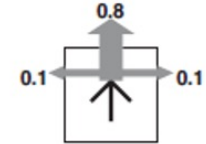
$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4




Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04	-0.04	+1
2	-0.04		-0.04	-1
1		-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

→ $(0.8 * 0 + 0.1 * 0 + 0.1 * 0) = 0$

↑ $(0.8 * 0 + 0.1 * 1 + 0.1 * 0) = 0$

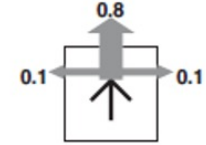
↓ $(0.8 * 0 + 0.1 * 0 + 0.1 * 0) = 0$

← $(0.8 * 0 + 0.1 * 0 + 0.1 * 0) = 0$

$U(1,1) = -0.04 + 0 = -0.04$

Example

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$



$R(s)$

3	-0.04	-0.04	-0.04	+1
2	-0.04			-1
1	-0.04	-0.04	-0.04	-0.04
	1	2	3	4

$U(s)$

3	0	0	0	+1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

→ $0.8 * -1 + 0.1 * 0 + 0.1 * 0 = -0.80$

↑ $0.8 * 0 + 0.1 * -1 + 0.1 * 0 = -0.1$

↓ $0.8 * 0 + 0.1 * 0 + 0.1 * 0 = -0.1$

← $0.8 * 0 + 0.1 * 0 + 0.1 * 0 = 0$

$U(2,3) = -0.04 + 0 = -0.04$

Example

$U =$

-0.0400	-0.0400	0.7600	1.0000
-0.0400	0	-0.0400	-1.0000
-0.0400	-0.0400	-0.0400	-0.0400

$U =$

-0.0800	0.5640	0.8320	1.0000
-0.0480	0	0.4680	-1.0000
-0.0800	-0.0480	-0.0800	-0.0800

$U =$

0.3984	0.6820	0.8900	1.0000
-0.0560	0	0.5256	-1.0000
-0.0912	-0.0560	0.3216	-0.1200

Value Iteration

- The Bellman equation is the basis of the value iteration algorithm for solving MDPs.
- If there are possible states, then there are Bellman equations, one for each state.
- The equations contain unknowns—the utilities of the states.
- There is one problem: the “nonlinear” equations are nonlinear, because the “max” operator is not a linear operator.
- Whereas systems of linear equations can be solved quickly using linear algebra techniques, systems of nonlinear equations are more problematic.

Value Iteration

- One thing to try is an **iterative approach**.
- We start with arbitrary initial values for the utilities, calculate the right-hand side of the equation, and plug it into the left-hand side—thereby updating the utility of each state from the utilities of its neighbors.
- We repeat this until we reach an equilibrium.
- Let $U_i(s)$ be the utility value for state s at the i th iteration. The iteration step, called a Bellman update, looks like this:

$$U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma U_i(s')]$$

Value Iteration

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function

inputs: mdp , an MDP with states S , actions $A(s)$, transition model $P(s' \mid s, a)$,
rewards $R(s, a, s')$, discount γ

ϵ , the maximum error allowed in the utility of any state

local variables: U, U' , vectors of utilities for states in S , initially zero

δ , the maximum relative change in the utility of any state

repeat

$U \leftarrow U'; \delta \leftarrow 0$

for each state s **in** S **do**

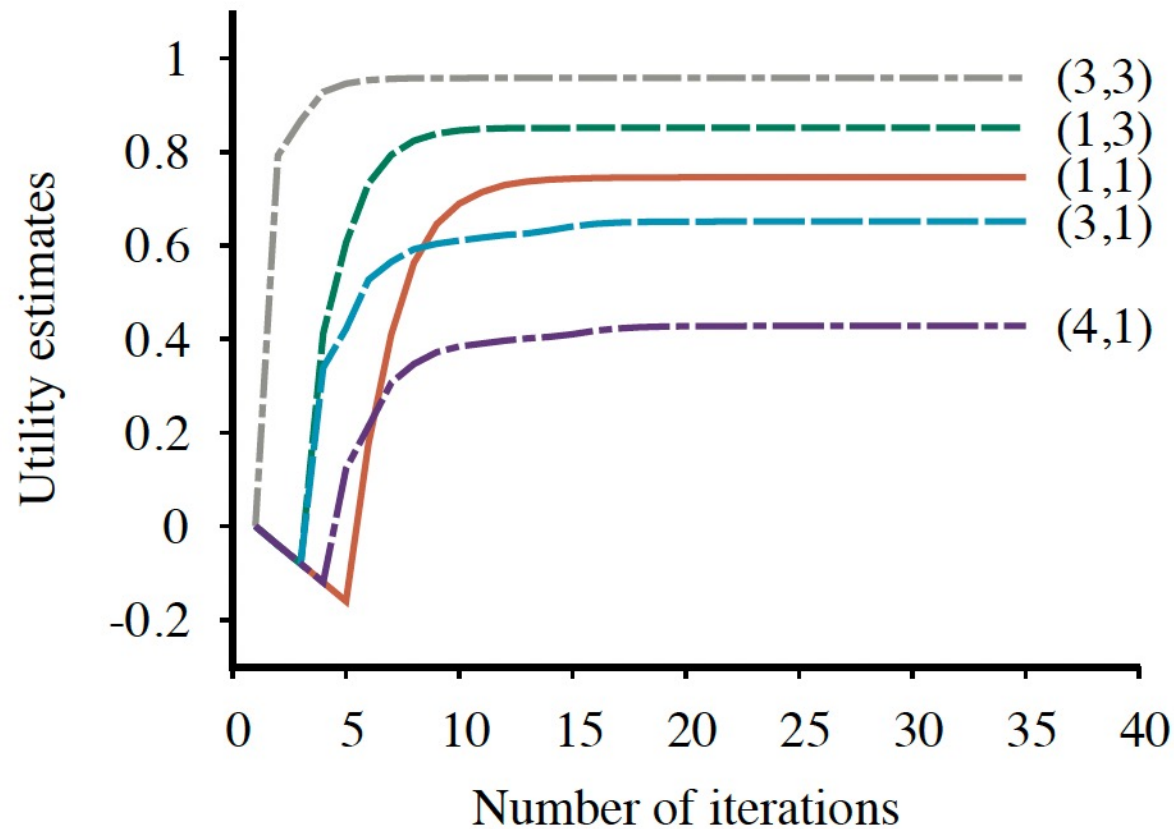
$U'[s] \leftarrow \max_{a \in A(s)} \text{Q-VALUE}(mdp, s, a, U)$

