CSCI 561
Foundation for Artificial Intelligence

07 - Reinforcement Learning
08 – Learn to Search and Play Games

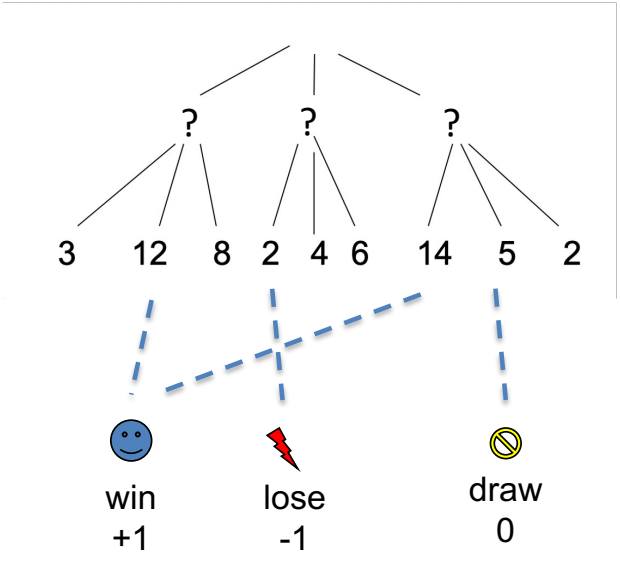
Professor Wei-Min Shen
University of Southern California

Outline

- Motivation
  - Agent and Environment (Search & Games)
- States, actions, utility, rewards, policy
- Utility value iteration
- Policy Iterations
- Reinforcement Learning
  - Model-based
  - Model-free
- Q-Learning
  - State space for advanced game playing

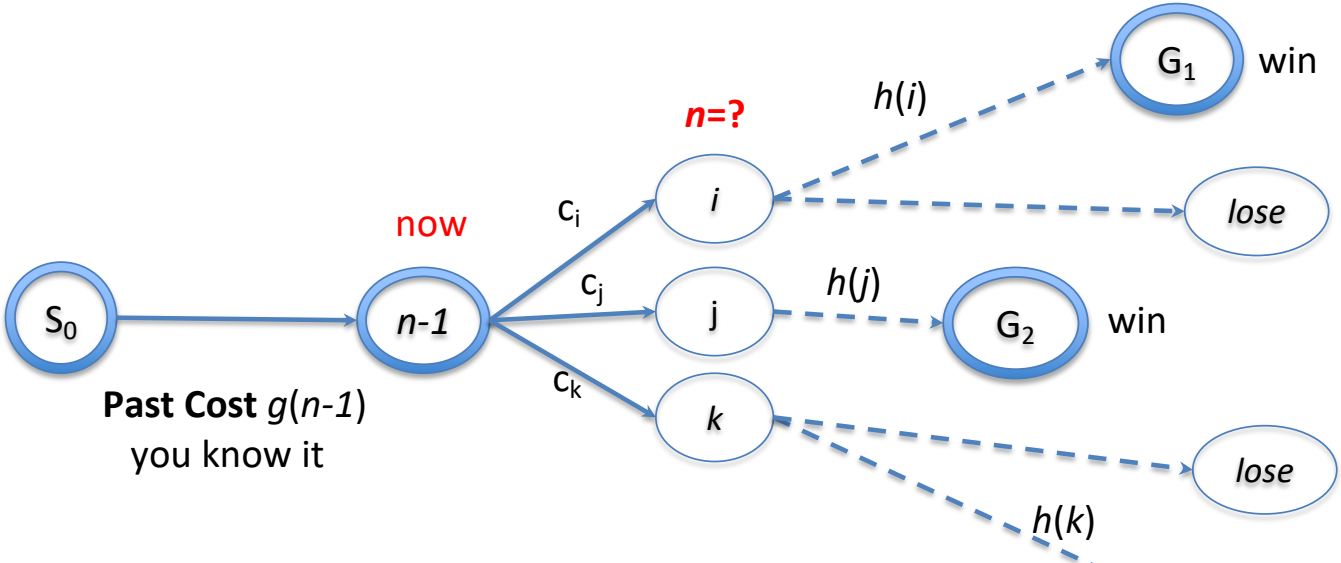
A Key Question

- In all search and game playing problems, what is the key information that you wish to have for finding the optimal solution?
  - What you wish to have but may not always have?



The True Future Values

(Only if we would know them !)



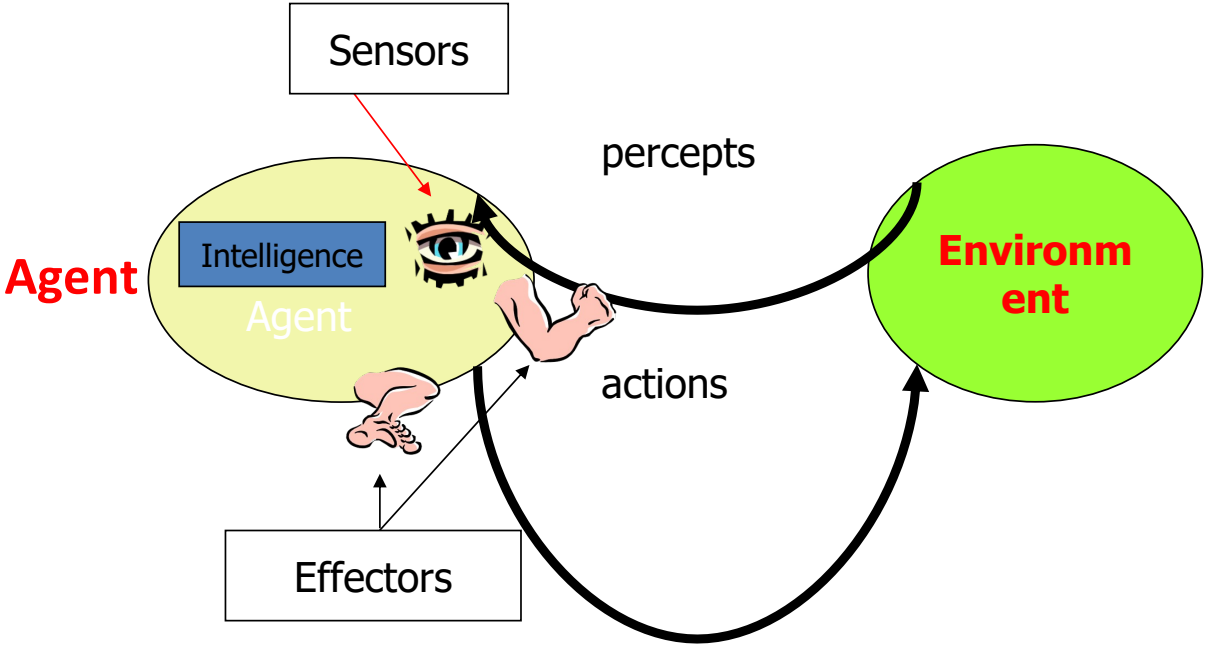
Past Cost *g*(*n-1*)
you know it

f(n) = *g*(*n*)+*h**(*n*)
f(n) = *g*(*n*)+*h*(*n*)

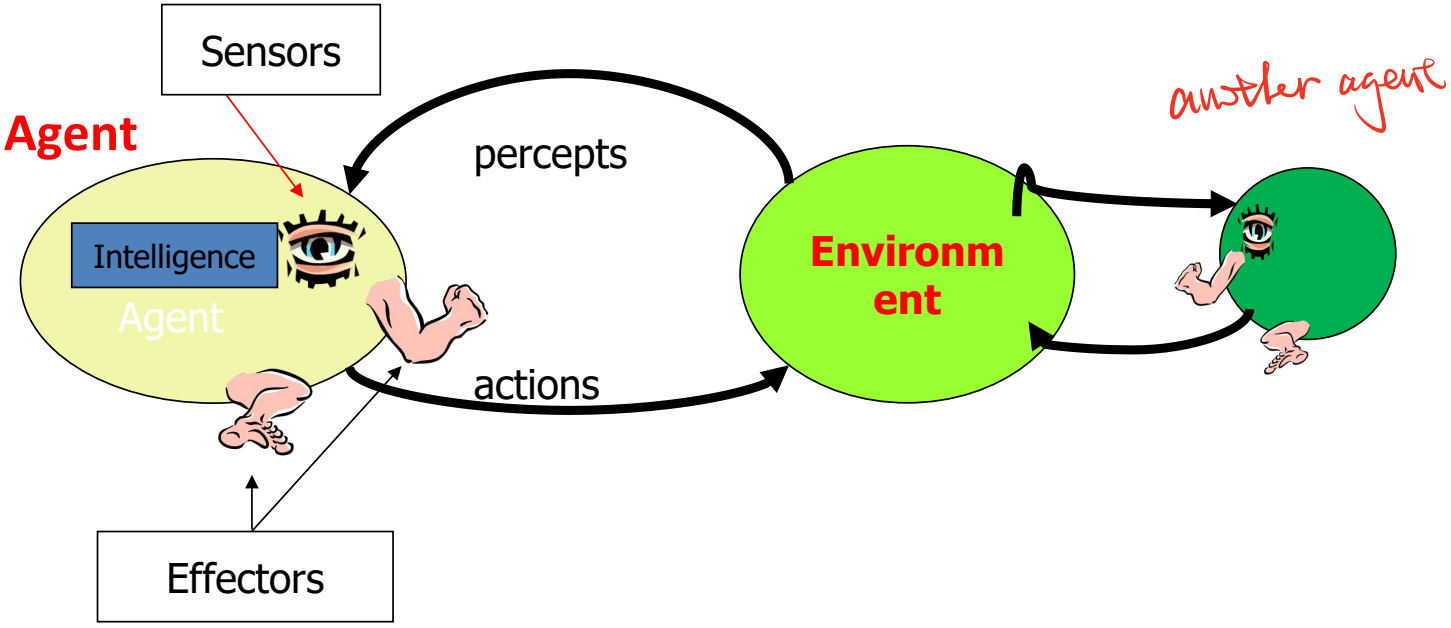
Future Cost Values

The true h*(n) is unknown
We can only estimate it by *h*(*n*)
Can we do better?

