Markov Decision Processes: Making Decision in the Presence of Uncertainty

(some of) R&N 16.1-16.6

R&N 17.1-17.4

Different Aspects of “Machine Learning”

- Supervised learning
  - Classification - concept learning
  - Learning from labeled data
  - Function approximation
- Unsupervised learning
  - Data is not labeled
  - Data needs to be *grouped*, *clustered*
  - We need distance metric
- Control and action model learning
  - Learning to select actions efficiently
  - Feedback: goal achievement, failure, *reward*
  - Control learning, reinforcement learning

Decision Processes: General Description

- Suppose you have to make decisions that impact your future... You know the current state around you.
- You have a choice of several possible actions.
- You cannot predict with certainty the consequence of these actions given the current state, but you have a guess as to the likelihood of the possible outcomes.
- How can you define a policy that will guarantee that you always choose the action that maximizes *expected* future profits?

Note: Russel & Norvig, Chapter 17.

Decision Processes: General Description

- Decide what action to take next, given:
  - A probability to move to different states
  - A way to evaluate the reward of being in different states

Robot path planning

Travel route planning

Elevator scheduling

Aircraft navigation

Manufacturing processes

Network switching & routing

Example

- Assume that time is discretized into *discrete time steps*
- Suppose that your world can be in one of a finite number of *states s*
  - this is a major simplification, but let's assume….
- Suppose that for every state *s*, we can anticipate a reward that you receive for being in that state *R(s).*
- Assume also that *R(s)* is bounded (*R(s)* < *M* for all s) meaning that there is a threshold in reward.

- Question: What is the total value of the reward for a particular configuration of states $\{s_1, s_2, \ldots\}$ over time?

Example

- Question: What is the total value of the reward for a particular configuration of states $\{s_1, s_2, \ldots\}$ over time?
- It is simply the sum of the rewards (possibly negative) that we will receive in the future:

$$U(s_1, s_2, .., s_n, ..) = R(s_1) + R(s_2) + .. + R(s_n) + ...$$

What is wrong with this formula???

Horizon Problem

$$U(s_0,\ldots, s_N) = R(s_0)+R(s_1)+\ldots+R(s_N)$$

The sum may be arbitrarily large depending on *N*

Need to know *N*, the length of the sequence (finite horizon)

Horizon Problem

- The problem is that we did not put any limit on the "future", so this sum can be infinite.
- For example: Consider the simple case of computing the total future reward if you remain forever in the same state:

$$U(s,s,.., s,..) = R(s)+R(s)+..+ R(s) +......$$

is clearly infinite in general!!

- This definition is useless unless we consider a finite time horizon.
- But, in general, we don't have a good way to define such a time horizon.

Discounting

$$U(s_0, \ldots.) = R(s_0) + \gamma R(s_1) + .. + \gamma^N R(s_N) + ..$$

The length of the sequence is arbitrary (infinite horizon)

Discount factor $0 < \gamma < 1$

Discounting

- $U(s_0,\ldots..) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots$
- Always converges if $\gamma < 1$ and R(.) is bounded
- γ close to 0 → instant gratification, don't pay attention to future reward
- γ close to 1 → extremely conservative, big influence of the future
- The resulting model is the *discounted reward*
  - Prefers expedient solutions (models impatience)
  - Compensates for uncertainty in available time (models mortality)
- Economic example:
  - Being promised $10,000 next year is worth only 90% as much as receiving $10,000 right now.
  - Assuming payment *n* years in future is worth only $(0.9)^n$ of payment now

Actions

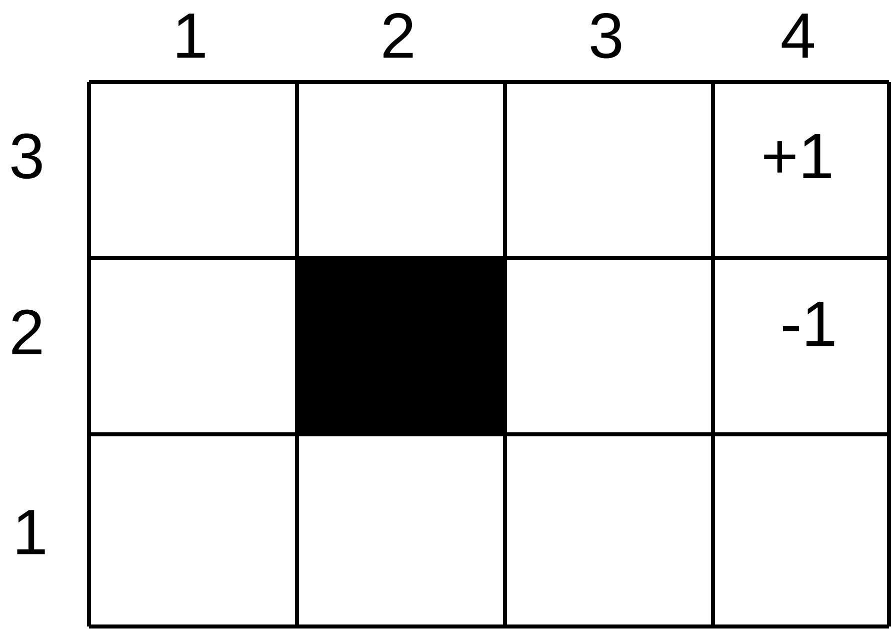
- Assume that we also have a finite set of actions *a*
- An action *a* causes a transition from a state *s* to a state *s'*

The Basic Decision Problem

- Given:
  - Set of states $S = \{s\}$
  - Set of actions $A = \{a\}$ $a: S \rightarrow S$
  - Reward function R(.)
  - Discount factor γ
  - Starting state s_1
- Find a sequence of *actions* such that the resulting sequence of states *maximizes* the total discounted reward:

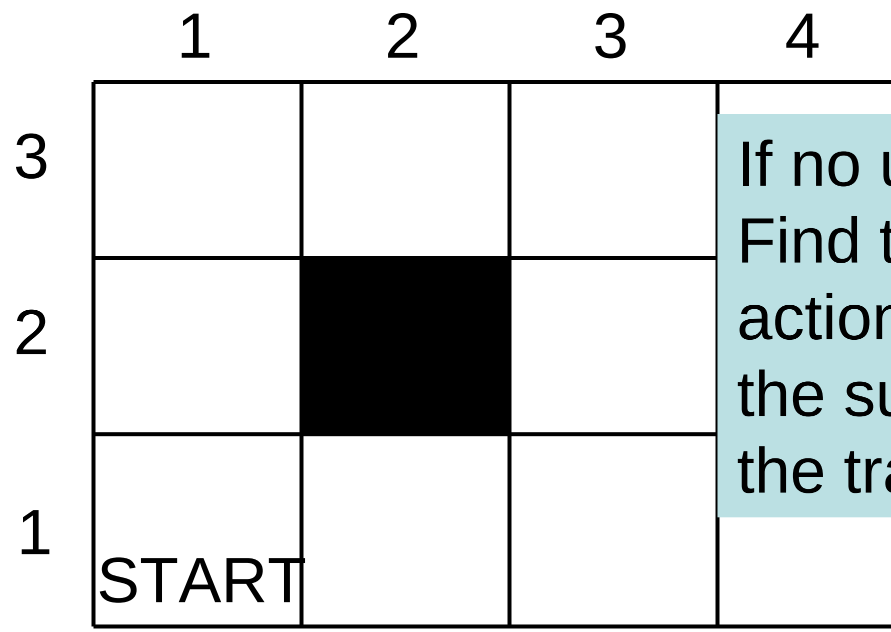
$$U(s_0,\ldots.)=R(s_0)+\gamma R(s_1)+..+\gamma^N R(s_N)+..$$

Maze Example: Utility



- Define the reward of being in a state:
  - $R(s) = -0.04$ if s is empty state
  - $R(4,3) = +1$ (maximum reward when goal is reached)
  - $R(4,2) = -1$ (avoid (4,2) as much as possible)
- Define the utility of a sequence of states:
  - $U(s_0,\ldots, s_N) = R(s_0) + R(s_1) + \ldots + R(s_N)$