if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$

until $\delta \leq \epsilon(1 - \gamma)/\gamma$

return U

Value Iteration



3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	-1
1	0.7453	0.6953	0.6514	0.4279
	1	2	3	4

Policy Iteration

- The policy iteration algorithm alternates the following two steps, beginning from some initial policy π_0 :
- **Policy evaluation:** given a policy π_i , calculate $U_i = U^{\pi_i}$, the utility of each state if π_i were to be executed.
- **Policy improvement:** Calculate a new MEU policy π_{i+1} , using one-step look-ahead based on U_i .
- Algorithm terminates when the policy improvement step yields no change in utilities.

Policy Iteration

```
function POLICY-ITERATION(mdp) returns a policy
  inputs: mdp, an MDP with states  $S$ , actions  $A(s)$ , transition model  $P(s' | s, a)$ 
  local variables:  $U$ , a vector of utilities for states in  $S$ , initially zero
                    $\pi$ , a policy vector indexed by state, initially random

  repeat
     $U \leftarrow \text{POLICY-EVALUATION}(\pi, U, \textit{mdp})$ 
    unchanged?  $\leftarrow$  true
    for each state  $s$  in  $S$  do
      if  $\max_{a \in A(s)} \sum_{s'} P(s' | s, a) U[s'] > \sum_{s'} P(s' | s, \pi[s]) U[s']$  then do
         $\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s' | s, a) U[s']$ 
        unchanged?  $\leftarrow$  false
  until unchanged?
  return  $\pi$ 
```

Figure 17.7 The policy iteration algorithm for calculating an optimal policy.



Recap

- Sequential decision problems in stochastic environments, also called **Markov decision processes**, or MDPs, are defined by a **transition model** specifying the probabilistic outcomes of actions and a **reward function** specifying the reward in each state.
- The utility of a state sequence is the sum of all the rewards over the sequence, possibly discounted over time. The solution of an MDP is a **policy** that associates a decision with every state that the agent might reach. An optimal policy maximizes the utility of the state sequences encountered when it is executed.
- The utility of a state is the expected sum of rewards when an optimal policy is executed from that state. The **value iteration** algorithm iteratively solves a set of equations relating the utility of each state to those of its neighbors.
- **Policy iteration** alternates between calculating the utilities of states under the current policy and improving the current policy with respect to the current utilities.

Problem formulation example

- Chemotherapy is a common and powerful treatment method in which anticancer drugs are used to destroy cancerous cells.
- Depending upon the type of cancer, the patient may be treated with a single medicine (monotherapy) or a combination of medicines (combination therapy). Today, most successful chemotherapy treatments for advanced cancers use multiple drugs simultaneously.
- Even though chemotherapy is often effective for treating cancer or relieving its symptoms, it causes some side effects. That is because the medicines used in chemotherapy may not distinguish between fast-growing and normal cancerous cells.
- Therefore, advising the best chemotherapy plans which result in both maximum reduction in the number of cancerous cells and minimum side effects in patients is a great challenge an oncologist faces when prescribing chemotherapy drugs.

Bazrafshan, N., & Lotfi, M. M. (2020). A finite-horizon Markov decision process model for cancer chemotherapy treatment planning: an application to sequential treatment decision making in clinical trials. *Annals of Operations Research*, 295(1), 483-502.

Problem formulation example

- Sequential or dynamic treatment regimen is another new challenging topic in the literature. A dynamic treatment regimen is a sequential intervention in which the severity and/or type of treatment varies according to the patient needs.
- Consideration of the patient's response to the prescribed treatment regimens during the chemotherapy treatment period is an important medical process which should not be ignored. During the chemotherapy treatment, acquiring new information about the tumor characteristics and consequent toxicities in patients may affect the subsequent chemotherapy decisions on drug type and dosage in later stages.

Bazrafshan, N., & Lotfi, M. M. (2020). A finite-horizon Markov decision process model for cancer chemotherapy treatment planning: an application to sequential treatment decision making in clinical trials. *Annals of Operations Research*, 295(1), 483-502.



Problem formulation example

MDP Component	Definition
State	Patient toxicity level
Action	Binary variable indicating whether drug i is a part of optimal chemotherapy regimen or not.
Transition probability	Probability of transiting patient toxicity level from s at current treatment cycle t to s' though selecting treatment regimen a
Cost/reward function	Cost of applying treatment regimen a in treatment cycle t when patient toxicity level is s .