Agent and Environment



Agent, Environment, Game



Agent and Environment (Star War)

1 	2	3	4 
5		6	7 
8	9	10	11

Agents:
- R2D2
- 3PIO
- Darth Vader

States: 1,..,11

Actions: ^,v,>,<

Percepts: …

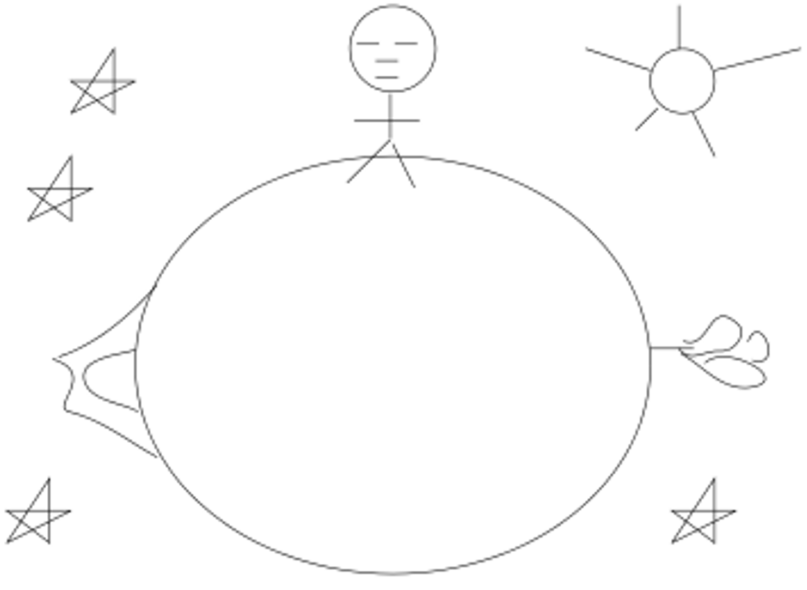
Goals:…

Rewards: …

State Utility Values: …
(your “crystal ball”)

Agent and Environment

The Little Prince's Planet



- Percepts: {rose, volcano, nothing}
- Actions: {forward, backward, turn-around}
- States: {north, south, east, west}

The Little Prince Planet Environment

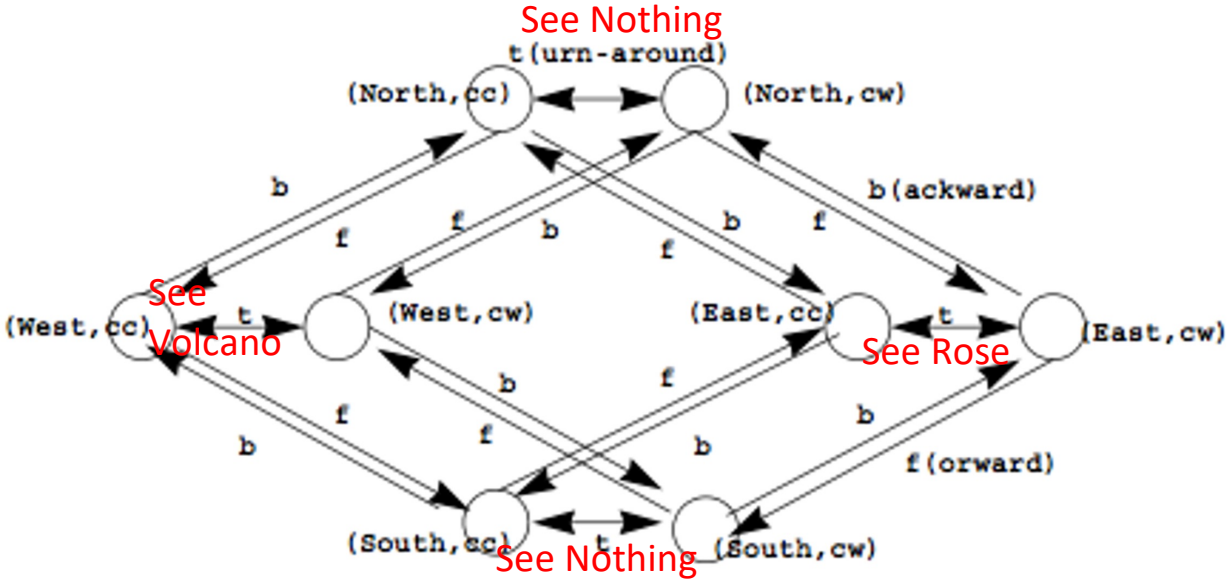


Figure 2.5: The little prince's actual environment.

4th month turning around = jump to South pole

A Model for Little Prince's Planet

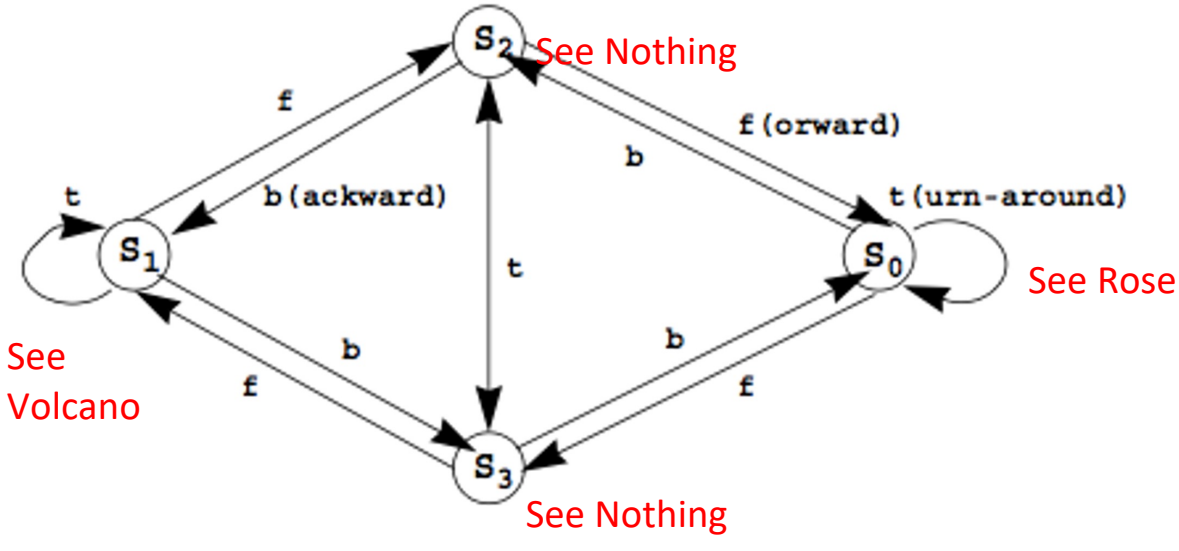


Figure 2.3: The little prince's model of his world.

His internal action model only needs 4 states! (why?)

State, Action and Sensor Models

new definition method

We represent a model M of the environment $\mathcal{E}$ as a machine $(A\ Z\ S, \phi, \theta, t)$, where

- A is a set of actions, which is the same as the set of input actions into $\mathcal{E}$,
- Z is a set of percepts, which is the same as the set of output symbols of $\mathcal{E}$,
- S is a set of model states,
- ϕ, a function from $S \times A$ to S, is the transition function of M,
- θ, a function from S to Z is the appearance function of M, and
- t is the current model state of M. When a basic action a is applied to M, t is updated to be $\phi(t, a)$.

also T.

state + action ⇒ state

in which state what can you see

Little Prince's Model

$$
\begin{aligned}
A &\equiv \{\text{forward, backward, turn-around}\} \\
P &\equiv \{\text{rose, volcano , nothing}\} \\
S &\equiv \{s_1, s_2, s_3, s_4\} \\
\phi &\equiv \phi(s_0, \text{forward}) = s_3, \phi(s_0, \text{backward}) = s_2, \ldots \\
\theta &\equiv \theta(s_1) = \{\text{volcano}\}, \theta(s_0) = \{\text{rose}\}, \theta(s_2) = \theta(s_3) = \{\}
\end{aligned}
$$

Transition Model ϕ can be deterministic or probabilistic

Sensor Model θ can be deterministic or probabilistic

Are the transition model φ here deterministic?