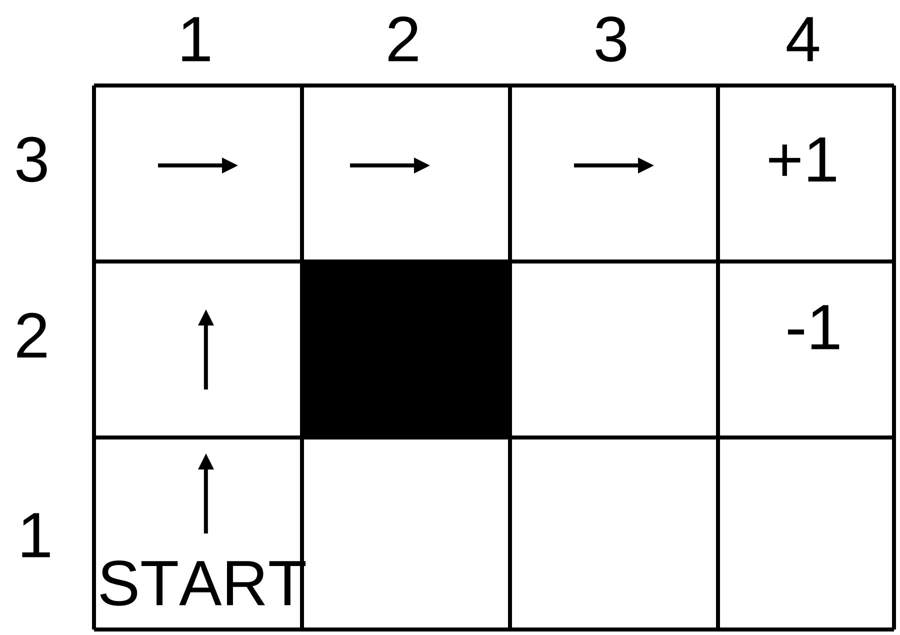
Maze Example: Utility



If no uncertainty:
Find the sequence of actions that maximizes the sum of the rewards of the traversed states

- Define the reward of being in a state:
  - $R(s) = -0.04$ if s is empty state
  - $R(4,3) = +1$ (maximum reward when goal is reached)
  - $R(4,2) = -1$ (avoid (4,2) as much as possible)
- Define the utility of a sequence of states:
  - $U(s_0,\ldots, s_N) = R(s_0) + R(s_1) + \ldots + R(s_N)$

Maze Example: No Uncertainty



- States: locations in maze grid
- Actions: Moves up/left left/right
- If no uncertainty: Find sequence of actions from current state to goal (+1) that maximizes utility → We know how to do this using earlier search techniques

What we are looking for: *Policy*

- ***Policy*** = Mapping from states to action $\pi(s) = a$
  → Which action should be taken in each state
- In the maze example, $\pi(s)$ associates a motion to a particular location on the grid
- For any state s, the utility $U(s)$ of s is the sum of discounted rewards of the sequence of states starting at s generated by using the policy π

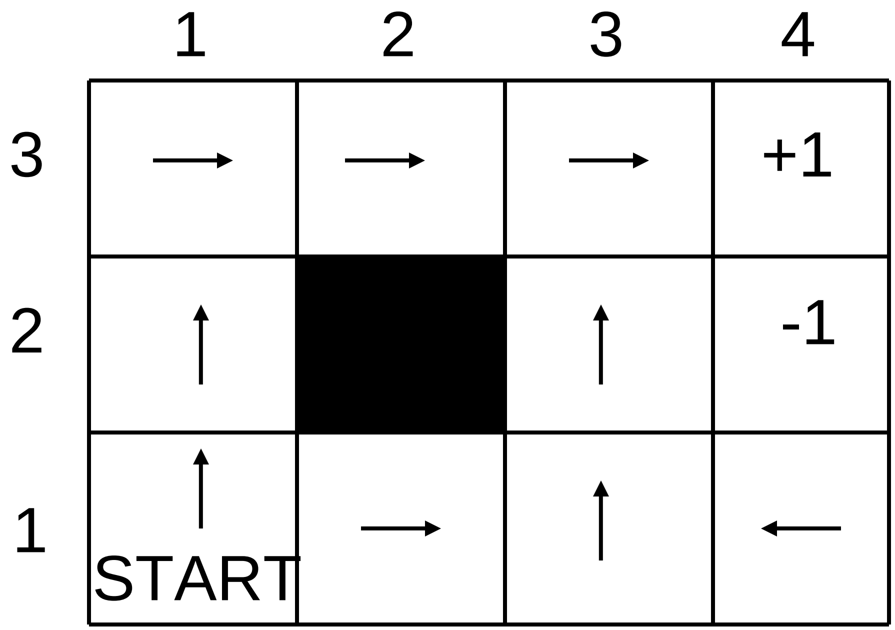
$$U(s) = R(s) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots..$$

- Where we move from s to s1 by action $\pi(s)$
- We move from s_1 to s_2 by action $\pi(s_1)$, etc.

Optimal Decision Policy

- *Policy*
  - Mapping from states to action $\pi(s) = a$
- ***Optimal Policy***
  - The policy π^* that maximizes the expected utility $U(s)$ of the sequence of states generated by π^*, starting at s
- In the maze example, $\pi^*(s)$ tells us which motion to choose at every cell of the grid to bring us closer to the goal

Maze Example: No Uncertainty



- π*((1,1)) = UP
- π*((1,3)) = RIGHT
- π*((4,1)) = LEFT

Maze Example: With Uncertainty

Intended action: ↑

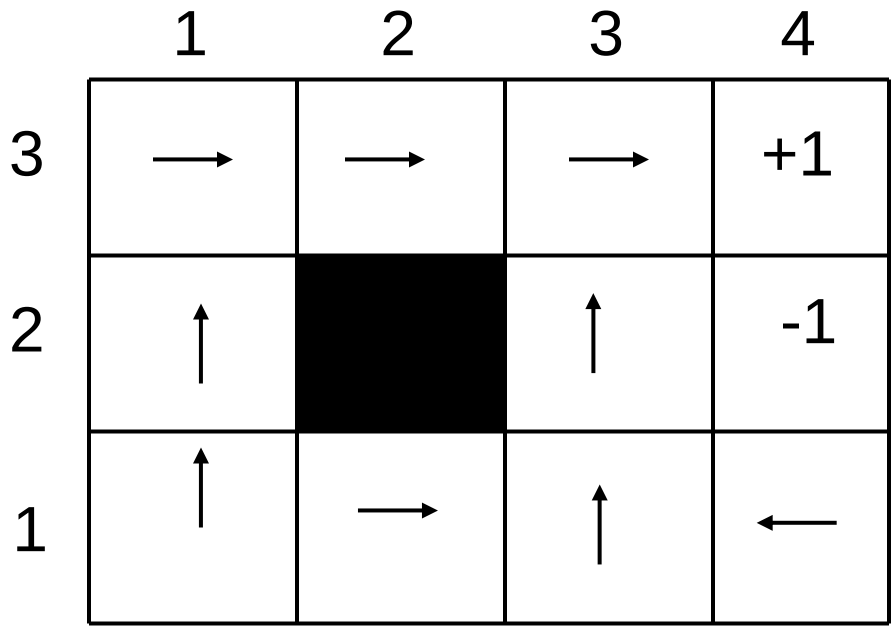
Executed action:	↑	↓	←	→
	Prob = 0.8	Prob = 0.0	Prob = 0.1	Prob = 0.1

- The robot may not execute exactly the action that is commanded → The outcome of an action is no longer deterministic
- Uncertainty:
  - We know in which state we are (fully observable)
  - But we are not sure that the commanded action will be executed exactly

Uncertainty

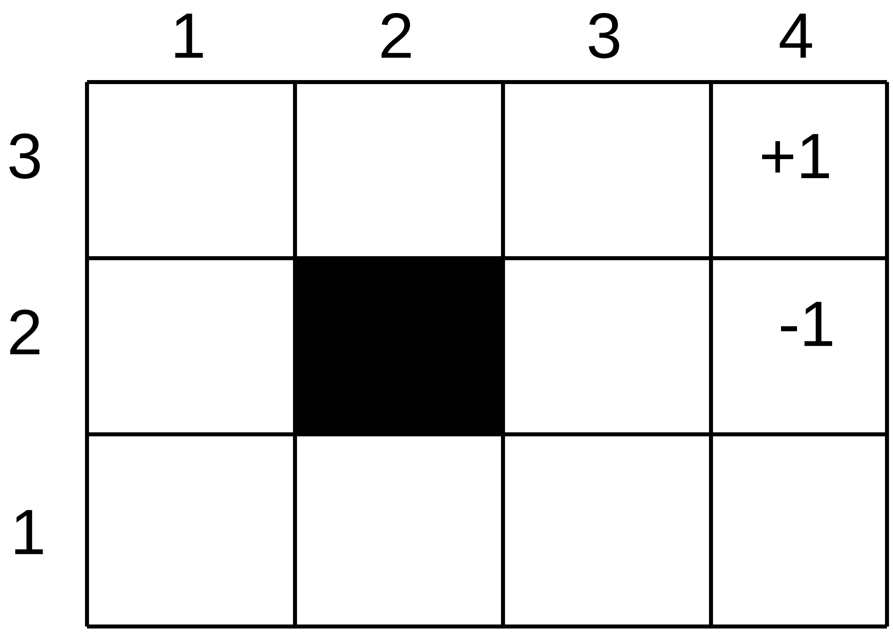
- No uncertainty:
  - An action *a* *deterministically* causes a transition from a state *s* to another state *s'*
- With uncertainty:
  - An action *a* causes a transition from a state *s* to another state *s'* with *some probability* $T(s,a,s')$
  - $T(s,a,s')$ is called the transition probability from state *s* to state *s'* through action *a*
  - In general, we need $|S|^2 \mathrm{x} |A|$ numbers to store all the transitions probabilities