Bazrafshan, N., & Lotfi, M. M. (2020). A finite-horizon Markov decision process model for cancer chemotherapy treatment planning: an application to sequential treatment decision making in clinical trials. *Annals of Operations Research*, 295(1), 483-502.

Problem formulation example 2

- The fast growing population of the ageing society will result in a dramatic increase in the number of people diagnosed with cognitive disabilities (such as Alzheimer's disease or other forms of dementia). People with cognitive disabilities suffer from memory loss and executive function impairment.
- To support independent living and reduce the cost of health care, researchers have developed various technologies and computing systems that can automate the prompting behavior and alleviate the burden on care-givers. This could be electronic devices that provide timely prompts and reminders to support schedule adherence and time management.

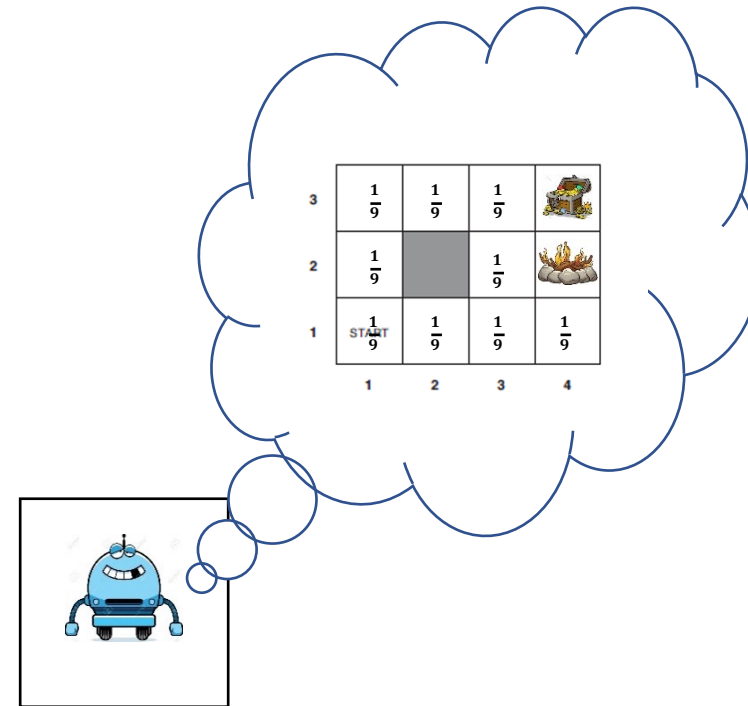
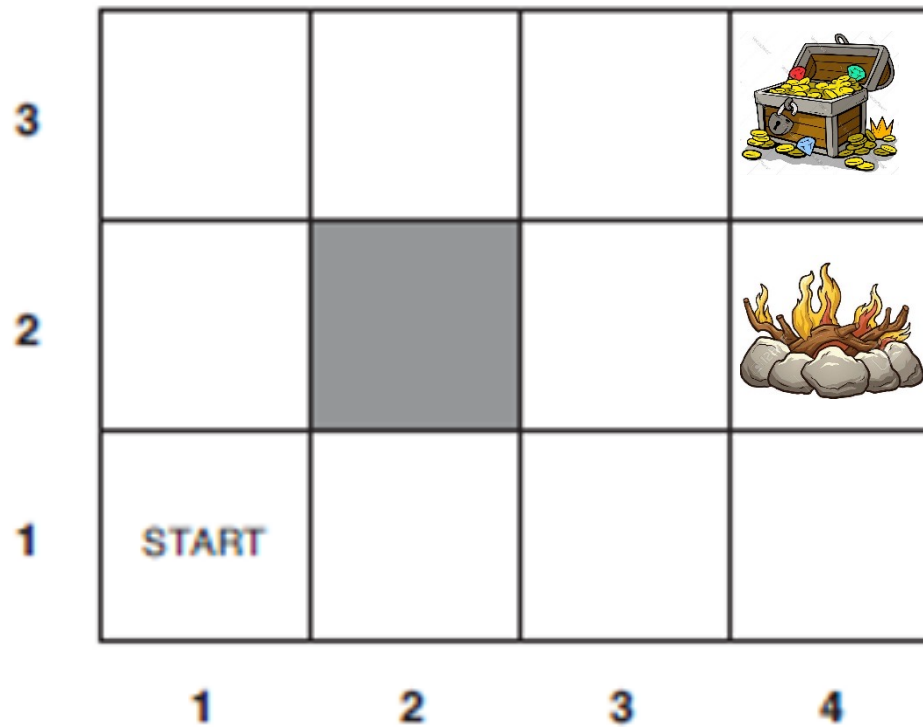
Chu, Yi, et al. "Interactive activity recognition and prompting to assist people with cognitive disabilities." Journal of Ambient Intelligence and Smart Environments 4.5 (2012): 443-459.

Problem formulation example 2










- To perform well an intelligent prompting system must be able to infer the state of the world, reason about the costs of varying system actions, handle uncertainties about the environment and the user, and to adapt to the user behavior pattern.
- The goals of the system are to ensure that the user adheres to a daily schedule by providing prompts to begin, resume, or end activities, to create logs of the user's activities, and to minimize interruptions to the user.