Are the sensor model ϑ deterministic?

Is there any hidden states in this example?

当see nothing时，不能确定是在S_2还是S_3

二、Uncertain Environment

The Environment may be Uncertain

1. 环境.

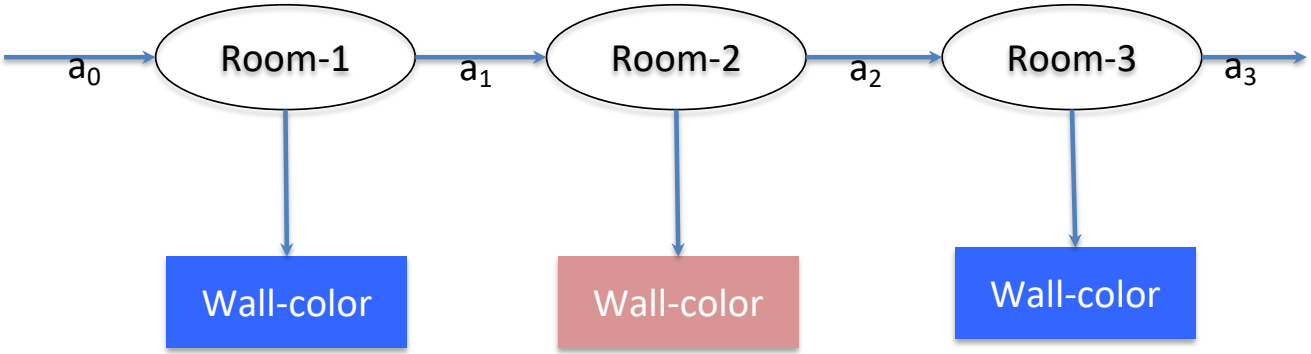
why

- "Seeing may not always be believing"
  - States and observations are not always 1-1 mapping
    - One state may be seen in many different ways
    - The same observation may be seen in many different states
    - E.g., When the Prince sees nothing, where is he? Facing which way?
    - Can you make the little prince's sensors non-deterministically?
- "Action may not always produce the same result"
  - Actions do not always take you to the same states
    - E.g., the Little Prince's planet may have "planet-quakes" ☺️
    - E.g., step forward on ice may not always succeed
  - Transitions between states by actions may be nondeterministic
    - E.g., Star War's transitions are not deterministic
    - Can you make the Little Prince's actions non-deterministically?

1). seeing

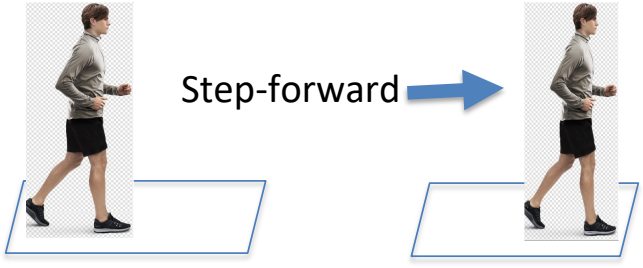
Sensors can be uncertain (in real world)

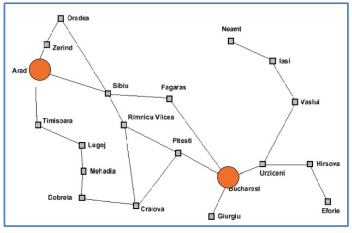
- Speech Recognition
  - "Listening is not always equal to hearing" ☺️
- Traveling through rooms with colored walls
  - "Seeing does not always tell where you are" ☺️



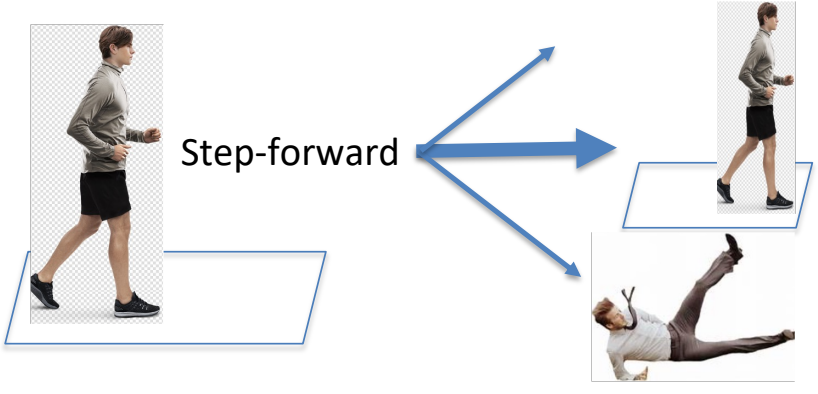
Actions can be uncertain (in real world)

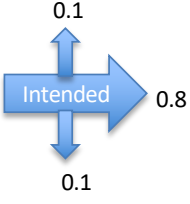
- Deterministic actions





- Non-deterministic actions (e.g., walking on ice)



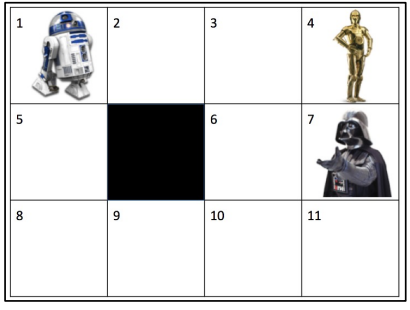


Other Uncertainties in the Real World

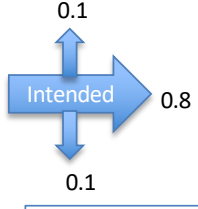
- Multiple Agents
  - The actions of other agents in the environment may be ”uncertain” from your point of view !
- Advanced Game Playing
  - Your opponent’s moves are definitely “uncertain” from your point of view !
- When you driving on a highway
  - What are uncertain?

Star War's Probabilistic Transition Model

- Transitions can be probabilistic
  - $P(state_{t+1} \mid state_t, action_t)$

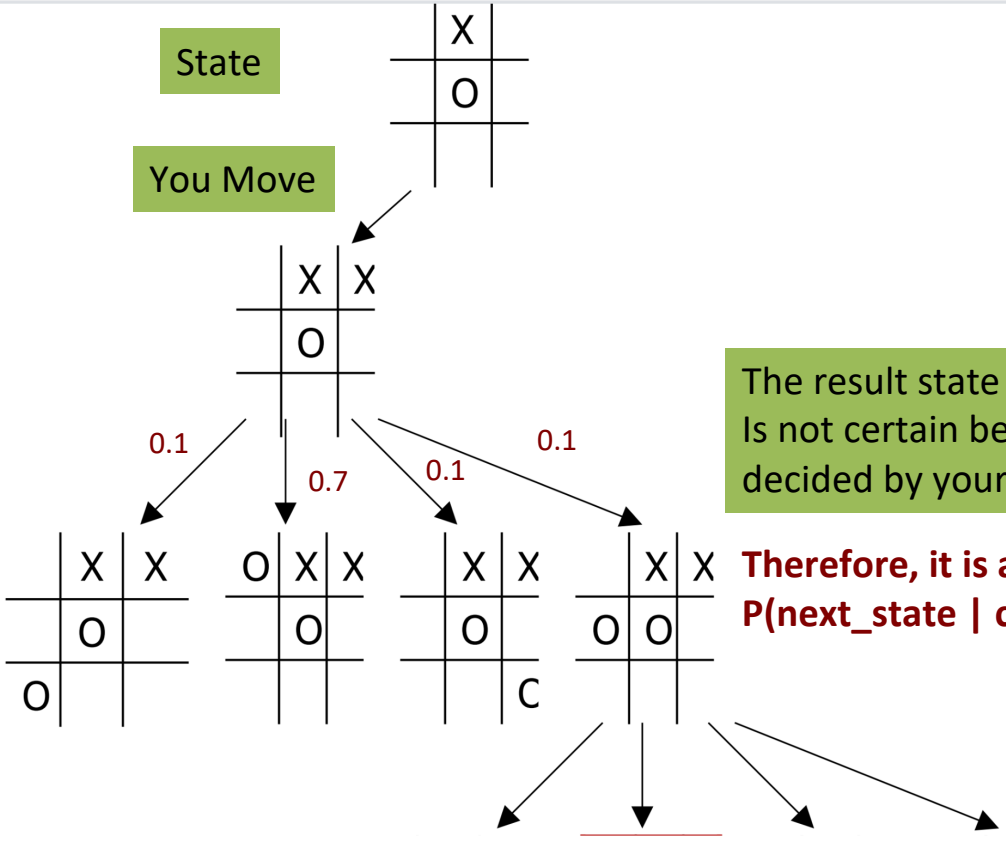


X_{t-1}	A_{t-1}	$P(X_t=1)$	$P(X_t=2)$	$P(X_t=3)$	$P(X_t=4)$	$P(X_t=5)$	...
1	up	0.8+0.1	0.1	0.0	0.0	0.0	
1	down	0.1	0.1	0.0	0.0	0.8	
1	left	0.8+0.1	0.0	0.0	0.0	0.1	
1	right	0.1	0.8	0.0	0.0	0.1	
2	up	0.1	0.8	0.1	0.0	0.0	
...							