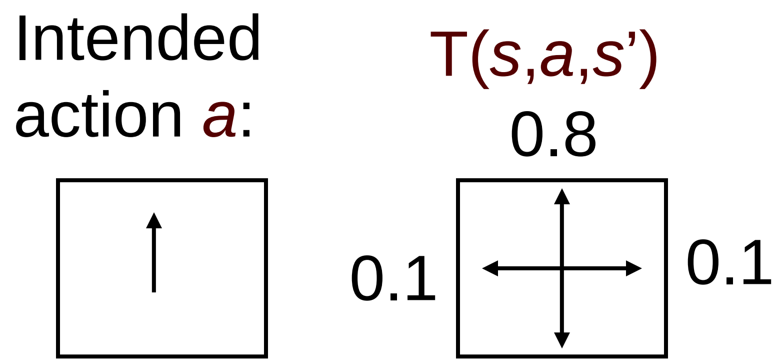
Maze Example: With Uncertainty



- We can no longer find a unique sequence of actions, but
- Can we find a *policy* that tells us how to decide which action to take from each state except that now the policy maximizes the *expected* utility

Maze Example: Utility Revisited

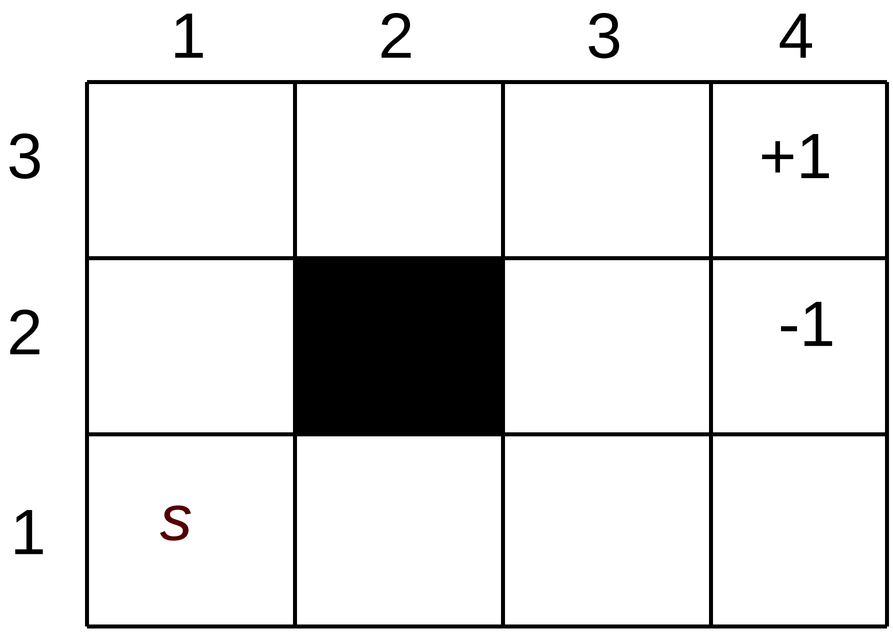




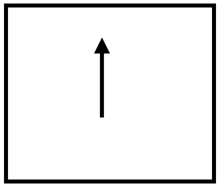
U(*s*) = Expected reward of future states starting at *s*

How to compute U after one step?

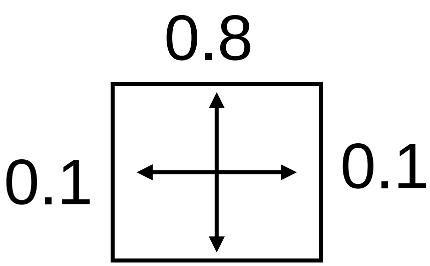
Maze Example: Utility Revisited



Intended action *a*:



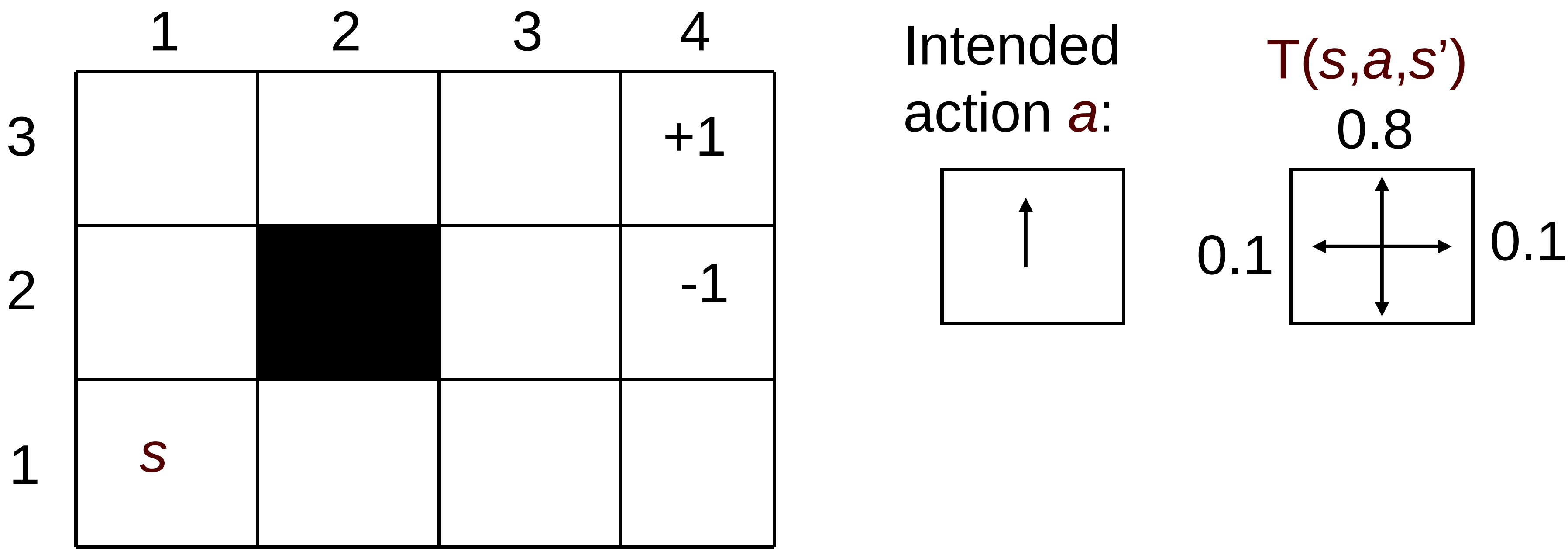
T(*s*,*a*,*s*')



Suppose *s* = (1,1) and we choose action Up.

U(1,1) = R(1,1) +

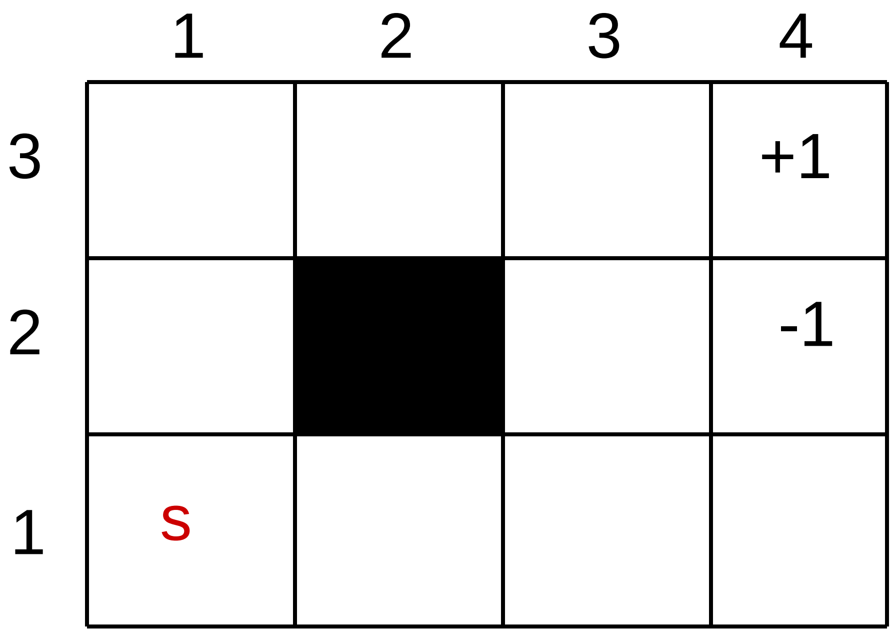
Maze Example: Utility Revisited



Suppose *s* = (1,1) and we choose action Up.