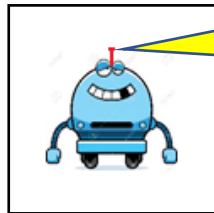
Chu, Yi, et al. "Interactive activity recognition and prompting to assist people with cognitive disabilities." *Journal of Ambient Intelligence and Smart Environments* 4.5 (2012): 443-459.

Where Am I?









Where Am I?

3				
2				
1	 S O U T			
	1	2	3	4












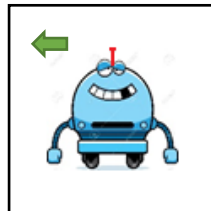
Number of adjacent walls = 2




3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	
2	$\frac{1}{9}$		$\frac{1}{9}$	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
	1	2	3	4




3	$\frac{1}{6}$	$\frac{1}{6}$	0	
2	$\frac{1}{6}$		0	
1	$\frac{1}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
	1	2	3	4

Where Am I?











3				
2				
1				
	1	2	3	4

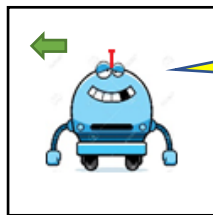


3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	
2	$\frac{1}{9}$		$\frac{1}{9}$	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
	1	2	3	4




3	$\frac{1}{6}$	$\frac{1}{6}$	0	
2	$\frac{1}{6}$		0	
1	$\frac{1}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
	1	2	3	4




Where Am I?

3				
2				
1				
	1	2	3	4






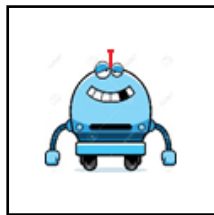
Number of adjacent walls = 1



3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	
2	$\frac{1}{9}$		$\frac{1}{9}$	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
	1	2	3	4



3	$\frac{1}{6}$	$\frac{1}{6}$	0	
2	$\frac{1}{6}$		0	
1	$\frac{1}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
	1	2	3	4



Where Am I?

3				
2				
1	START			
	1	2	3	4



3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	
2	$\frac{1}{9}$		$\frac{1}{9}$	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
	1	2	3	4

3	$\frac{1}{6}$	$\frac{1}{6}$	0	
2	$\frac{1}{6}$		0	
1	$\frac{1}{6}$	$\frac{1}{6}$	0	$\frac{1}{6}$
	1	2	3	4

3	0	0	0	
2	0		0	
1	0	0	≈ 1	0
	1	2	3	4

(Partially Observable)MDPs

MDPs	POMDPs
A set of states $s \in S$	A set of states $s \in S$
A set of actions $a \in A$	A set of actions $a \in A$
A transition function $T(s, a, s')$ $P(s' s, a)$	A transition function $T(s, a, s')$ $P(s' s, a)$
A reward function $R(s)$	A reward function $R(s)$
	A sensor model $P(e s)$

Partially Observable MDPs (POMDPs)

- In a partially observable environment:
 - The agent does not necessarily know which state it is in, so it cannot execute the action $\pi(s)$ recommended for that state.
 - The utility of a state s and the optimal action in s depend not just on s , but also on how much the agent knows when it is in s .
- A POMDP has the following elements:
 - The transition model $P(s' | s, a)$,
 - Actions $A(s)$,
 - Reward function $R(s)$, and
 - A sensor model $P(e | s)$.
- In POMDPs, the belief state b is a probability distribution over all possible states.
- $b(s)$ is the probability assigned to the actual state s by belief state b .

Partially Observable MDPs (POMDPs)



- The agent can calculate its current belief state as the conditional probability distribution over the actual states given the sequence of percepts and actions so far. (filtering)
- If $b(s)$ was the previous belief state, and the agent does action a and then perceives evidence e , then the new belief state is given by

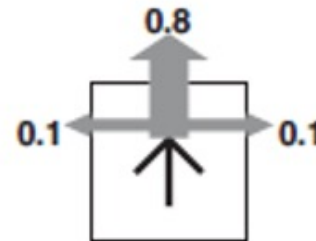
$$b'(s') = \alpha P(e | s') \sum_s P(s' | s, a) b(s)$$
$$b' = \text{FORWARD}(b, a, e)$$



Partially Observable MDPs (POMDPs)

$$b'(s') = \alpha P(e | s') \sum_s P(s' | s, a) b(s)$$

Number of adjacent walls = 1

3	0	0	$\frac{1}{3}$	
2	0		$\frac{1}{3}$	
1	0	0	$\frac{1}{3}$	0
	1	2	3	4



3	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	
2	$\frac{1}{9}$		$\frac{1}{9}$	
1	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
	1	2	3	4



Partially Observable MDPs (POMDPs)



The optimal action depends only on the agent's current belief state.