Bounce back if hits a wall

In Game Playing: Opponent's Actions are not certain



The result state of your move Is not certain because it will be decided by your opponent

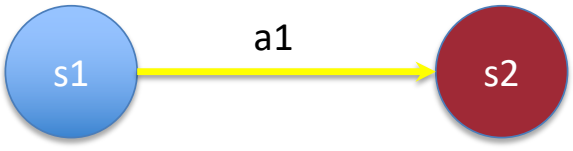
Therefore, it is appropriate to use:
P(next_state | current_state, your_move)

2. The Key Representations

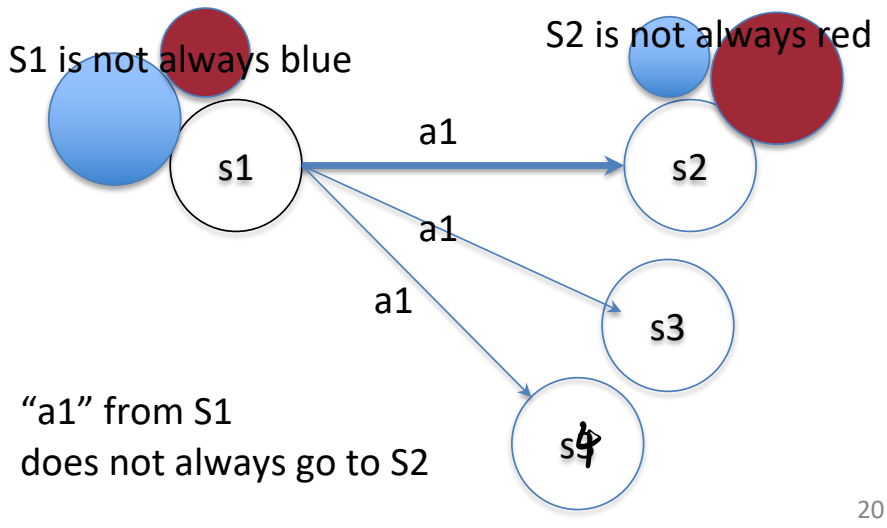
- Model the environment by States, Actions, Percepts, and
- Transition (action) model $\varphi = P(s_{t+1} | s_t, a_t)$ may be probabilistic
- Sensor Model $\vartheta = P(z|s)$ may be probabilistic
- States may have Utility Value U(s) or V(s)
- Agents may receive Reward ***r*** (implicit "goals") occasionally:
  - rewards may be given to agent in different format or time
- **Behaviors** may be represented as Policy: state -> action
- **Objective**: find the optimal policy based on utilities and rewards

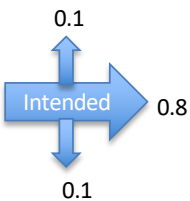
3. Certainty vs. Uncertainty

The certain way



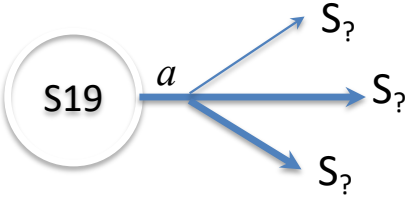
The uncertain way





Uncertainty in Sensor & Actions

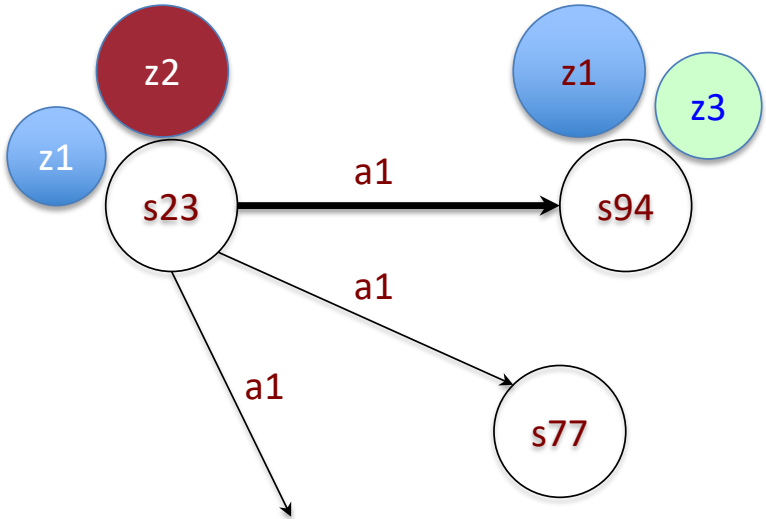
State, action



Observations

- Solution: let's use probabilities
  - Action/Transition Model (for states and actions)
    Use Probabilistic Transitions
    P(NextState | CurrentState, Action)
  - Sensor Model (for states and observations)
    P(Observations | State)

Hidden Markov Model (with actions)



1. Actions (可以没有)
2. Percepts (observations)
3. States
4. Appearance: states -> observations
5. Transitions: (states, actions) -> states
6. Current State

1. definition

Hidden Markov Model (1/2)

hidden Markov model M is a stochastic process $(B, Z, S, P, \theta, \pi)$ with the components defined as follows:

- B is a set of basic actions. As before, the actions can be applied to both the environment and the model.
- Z is a set of percepts and represents the output symbols of the environment that can be observed by the learner.
- S is a finite set of (internal) model states. We assume that at any single instant t, the current environmental state q_t corresponds to exactly one model state s_t, and the identity of s_t is sufficient to stochastically determine the effects of action in the environment. This is the *Markov assumption*. → non-episodic
- $\phi = \{P_{ij}[b]\}$ is a set of probabilities concerning model state transitions. For each basic action $b \in B$ and a pair of model states s_i and $s_j \in S$, the quantity $P_{ij}[b]$ specifies the probability that executing action b when the current model state is s_i will move the environment to an environmental state that corresponds to the model state s_j.

Action Model

Hidden Markov Model (2/2)

Sensor Model

- $\theta = \{\theta_i(k)\}$, where $\theta_i(k) = p(z_k|s_i)$, $z_k \in Z$, and $s_i \in S$, is a set of probability distributions of observation symbols in each model state. (This corresponds to the appearance function for the deterministic models defined at the beginning of this chapter.) For each observation symbol z_k, the quantity $\theta_i(k)$ specifies the probability of observing z_k if the current model state is s_i.

current state

- $\pi(t) = \{\pi_i(t)\}$ is the probability distribution of the current model state at time t. That is, $\pi_i(t) = p(i_t = s_i)$, where i_t denotes the current model state and $s_i \in S$ specifies the probability of s_i being the current model state at time t.

This is also called "localization"

2. eg:

The HMM for Little Prince's Planet

(with uncertain actions and sensors)

$A \equiv$ {forward, backward, turn-around} {*f*, *b*, *t*}

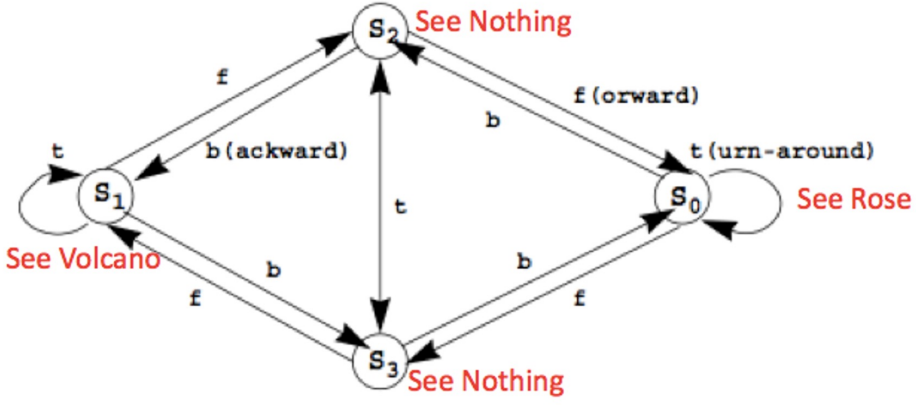
$Z \equiv$ {rose, volcano , nothing}

$S \equiv \{s_1, s_2, s_3, s_4\}$

$\phi \equiv$ {$P(s_3|s_0,f)=.51$, $P(s_2|s_1,b)=.32$, $P(s_4|s_3,t)=.89$, …}

$\theta \equiv$ {$P(rose|s_0)=.76$, $P(volcano|s_1)=.83$, $P(nothing|s_3)=.42$, …}

$$\pi_1(0)=0.25, \pi_2(0)=0.25, \pi_3(0)=0.25, \pi_4(0)=0.25$$



Little Prince Example (may add Rewards)

- (action) Transition Probabilities ϕ (Fwd, Back, Turn)