U(1,1) = R(1,1) + 0.8 x U(1,2) +

Maze Example: Utility Revisited

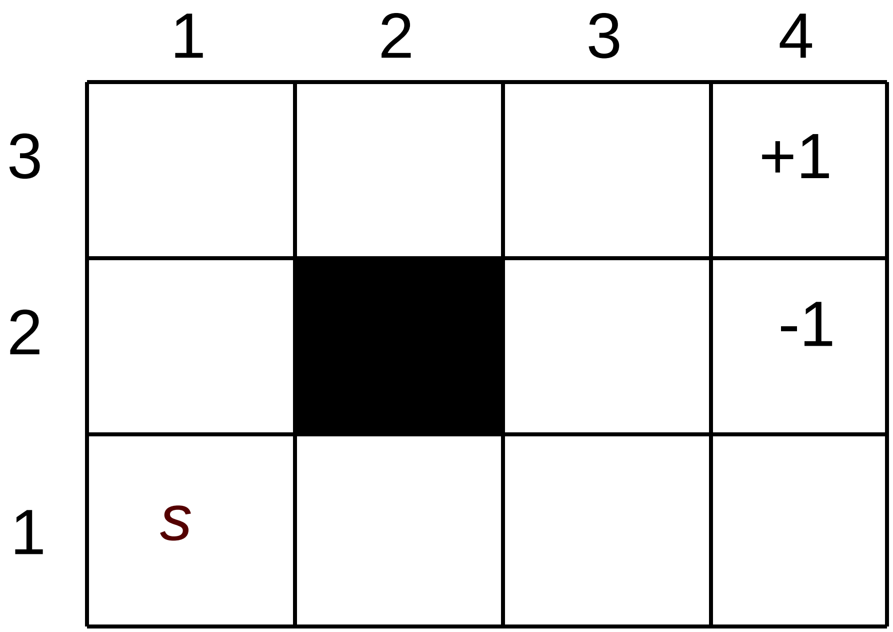


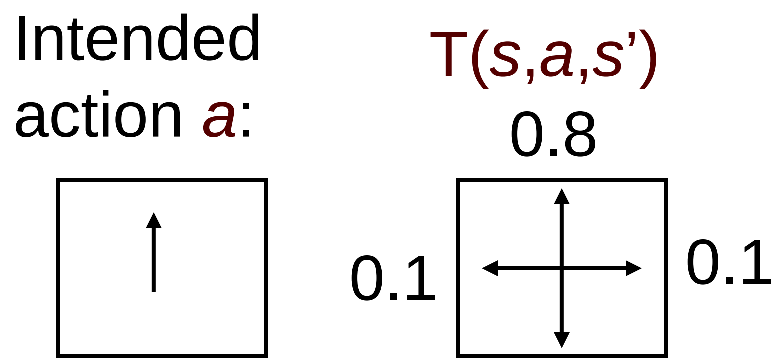
Intended action *a*:

Suppose *s* = (1,1) and we choose action Up.

U(1,1) = R(1,1) + 0.8 x U(1,2) + 0.1 x U(2,1) +

Maze Example: Utility Revisited

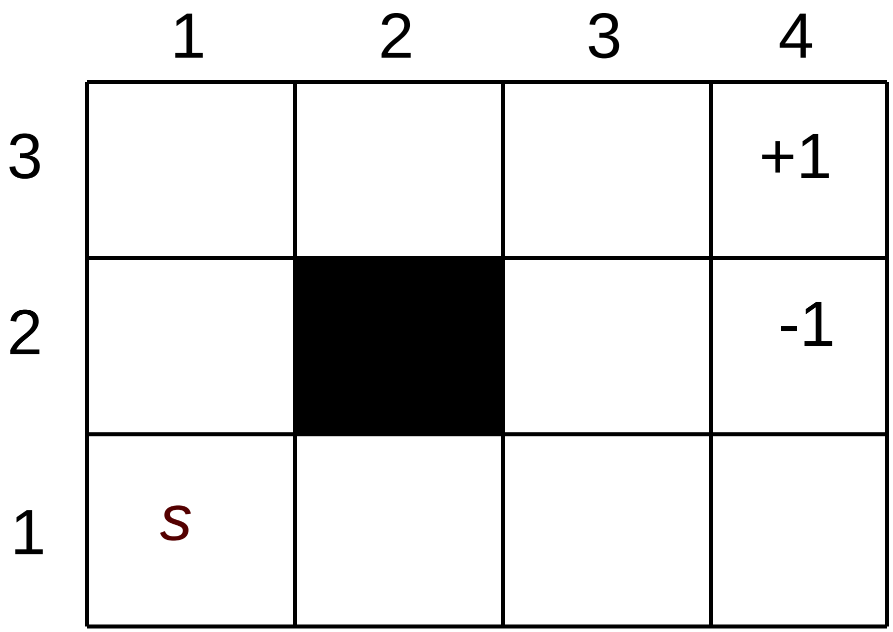


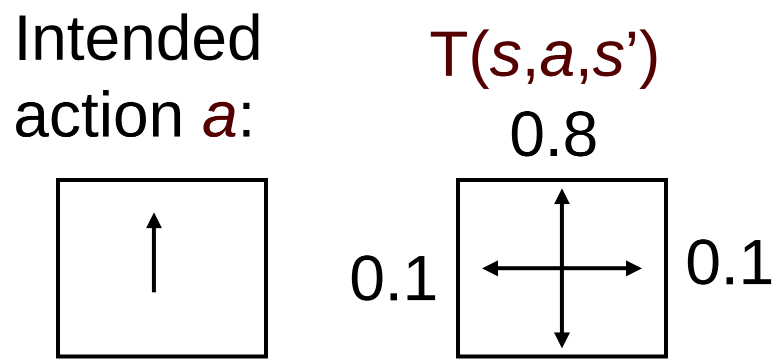


Suppose s = (1,1) and we choose action Up.

U(1,1) = R(1,1) + 0.8 x U(1,2) + 0.1 x U(2,1) + 0.1 x R(1,1)

Maze Example: Utility Revisited





Suppose *s* = (1,1) and we choose action Up.

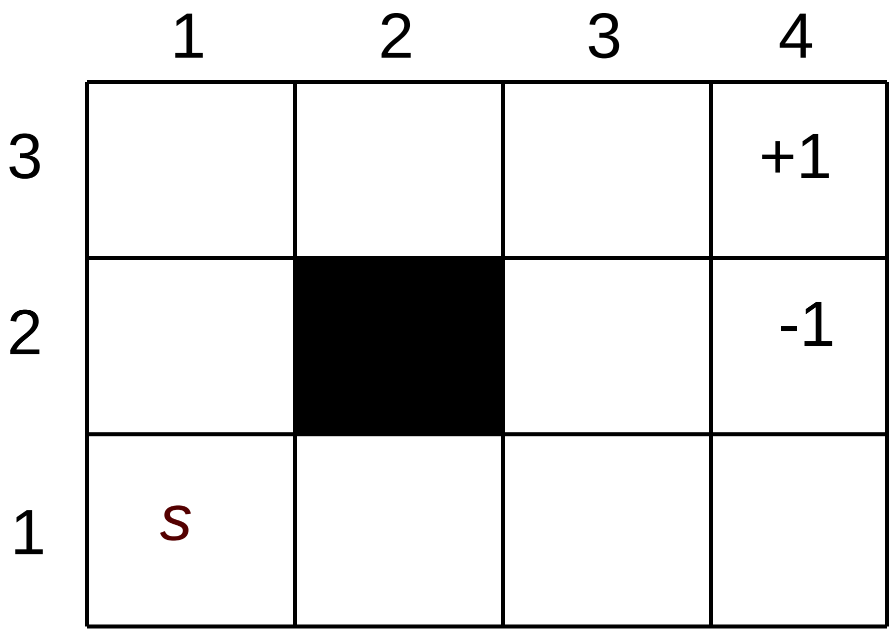
Move up with prob. 0.8

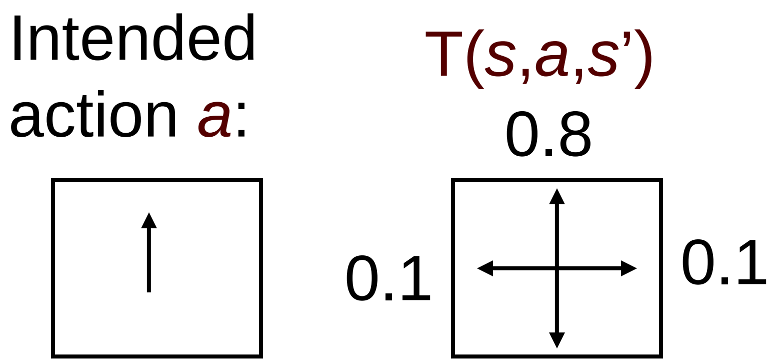
U(1,1) = R(1,1) + 0.8 x U(1,2) + 0.1 x U(2,1) + 0.1 x R(1,1)