$$MDP: a = \pi^*(s)$$

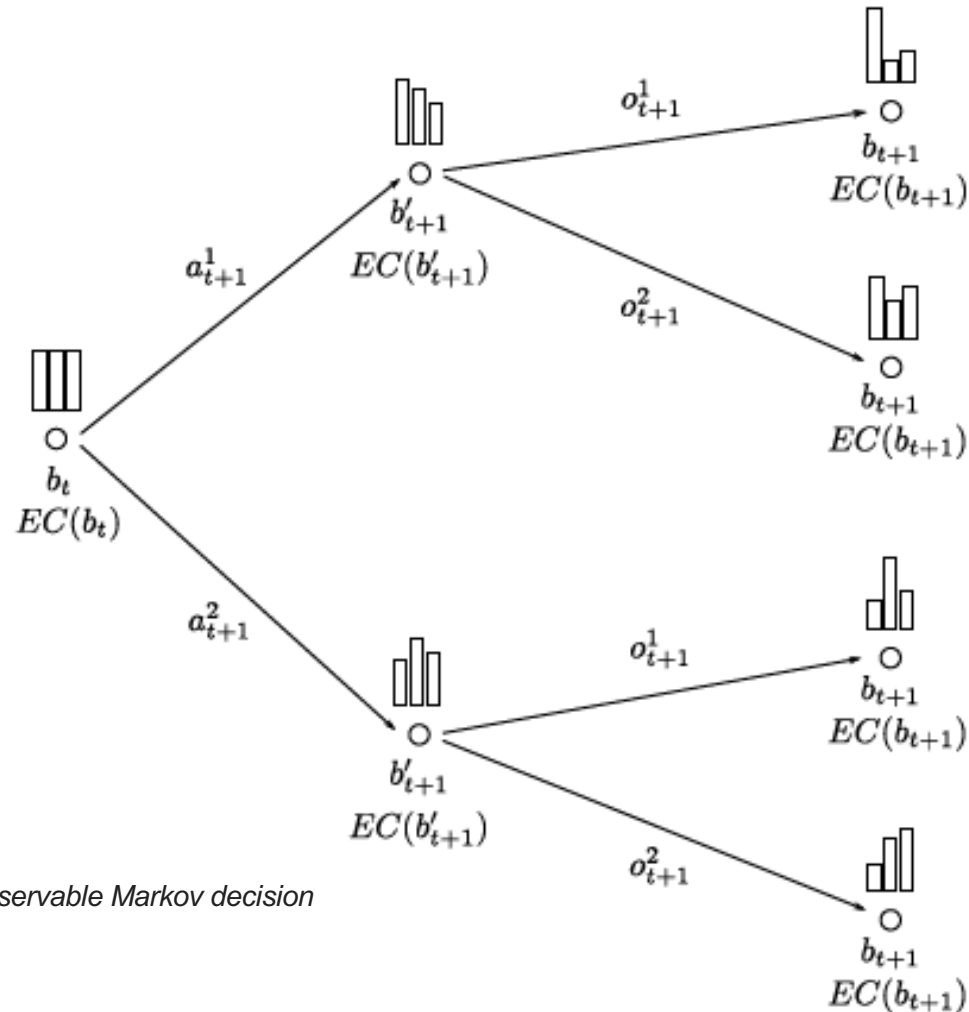
$$POMDP: a = \pi^*(b)$$

Partially Observable MDPs (POMDPs)

- In POMDPs, the optimal action depends only on the agent's current belief state. That is, the optimal policy can be described by a mapping $\pi^*(b)$ from belief states to actions.
- The decision cycle of a POMDP agent:
 - Given the current belief state b , execute the action $a = \pi^*(b)$.
 - Receive percept e .
 - Set the current belief state to $FORWARD(b, a, e)$ and repeat.
- Now we can think of POMDPs as requiring a search in belief-state space.
 - The POMDP belief-state space is continuous, because the belief state is a probability distribution.

Partially Observable MDPs (POMDPs)

$$\rho(b) = \sum_s b(s)R(s)$$

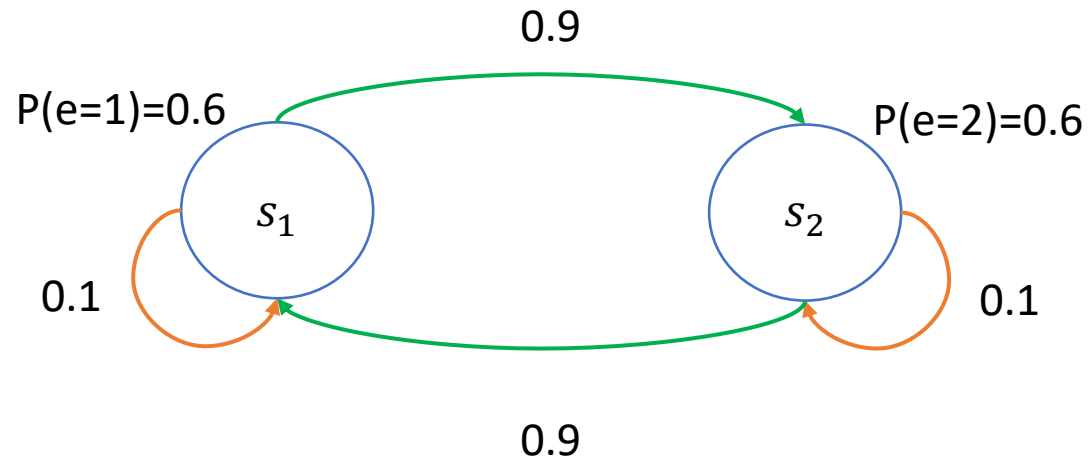


Woodward, Mark P. *Framing human-robot task communication as a partially observable Markov decision process*. Diss. 2012.