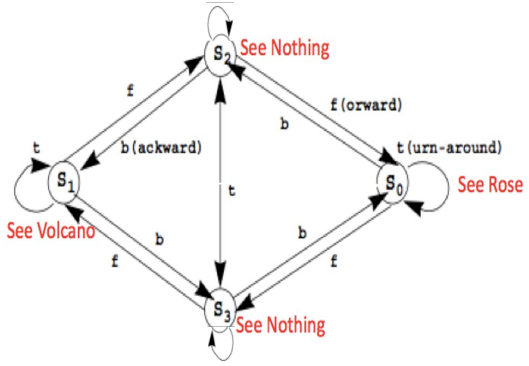
F	S0	S1	S2	S3
S0	0.1	0.1	0.1	0.7
S1	0.1	0.1	0.7	0.1
S2	0.7	0.1	0.1	0.1
S3	0.1	0.7	0.1	0.1

B	S0	S1	S2	S3
S0	0.1	0.1	0.7	0.1
S1	0.1	0.1	0.1	0.7
S2	0.1	0.7	0.1	0.1
S3	0.7	0.1	0.1	0.1

T	S0	S1	S2	S3
S0	0.7	0.1	0.1	0.1
S1	0.1	0.7	0.1	0.1
S2	0.1	0.1	0.1	0.7
S3	0.1	0.1	0.7	0.1

- (sensor) Appearance Probabilities θ

θ	Rose	Volcano	Nothing
S0	0.8	0.1	0.1
S1	0.1	0.8	0.1
S2	0.1	0.1	0.8
S3	0.1	0.1	0.8



- Initial State Probabilities π

π	S0	S1	S2	S3
	.25	.25	.25	.25

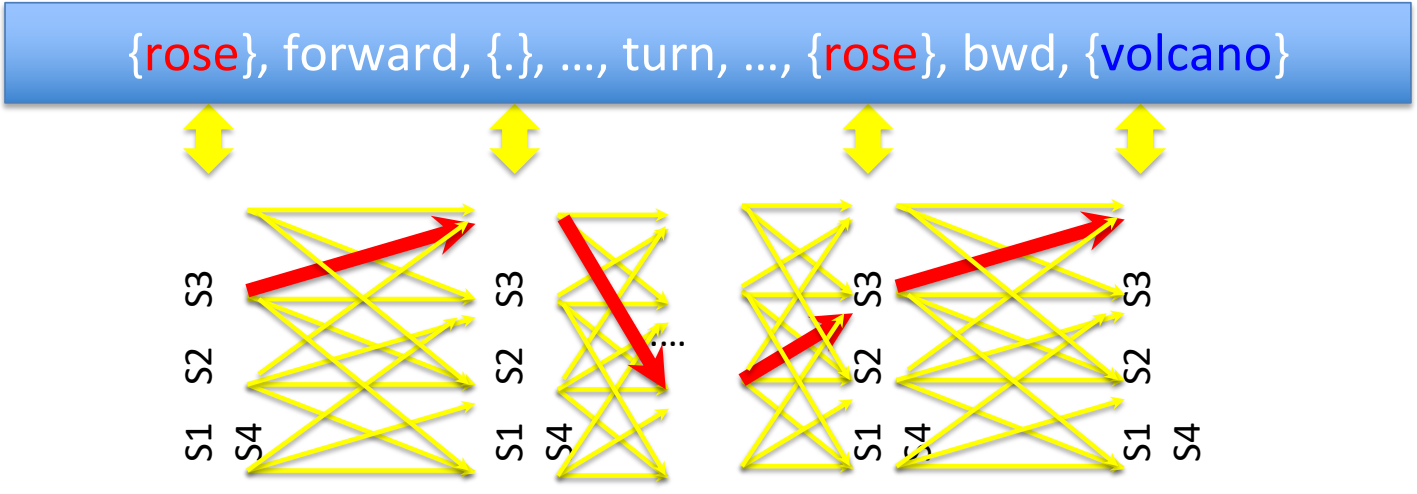
You may add rewards to states (e.g., prefer to see rose)

Experience and State Sequence

- $\mathbf{E_{1:T}}$ is an experience which is a sequence of
  - $\{observe_1, act_1, observe_2, act_2, \ldots., act_{T-1}, observe_T\}$
  - $\mathbf{E_{1:T}} = \{o_1, a_1, o_2, a_2, o_3, \ldots, o_{T-1}, a_{T-1}, o_T\}$
- $\mathbf{X_{1:T}}$ is a sequence of states that may be hidden but correspond to the experience
  - $X_{1:T} = \{X_1, X_2, X_3, \ldots, X_{T-1}, X_T\}$

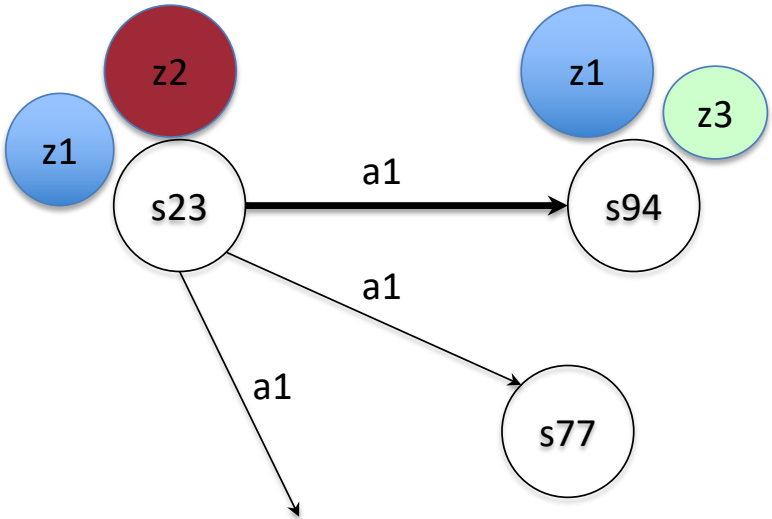
Little Prince Example

- From an "*experience*" (time$_1$ through time$_t$)
- Infer the most likely "*sequence of states*"



Find the "best sequence of (hidden) states" that support the experience!

Action and Sensor Models (review)

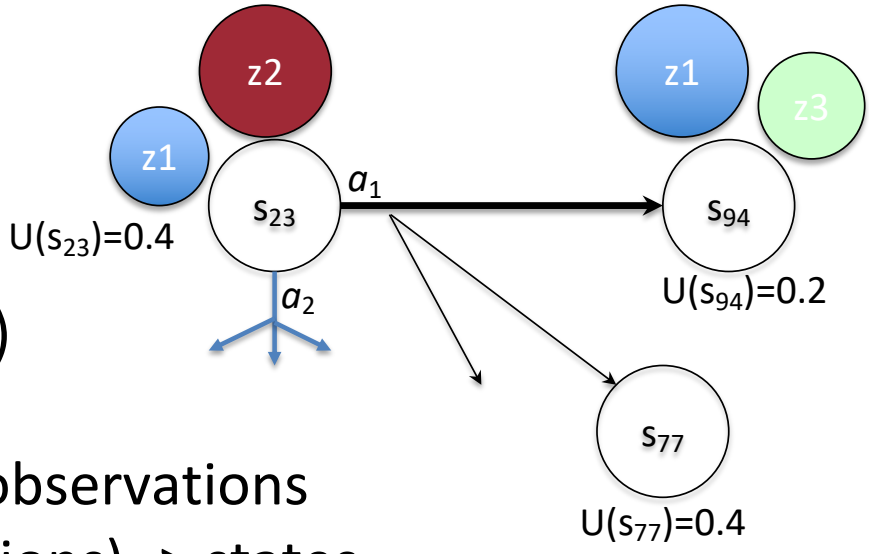


1. Actions
2. Percepts (observations)
3. States
4. Appearance: states -> observations
5. Transitions: (states, actions) -> states
6. Current State

What about the goals?

Utility Value of States

Utility Value ⇔ Goal Information

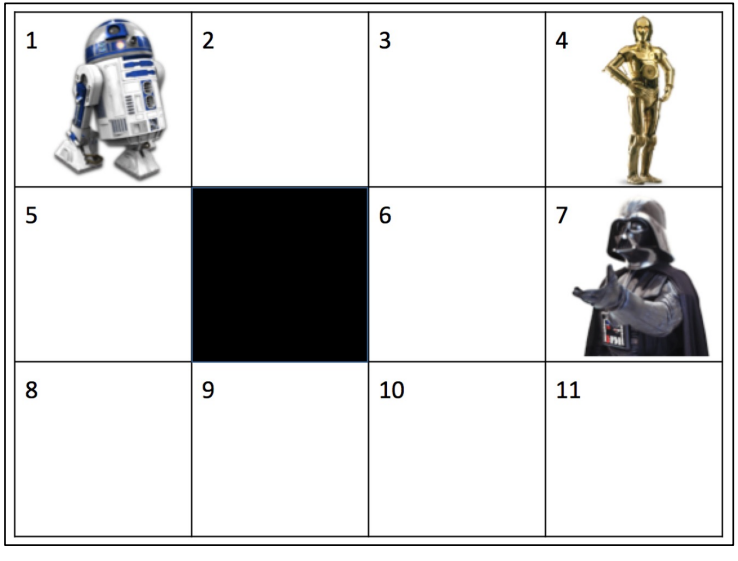


1. Actions
2. Percepts (observations)
3. States
4. Appearance: states -> observations
5. Transitions: (states, actions) -> states
6. Current State
7. **Rewards**: R(s) or R(s,a) (related to the goals, given to the agent)