Move right with prob. 0.1

Move left with prob. 0.1 (notice the wall!)

Same with Discount





Suppose s = (1,1) and we choose action Up.

$$U(1,1) = R(1,1) + \gamma\,(0.8 \times U(1,2) + 0.1 \times U(2,1) + 0.1 \times R(1,1))$$

More General Expression

- If we choose action *a* at state *s*, *expected* future rewards are:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s')\, U(s')$$

More General Expression

- If we choose action *a*

Expected sum of future discounted rewards starting at *s'*

Reward at current state *s*

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s')\, U(s')$$

Expected sum of future discounted rewards starting at *s*

Probability of moving from state *s* to state *s'* with action *a*

More General Expression

- If we are using policy π, we choose action *a*=π*(s)* at state *s*, *expected* future rewards are:

$$U_\pi(s) = R(s) + \gamma \sum_{s'} T(s,\pi(s),s')\ U_\pi(s')$$

Formal Definitions

- Finite set of states: S
- Finite set of allowed actions: A
- Reward function $R(s)$

- Transitions probabilities: $T(s,a,s') = P(s'|a,s)$
- Utility = sum of discounted rewards:
  - $U(s_0, \ldots) = R(s_0) + \gamma R(s_1) + \ldots + \gamma^N R(s_N) + \ldots$

- Policy: $\pi : S \rightarrow A$
- Optimal policy: $\pi^*(s)$ = action that maximizes the *expected* sum of rewards from state s

Markov Decision Process (MDP)

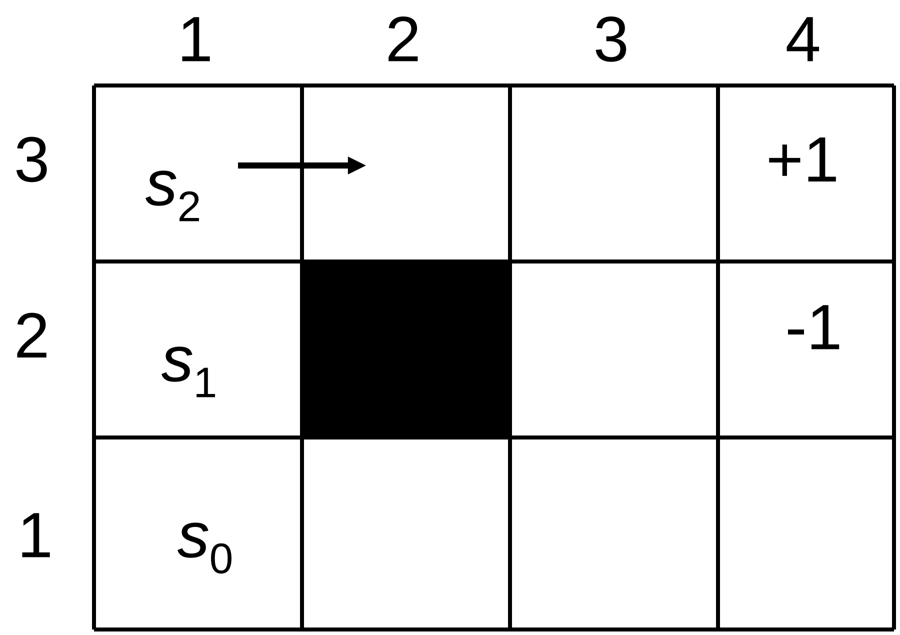
- Key property (Markov):

$$P(s_{t+1} \mid a, s_0,..,s_t) = P(s_{t+1} \mid a, s_t)$$

- In words: The new state reached after applying an action depends only on the *previous* state and it does not depend on the previous history of the states visited in the past

→ *Markov Process*

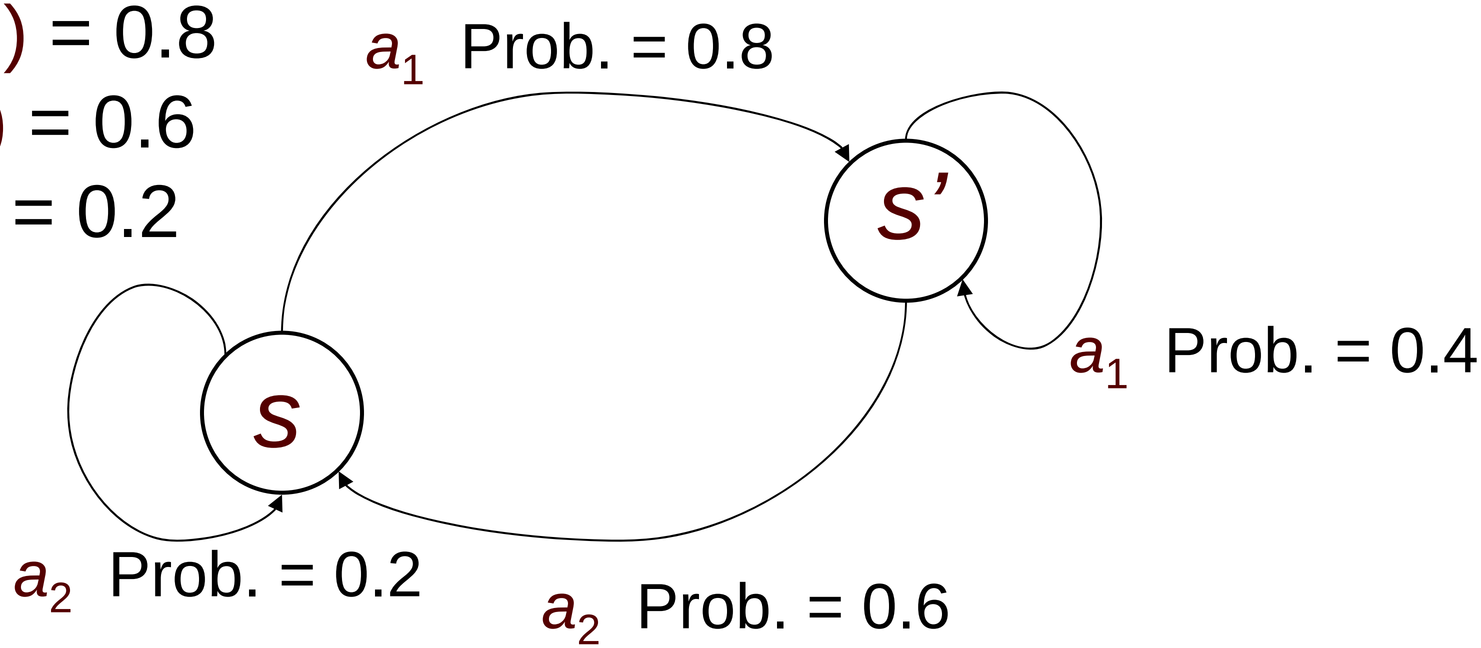
Markov Example



- When applying the action "Right" from state s_2 = (1,3), the new state depends only on the previous state s_2, not the entire history $\{s_1, s_0\}$