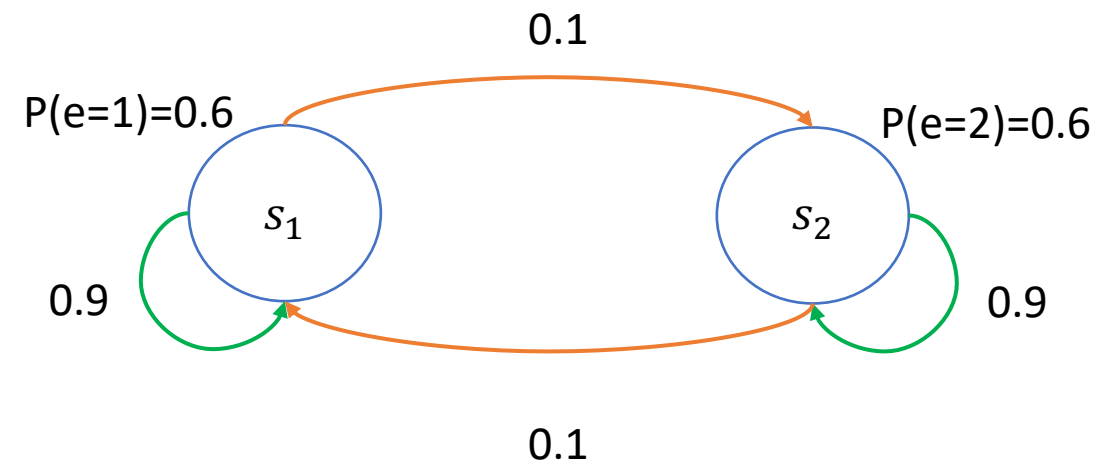
Example

- An example two-state world:
 - **States:** 1 and 2
 - **R(1)=0, R(2)=1**
 - **Two actions:** *Stay* stays put with probability 0.9 and *Go* switches to the other state with probability 0.9.
 - **Discount factor $\gamma = 1$.**
 - The sensor reports the correct state with probability 0.6.
 - Obviously, the agent should *Stay* when it thinks it's in state 2 and *Go* when it thinks it's in state 1.
 - Belief space is one-dimensional as $b(1)+b(2)=1$.