(8) **Utility Value** of States: $U(s)$ (how good for the goals, must compute)

Rewards and Utilities

- Three types of rewards for agents
  1. Being at a state R(s), e.g., holding an ice cream
  2. Do an action on a state R(s, a), e.g., eating the ice cream
  3. Making a transition R(s, a, s'), e.g., feeling good after eating
- Every state may have a Utility Value U(s) or V(s)



Rewards and Utility for R2D2:

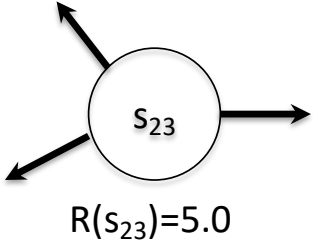
X_t	Reward	Utility
4	+1	0.0
7	-1	0.0
Else	-0.04	0.0

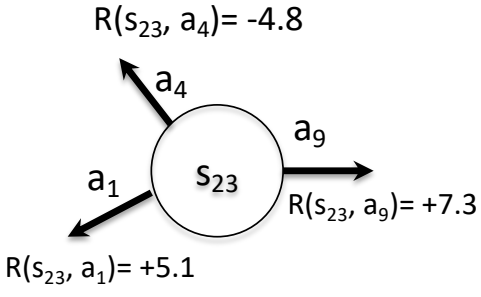
Given

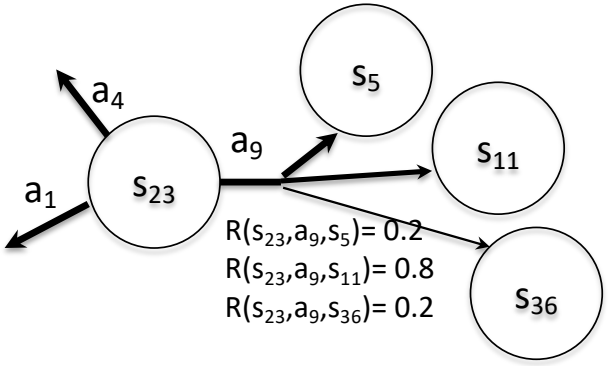
To be learned

Rewards (caution, 3 types)

- Three types of rewards for agents
  1. Being at a state R(s), e.g., holding an ice cream
  2. Do an action on a state R(s, a), e.g., eating the ice cream
  3. Making a transition R(s, a, s'), e.g., feeling good after eating





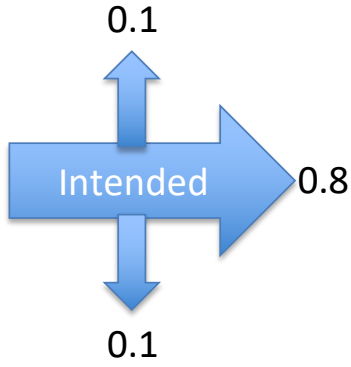


Type I: R(s) Type II: R(s,a) Type III: R(s,a,s')

Reward and Action Example

first type

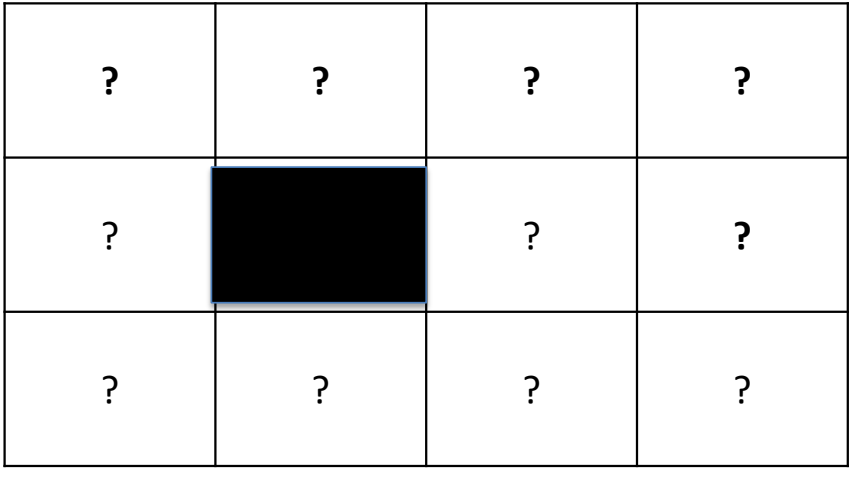
-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04 (start)	-0.04	-0.04	-0.04

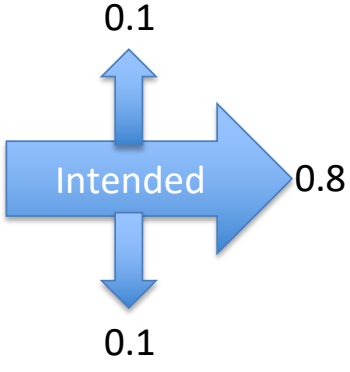


Rewards (not utility) as shown:
Two terminal states: -1, +1 (goal)
-0.04 for all nonterminal states

Actions: >, <, ^, v,
outcome probability:
0.8 for the intended
0.2 sideways
// bounce back if hits wall
// In a terminal state, the probability
// to go anywhere is 0.0

State Utility Value Example





State Utility Values:

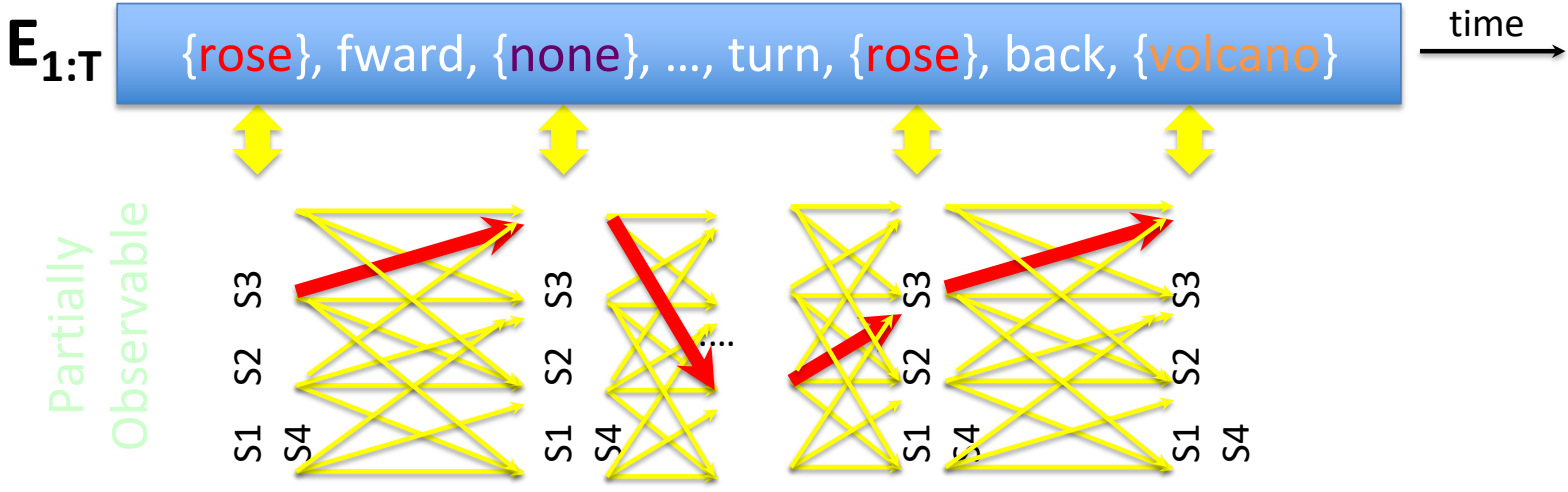
Must be learned or computed by the agent

Initially, could be all 0.0

4.

partially observable

Little Prince in Action (POMDP)



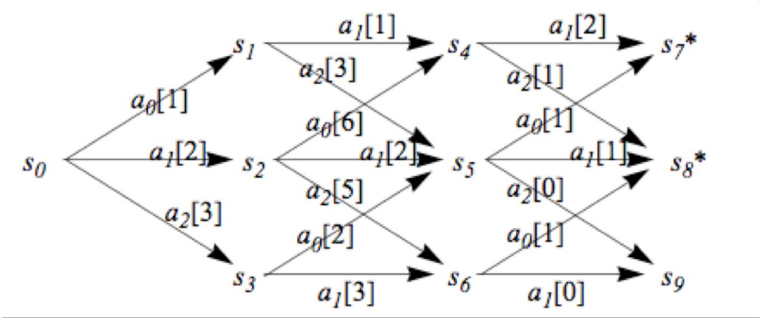
- Given: "*experience*" $\mathbf{E}_{1:T}$ (time 1 through T)
- Definitions:
  - State S and actions A (Forward, Back, Turn)
  - (Action model) Transition Probabilities Φ
  - (Sensor model) Appearance Probabilities θ (rose, volcano, none)
  - (localization) Initial/current State Probabilities π
- Partially Observable: agent sees only the percepts, not the states