Graphical Notations

$T(s, a_1, s') = 0.8$
$T(s', a_2, s) = 0.6$
$T(s, a_2, s) = 0.2$

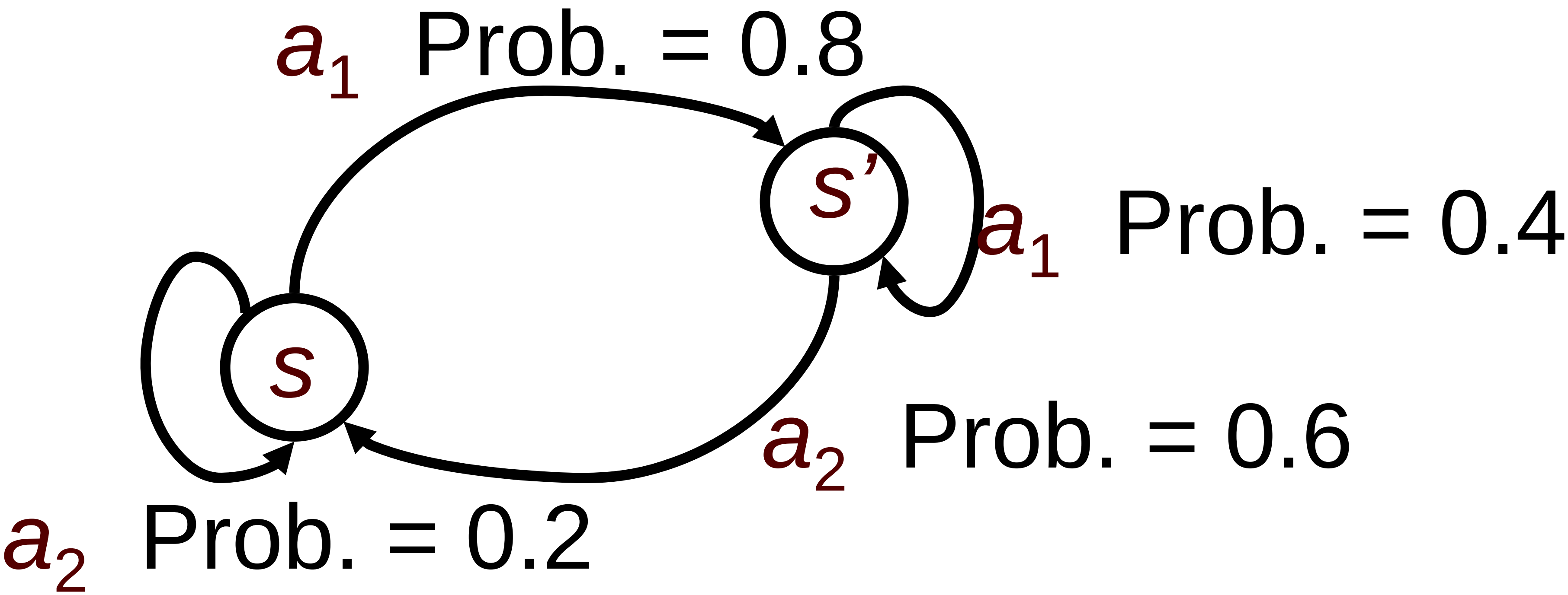


- Nodes are states
- Each arc corresponds to a possible transition between two states given an action
- Arcs are labeled by the transition probabilities

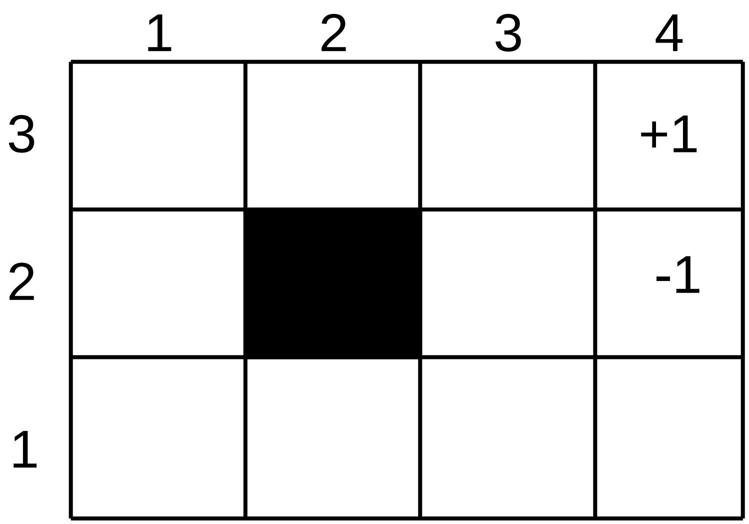
$T(s, a_1, s') = 0.8$

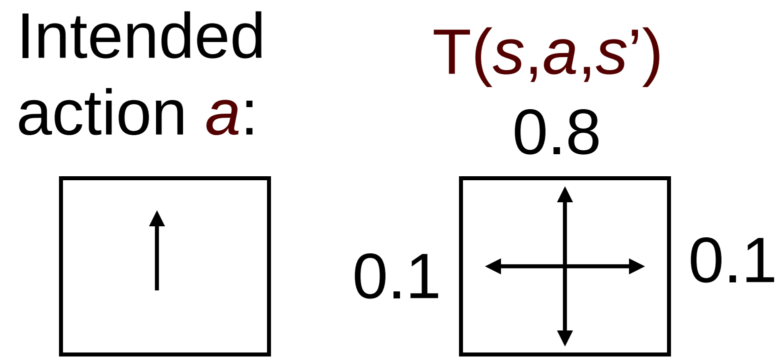
$T(s', a_2, s) = 0.6$

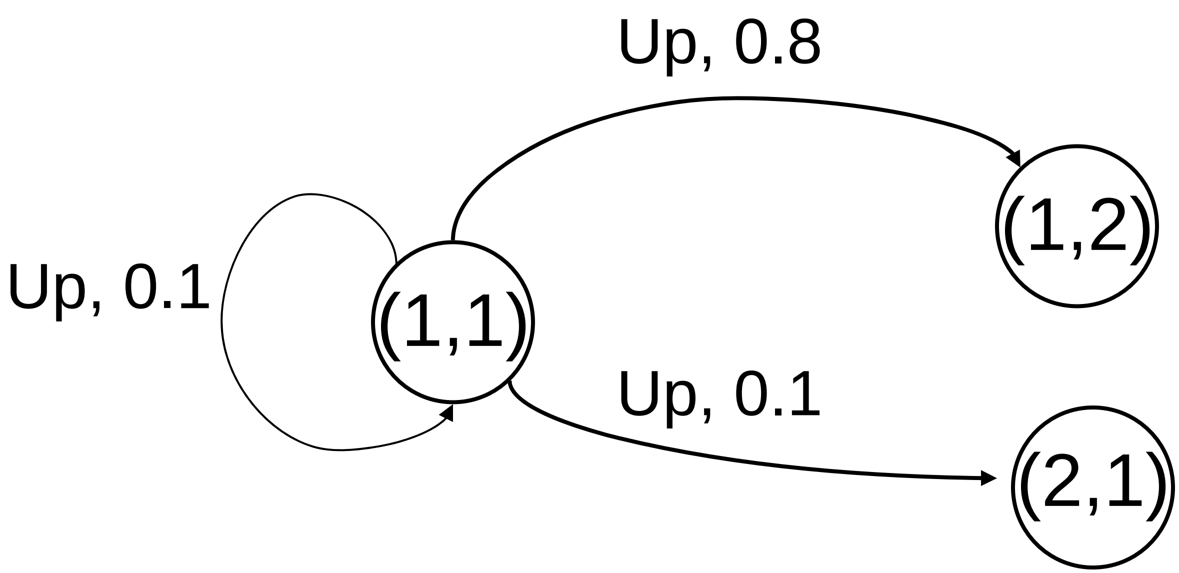
$T(s, a_2, s) = 0.2$



Example (Partial)