Example

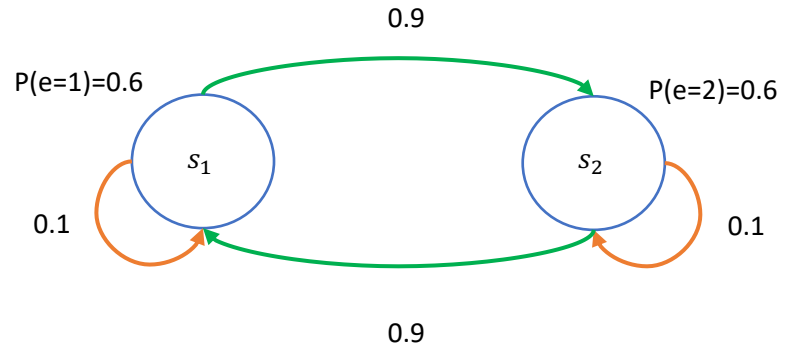


Action: Go

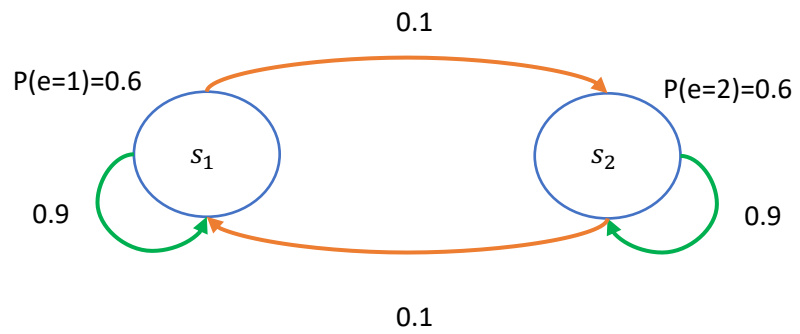


Action: Stay

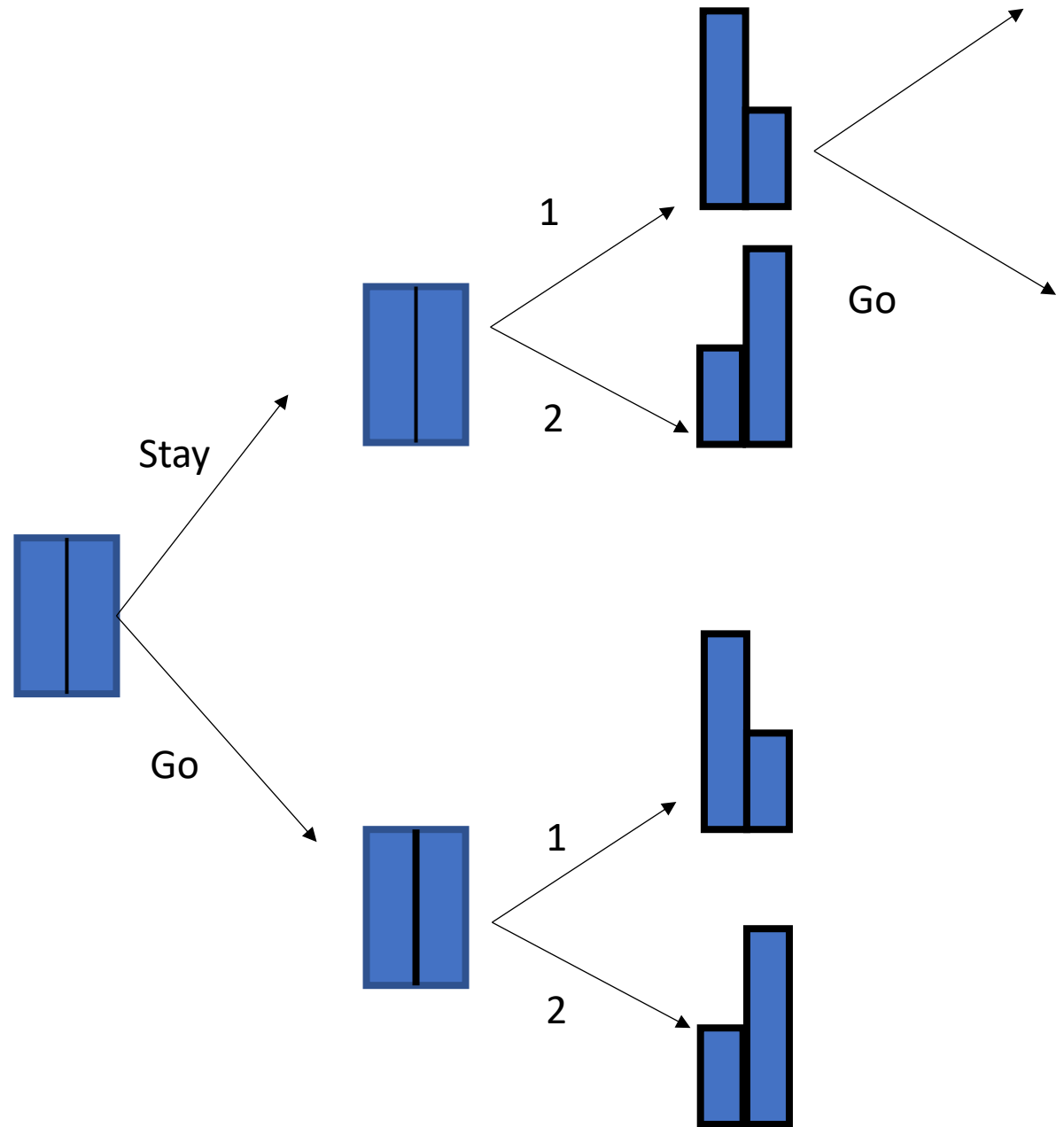
Example



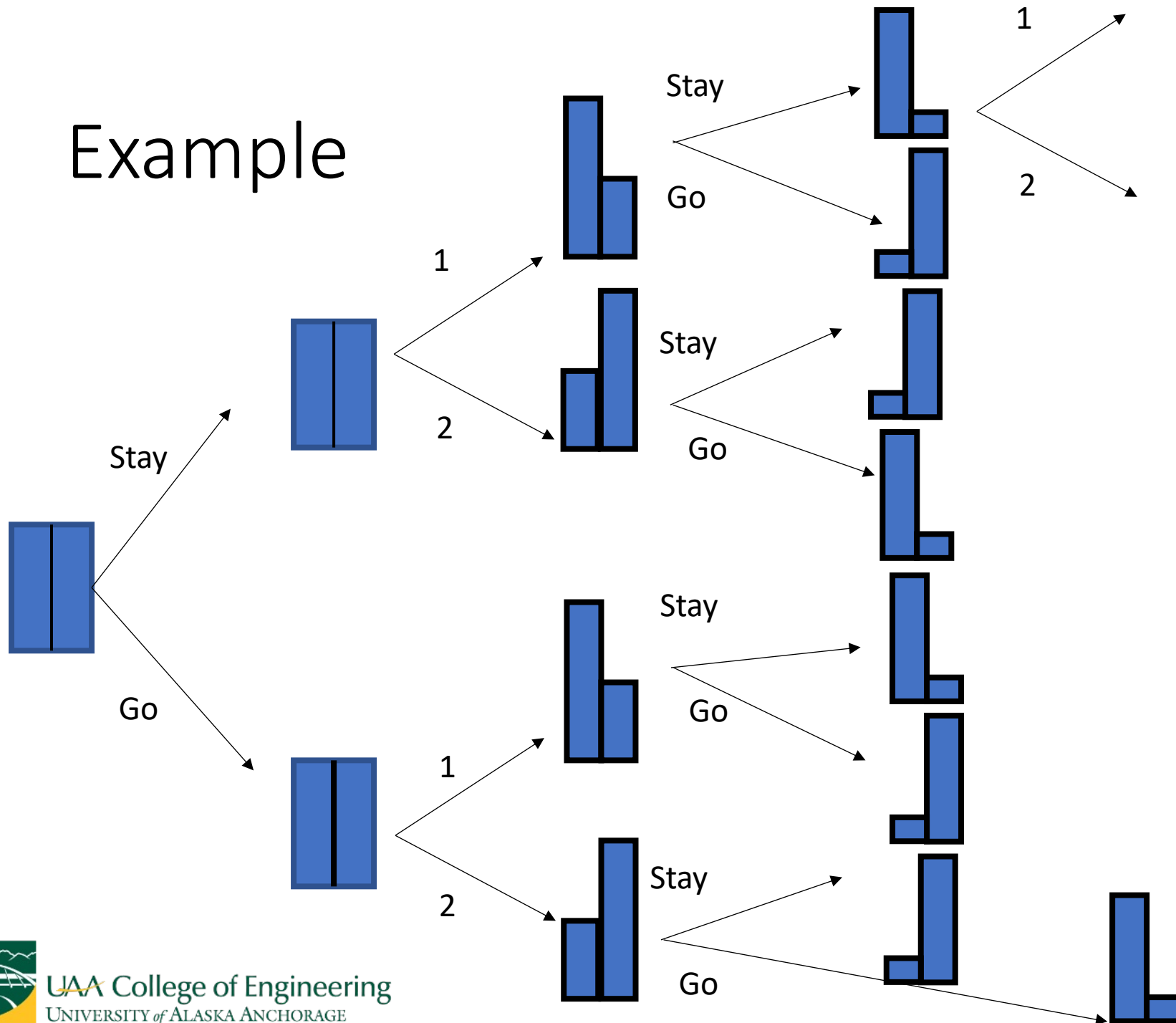
Action: Go



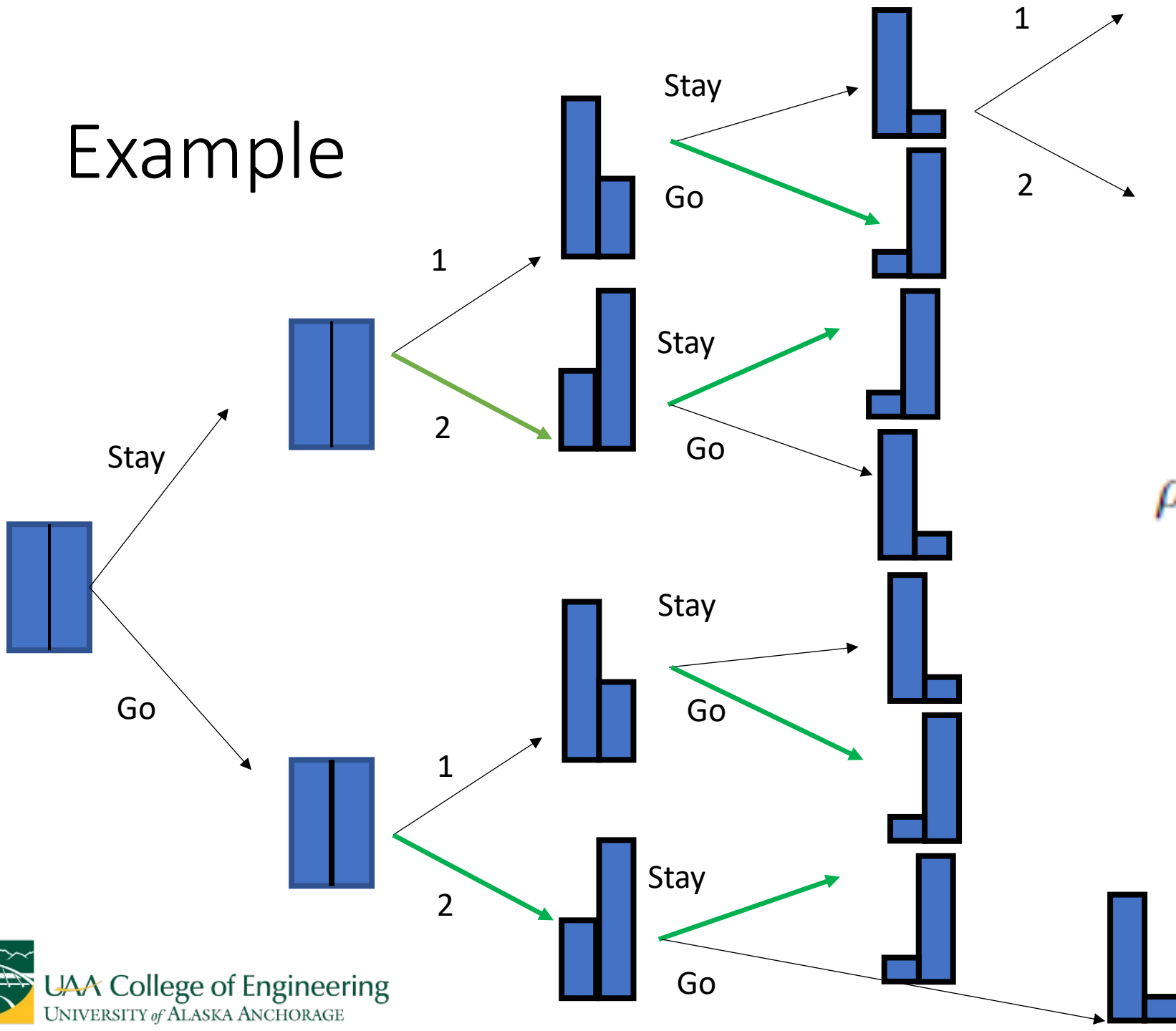
Action: Stay



Example



Example



$$\rho(b) = \sum_s b(s)R(s)$$

Example

The diagram illustrates a game tree structure. It begins with a root node (a blue rectangle with a vertical line). From this node, two branches emerge: one labeled "Stay" leading to a node with two vertical lines, and one labeled "Go" leading to another node with two vertical lines. Each of these nodes further branches into two nodes, labeled "1" and "2". This pattern continues, with nodes branching into "Stay" and "Go" paths, and further into "1" and "2" paths. The diagram shows a sequence of decisions and chance events, ending with a final node on the right.

UAA College of Engineering
UNIVERSITY of ALASKA ANCHORAGE

What should I do?