State Utility and Decision on Actions

- Combine the following together
  - Partially observable action/perception model
    - E.g., Little Prince Example
  - Compute state utility values from rewards (goals)
    - E.g., use Dynamic Programming (next slide)
    - Bellman equations, Bellman iterations (later slides)
  - Policy (decide actions based on state utility values)
    - To gain as much as rewards as you can
    - To go to the goals as close as you can

Compute State Values by Dynamic Programming
(review, from ALFE 6.1.1)

Backward recursion: compute utility/values by backing from the goal* stage by stage



Rewards are given $a_i[R] = R(s,a_i)$ from an action a_i on a state s

are recorded. For example, the best value one can get from state s_4 to a goal state is 2, denoted as $V(s_4) = 2$. Similarly, $V(s_5) = 1$ and $V(s_6) = 1$. These

$$
\begin{aligned}
V(s_1) &= \max\{R(s_1,a_1) + V(s_4), R(s_1,a_2) + V(s_5)\} = \max\{1+2, 3+1\} = 4 \\
V(s_2) &= \max\{R(s_2,a_0) + V(s_4), R(s_2,a_1) + V(s_5), R(s_2,a_2) + V(s_6)\} \\
&= \max\{6+2, 2+1, 5+1\} = 8 \\
V(s_3) &= \max\{R(s_3,a_0) + V(s_5), R(s_3,a_1) + V(s_6)\} = \max\{2+1, 3+1\} = 4
\end{aligned}
$$

$$
\begin{aligned}
V(s_0) &= \max\{R(s_0,a_0) + V(s_1), R(s_0,a_1) + V(s_2), R(s_0,a_2) + V(s_3)\} \\
&= \max\{1+4, 2+8, 3+4\} = 10
\end{aligned}
$$

Backward recursion equation: $V(s_i) = \max_a\{R(s_i,a) + V(s_j)\}$ where $s_i \xrightarrow{a} s_j$

Markov Decision Process (MDP)

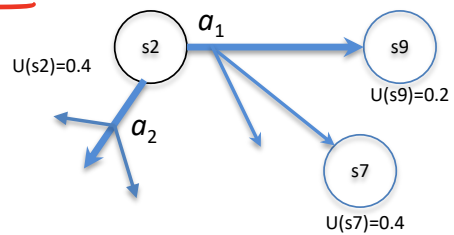
- A MDP consists of
  - State S and actions A
  - Initial State s_0 Probability Distribution π
  - Transition Model Φ(s'|s,a)
  - ~~Sensor Model θ(z|s)~~ what you see is what you get
  - A reward function R(s)

Note: A typical MDP has no sensor model

Maximum Expected Utility (MEU) and Rational Agents

- Every state has a utility value $U(s)$
- The expected utility of an action given the current evidence or observation e, is the average utility value of the outcomes, weighted by the probability that the outcome occurs:

$$EU(a \mid e) = \sum_{s'} P(result(a) = s' \mid a, e) U(s')$$



- The principle of maximum expected utility (MEU) is that a **rational** agent should choose the action that maximizes its expected utility:

$$action = \arg\max_{a} EU(a \mid e)$$

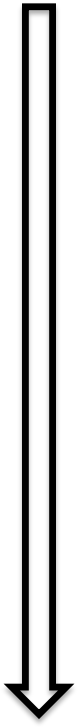
POMDP: Sensor Model, Belief States

- A Partially Observable MDP (POMDP) is defined as follows:
  - A set of states S (with an initial state s_0)
  - A set of Actions: A(s) of actions in each state s
  - A transition model P(s'|s,a), or T(s,a,s')
  - A reward function R(s), or R(s,a)
  - A sensor model P(*e*|*s*)
  - A belief of what the current state is *b*(*s*)
- Belief States (where am I now? What is my current state?):
  - If *b*(*s*) was the previous belief state, and the robot does action "*a*" and then perceives a new evidence "*e*", then the new belief state:

$$b'(s') = \alpha P(e \mid s') \sum P(s' \mid s,a) b(s)$$

where α is a normalization constant making the belief states sum to 1

Goals, Rewards, Utilities, Policies



- Goals
  - Given to the agent from the problem statements
- Rewards
  - Given to the agent, designed based on the goals
- Utility values for states
  - Computed by the agent, based on the rewards
- Policies
  - Computed or learned by the agent
  - Used by the agent to select its actions
  - The better a policy, the more rewards it collects

Compute Utilities from Rewards over Time

- Utility Value = sum of all future rewards
  - Add the rewards as they are

$$U_h = ([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$$

  - Discount the far-away rewards in the future

$$U_h = ([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

- The expected future utility value $U^{\pi}(s)$ obtained by executing a policy π starting from s

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

The Optimal Policy

$$\pi_s^* = \arg\max_{\pi} U^{\pi}(s)$$

Compute Utilities (example)

-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04 (start)	-0.04	-0.04	-0.04

Rewards (given)

0.812	**0.868**	**0.918**	**+1**
0.762		**0.660**	**-1**
0.705	**0.655**	**0.611**	**0.388**

Utilities (computed)

Given the rewards in all nonterminal state are R(s) = - 0.04,
Compute utilities using a future discount γ = 1
(we will see how these utilities are computed)

2. State Utility Value Iteration

(improving $U(s)$ every step)

- $U(s)$: the expected sum of maximum rewards achievable starting from a particular state s
- Bellman equations:
  - For n states, there are n equations must be solved *simultaneously*

$$U^*(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U^*(s') \qquad (17.5)$$

- Bellman iteration:
  - Converge to $U^*(s)$ step by step

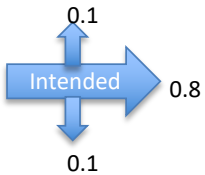
$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s') \qquad (17.6)$$

Bellman Iteration Example

Reward *R(s)*

-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04	-0.04	-0.04	-0.04

Transaction probability
T(s, a, s') = 0.8 for intended,
0.2 for sideways
Discount Gamma γ = 1.0



The Initial Utility Values

0.0	0.0	0.0	0.0
0.0		0.0	0.0
0.0	0.0	0.0	0.0

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$$

0.812	0.868	0.918	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

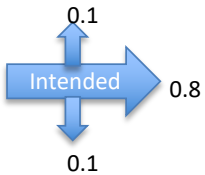
The Final Utility Values

Bellman Iteration Example (1st iteration)

Reward $R(s)$

-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04	-0.04	-0.04	-0.04

Transaction probability
T(s, a, s') = 0.8 for intended,
0.2 for sideways
Discount Gamma γ = 1.0



The Initial Utilities U_0

0.0	0.0	0.0	0.0
0.0		0.0	0.0
0.0	0.0	0.0	0.0

$U_1(s_1)$ = -0.04 + 1.0*max(0.0*0.0)

$U_1(s_4)$ = +1.0 + 1.0*max(0.0*0.0)
$U_1(s_7)$ = -1.0 + 1.0*max(0.0*0.0)

-0.04	-0.04	-0.04	+1.0
-0.04		-0.04	-1.0
-0.04	-0.04	-0.04	-0.04

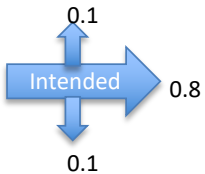
The Utility Values U_1 after 1st iteration

Bellman Iteration Example (2nd iteration)

Reward *R(s)*

-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04	-0.04	-0.04	-0.04

Transaction probability
T(s, a, s') = 0.8 for intended,
0.2 for sideways
Discount Gamma γ = 1.0



U_1

-0.04	-0.04	-0.04	+1.0
-0.04		-0.04	-1.0
-0.04	-0.04	-0.04	-0.04

$U_2(s_4)$ = +1.0 + 1.0*max(0, 0, 0)
$U_2(s_7)$ = -1.0 + 1.0*max(0, 0, 0)
$U_2(s_3)$ = -.04 + 1.0*max(.8*1-.1*.04-.1*.04, …)
$U_2(s_6)$ = -.04 + 1.0*max(-.8*.04-.1*.04-.1*.04,…)

U_2

?	?	0.752	+1.0
?		-0.08	-1.0
?	?	?	?

The Utility Values U_2 after 2nd iteration
Please fill in the values for "?"

Bellman Iteration Example

Reward *R(s)*

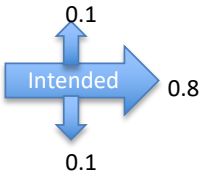
-0.04	-0.04	-0.04	**+1**
-0.04		-0.04	**-1**
-0.04	-0.04	-0.04	-0.04

The Initial Utilities

0.0	0.0	0.0	0.0
0.0		0.0	0.0
0.0	0.0	0.0	0.0

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$$

Transaction probability
T(s, a, s') = 0.8 for intended,
0.2 for sideways
Discount Gamma γ = 1.0



0.812	0.868	0.918	+1
0.762		0.660	-1
0.705	0.655	0.611	0.388

Please verify these final Utility Values by yourself!!