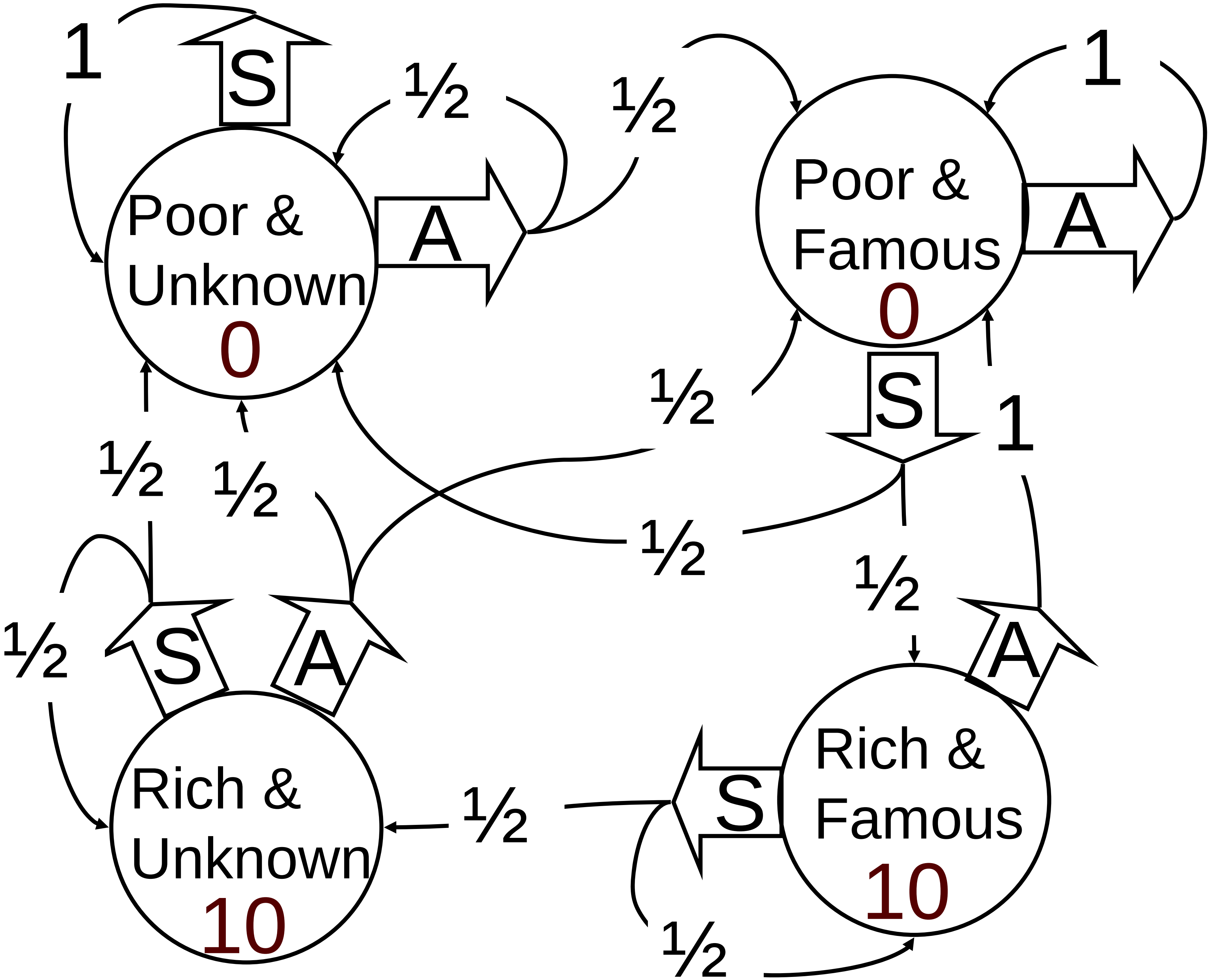




Warning: The transitions are *NOT* all shown in this example!

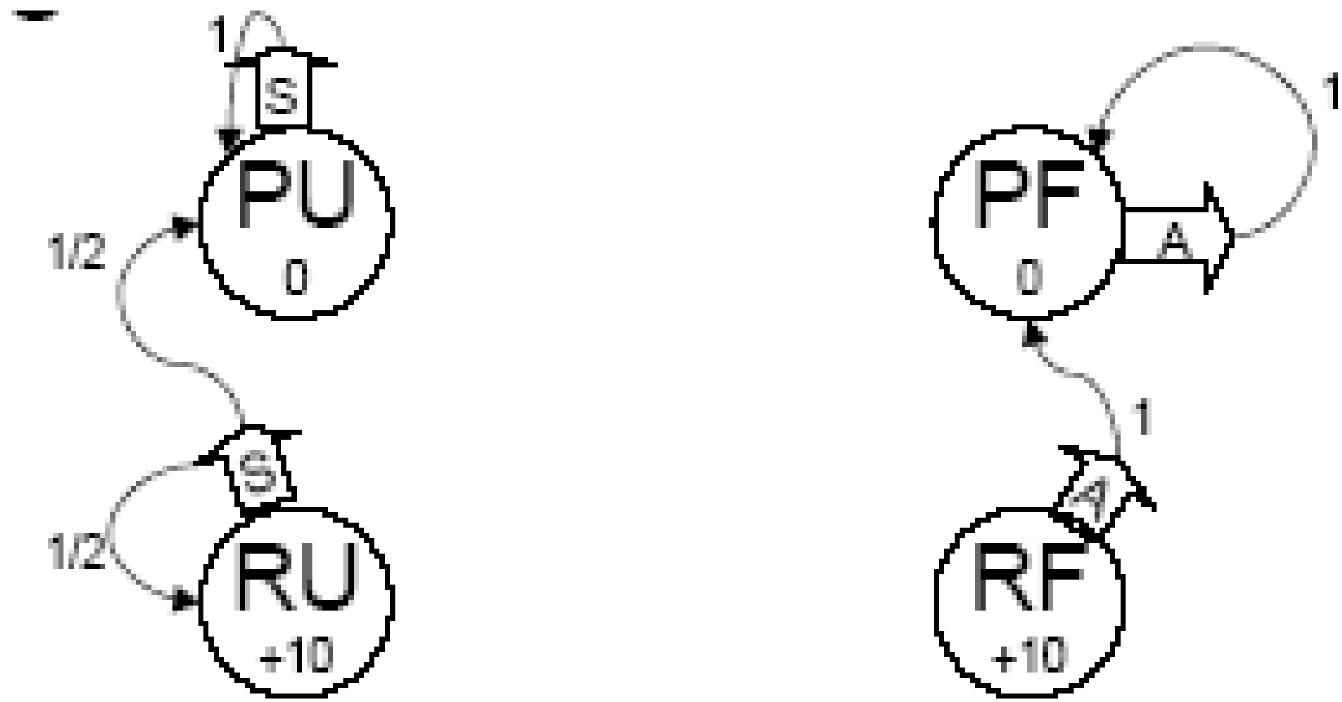
Example

- I run a company
- I can choose to either save money or spend money on advertising
- If I advertise, I may become famous (50% prob.) but will spend money so I may become poor
- If I save money, I may become rich (50% prob.), but I may also become unknown because I don’t advertise
- What should I do?


Poor & Unknown
0
Poor & Famous
0
Rich & Unknown
10
Rich & Famous
10
S
A
1
½

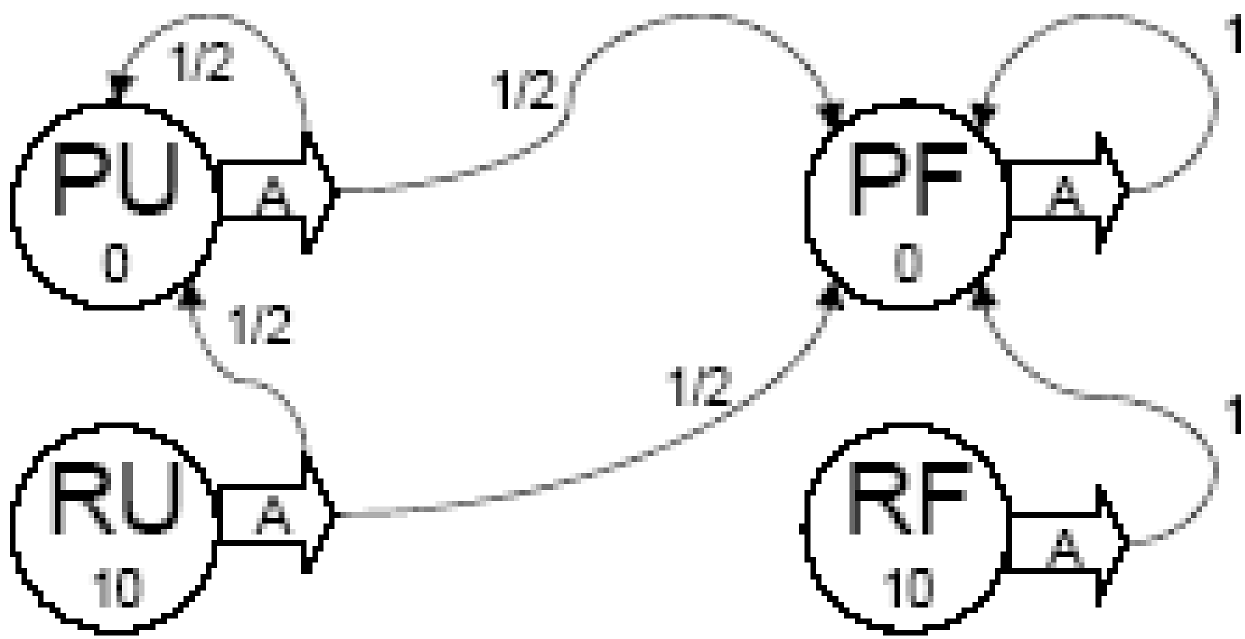
Policy Number 1:

STATE →	ACTION
PU	S
PF	A
RU	S
RF	A



Policy Number 2:

STATE →	ACTION
PU	A
PF	A
RU	A
RF	A

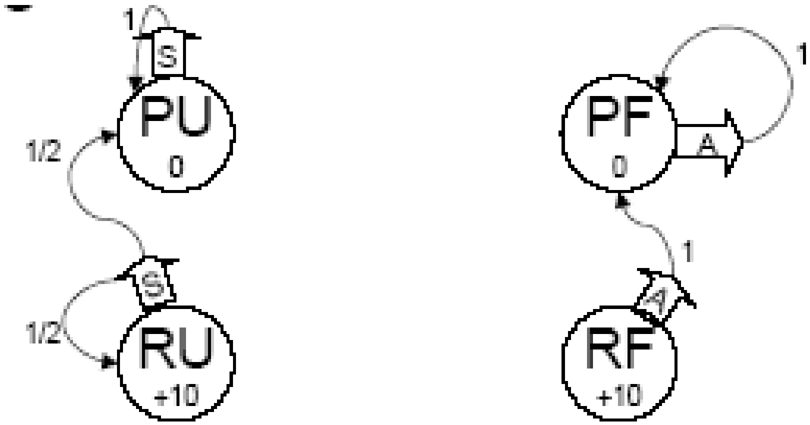


Example Policies

Examples

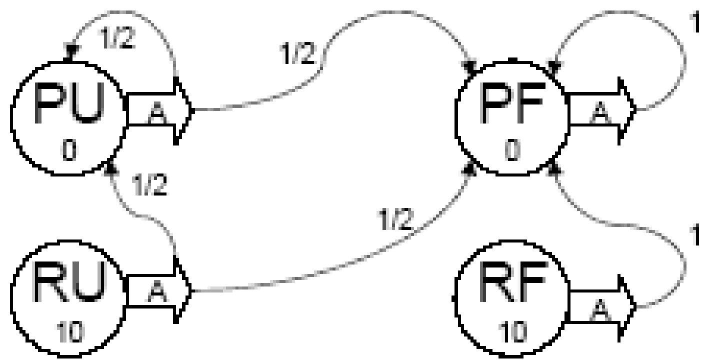
Policy Number 1:

STATE →	ACTION
PU	S
PF	A
RU	S
RF	A



Policy Number 2:

STATE →	ACTION
PU	A
PF	A
RU	A
RF	A



- Many policies
- The best policy?
- How to compute the optimal policy?

Key Result

- For every MDP, there exists an optimal policy
- There is no better option (in terms of expected sum of rewards) than to follow this policy

- How to compute the optimal policy? → We cannot evaluate all possible policies (in real problems, the number of states is very large)

Bellman's Equation

If we choose an action *a*:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s')\, U(s')$$

Bellman's Equation

If we choose an action *a*:

$$U(s) = R(s) + \gamma \sum_{s'} T(s,a,s')\, U(s')$$

In particular, if we always choose the action *a* that maximizes future rewards (optimal policy), U(*s*) is the maximum U*(*s*) we can get over all possible choices of actions:

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

Bellman's Equation

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

- The optimal policy (choice of *a* that maximizes U) is:

$$\pi^*(s) = \text{argmax}_a \left(\sum_{s'} T(s,a,s')\, U^*(s')\right)$$

Why it cannot be solved directly

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

Set of |S| equations. Non-linear because of the "max": Cannot be solved directly!

- The optimal p
  maximizes U

$$\pi^*(s) = \text{argmax}_a \left(\sum_{s'} T(s,a,s')\, U^*(s')\right)$$

Expected sum of rewards using policy π^* → The right-hand depends on the unknown. Cannot solve directly

First Solution: Value Iteration

- Define $U_1(s)$ = best value after *one* step

$$U_1(s) = R(s)$$

- Define $U_2(s)$ = best possible value after *two* steps

$$U_2(s) = R(s) + \gamma \max_a \left(\sum_{s'} T(s,a,s')\, U_1(s')\right)$$

…………………………………….

- Define $U_k(s)$ = best possible value after *k* steps

$$U_k(s) = R(s) + \gamma \max_a \left(\sum_{s'} T(s,a,s')\, U_{k-1}(s')\right)$$

First Solution: Value Iteration

- Define $U_1(s)$ = best value after *one* step

$$U_1(s) = R(s)$$

- Define $U_2(s)$ = best value after *two* steps

Maximum possible expected sum of discounted rewards that I can get if I start at state *s* and I survive for *k* time steps.

$$s') \, U_1(s'))$$

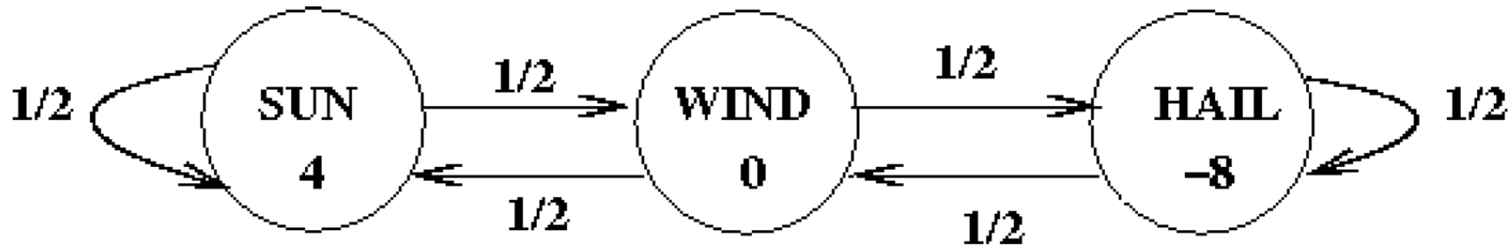
……………………………

- Define $U_k(s)$ = best value after *k* steps

$$U_k(s) = R(s) + \gamma \max_a \left(\sum_{s'} T(s,a,s') \, U_{k-1}(s')\right)$$

3-State Example – Value Iteration

Computation for Markov chain (no policy)



3-State Example: Values $\gamma = 0.5$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	5.0	-1.0	-10.0
3	5.0	-1.25	-10.75
4	4.9375	-1.4375	-11.0
5	4.875	-1.515625	-11.109375
6	4.8398437	-1.5585937	-11.15625
7	4.8203125	-1.5791016	-11.178711
8	4.8103027	-1.5895996	-11.189453
9	4.805176	-1.5947876	-11.194763
10	4.802597	-1.5973969	-11.197388
11	4.8013	-1.5986977	-11.198696
12	4.8006506	-1.599349	-11.199348
13	4.8003254	-1.5996745	-11.199675
14	4.800163	-1.5998373	-11.199837
15	4.8000813	-1.5999185	-11.199919

3-State Example: Values $\gamma = 0.9$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	5.8	-1.8	-11.6
3	5.8	-2.6100001	-14.030001
4	5.4355	-3.7035	-15.488001
5	4.7794	-4.5236254	-16.636175
6	4.1150985	-5.335549	-17.521912
7	3.4507973	-6.0330653	-18.285858
8	2.8379793	-6.6757774	-18.943516
9	2.272991	-7.247492	-19.528683
...	...	...	...
50	-2.8152928	-12.345073	-24.633476
51	-2.8221645	-12.351946	-24.640347
52	-2.8283496	-12.3581295	-24.646532
...	...	...	...
86	-2.882461	-12.412242	-24.700644
87	-2.882616	-12.412397	-24.700798
88	-2.8827558	-12.412536	-24.70094

3-State Example: Values $\gamma = 0.2$

Iteration	SUN	WIND	HAIL
0	0	0	0
1	4	0	-8
2	4.4	-0.4	-8.8
3	4.4	-0.44000003	-8.92
4	4.396	-0.452	-8.936
5	4.3944	-0.454	-8.9388
6	4.39404	-0.45443997	-8.93928
7	4.39396	-0.45452395	-8.939372
8	4.393944	-0.4545412	-8.939389
9	4.3939404	-0.45454454	-8.939393
10	4.3939395	-0.45454526	-8.939394
11	4.3939395	-0.45454547	-8.939394
12	4.3939395	-0.45454547	-8.939394

Next

- More value iteration
- Policy iteration