3. Utility Value Iteration (3 types of rewards)

- When rewards are given as R(s)
  - $U_{t+1} \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s') U_t(s')$
- When rewards are given as R(s,a)
  - $U_{t+1} \leftarrow R(s,a) + \gamma \max_a \sum_{s'} T(s,a,s') U_t(s')$
- When rewards are given as R(s,a,s')
  - $U_{t+1} \leftarrow \gamma \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + U_t(s')]$
  - $U_{t+1} \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U_t(s')]$

Utility Value Iteration for POMDP

- In MDP, utility value iteration, when sensors are certain, we know the next state *s'*

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U_i(s')$$

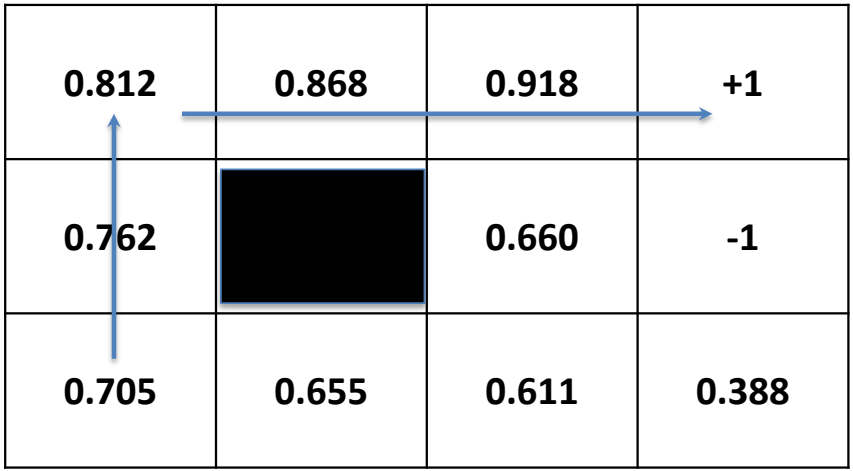
- In POMDP, where sensors are uncertain, we must average over all possible evidences for the next state *s'* (see the last term below):

$$\alpha_p(s) = R(s) + \gamma \left(\sum_{s'} T(s,a,s') \sum_e P(e \mid s') \alpha_{p.e}(s') \right)$$

Policy and Optimal Policy

- A solution of (PO)MDP can be represented as
  - A Policy $\pi(s)=a$
  - Each execution of policy from s_0 may yield a different history or path due to the probabilistic nature of the transition model
  - An optimal policy $\pi^*(s)$ is a policy that yields the highest expected utility

Policy Based on Known Utility Values

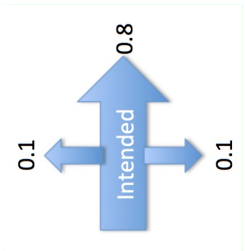


Suppose we know the utility values of the states as above,
What path would an optimal policy would choose?

Rewards -> Utility Values -> Policies

Optimal Policy Examples
Depending on **Reward** distribution R(s)

If the **rewards** for the nonterminal states are evenly distributed (e.g., -0.04), then the path chosen will depend on the probabilities of the transition model



3	->	->	->	+1
2	^		^	-1
1	^	<-	^	<-
	1	2	3	4

Nonterminal R(s)= - 0.04

Caution: because the nondeterministic actions, from state (3,2) or (4,1), you may "accidentally" go to (4,2). So there is a "risk" in this policy.

Optimal Policy Examples

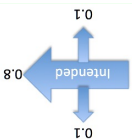
Depending on the reward distribution R(s)

Nonterminal rewards R(s)=-0.04

>	>	>	+1
^		^	-1
^	<	< ?	<

Why not go up?

Hint: Consider the nature of actions, deterministic or not
Go up is "risky"



>	>	>	+1
^		> suicide	-1
^	>	>	^ suicide

R(s)<-1.6284
(life is painful, death is better)

>	>	>	+1
^		^ risky	-1
^	>	^	<

-0.4278 < R(s)<-0.0850
(life is OK, willing to risk)

>	>	> end nicely	+1
^		< no risk	-1
^	<	<	< risky

-0.0221<R(s)<0
(life is good, minimize risks, willing to end nicely)

❖	❖	< don't end	+1
❖		< no risk	-1
❖	❖	❖	V no risk

R(s)>0
(life is rewarding, I don't want to end it)

Optimal Policy

- You can compute the optimal policy once after all $U^*(s)$ are known

$$\pi_s^* = \underset{a}{\operatorname{argmax}} \sum T(s,a,s')U^*(s')$$

- Or, you can compute it incrementally every iteration when $U_i(s)$ is updated
  - This is called "Policy Iteration" (see the next slide)

Policy Iteration
(improving π(s) every step)

- Start with a randomly chosen initial policy π_0
- Iterate until no change in utility values of the state:
- Policy evaluation: given a policy π_i, calculate the utility $U_i(s)$ of every state *s* using policy π_i by solving the system of equations:

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

- Policy improvement: calculate the new policy π_{i+1} using one-step look-ahead based on the current $U_i(s)$:

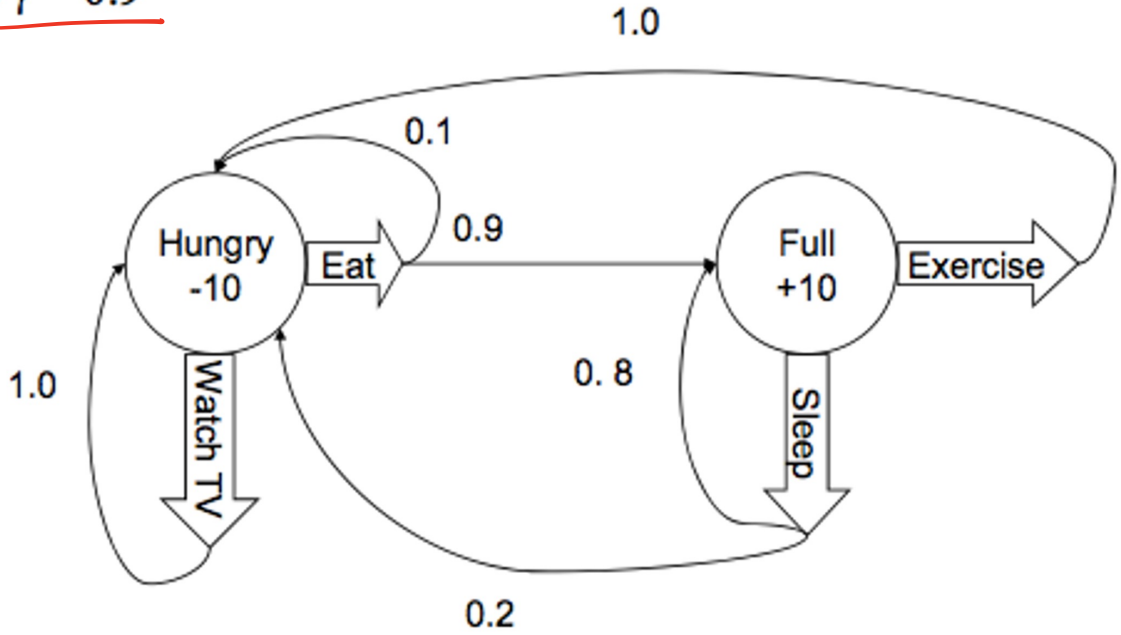
$$\pi_{i+1}(s) = \operatorname*{argmax}_a \sum_{s'} T(s, a, s') U^*(s')$$

Policy Iteration Comments

- Each step of policy iteration is guaranteed to strictly improve the policy at some state when improvement of state utility value is possible
- Converge to the optimal policy
- Gives the exact value of the optimal policy

Policy Iteration Example

Do one iteration of policy iteration on the MDP below. Assume an initial policy of $\pi_1(\text{Hungry}) = \text{Eat}$ and $\pi_1(\text{Full}) = \text{Sleep}$.
Let $\gamma = 0.9$



Policy Iteration Example

Use initial policy for Hungry: $\pi_1(\text{Hungry}) = \text{Eat}$

$$U_1(\text{Hungry}) = -10 + (0.9)[(0.1)U_1(\text{Hungry}) + (0.9)U_1(\text{Full})]$$
$$\Rightarrow U_1(\text{Hungry}) = -10 + (0.09)U_1(\text{Hungry}) + (0.81)U_1(\text{Full})$$
$$\Rightarrow (0.91)U_1(\text{Hungry}) - (0.81)U_1(\text{Full}) = -10$$

Use initial policy for Full: $\pi_1(\text{Full}) = \text{Sleep}$.

$$U_1(\text{Full}) = 10 + (0.9)[(0.8)U_1(\text{Full}) + (0.2)U_1(\text{Hungry})]$$
$$\Rightarrow U_1(\text{Full}) = 10 + (0.72)U_1(\text{Full}) + (0.18)U_1(\text{Hungry})]$$

Policy Iteration Example

$(0.91)U_1(\text{Hungry})-(0.81)U_1(\text{Full}) = -10$ ....(Equation 1)

$(0.28)U_1(\text{Full}) - (0.18)U_1(\text{Hungry})=10$ ...(Equation 2)

} Solve for $U_1(\text{Hungry})$ and $U_1(\text{Full})$

From Equation 1:

$(0.91)U_1(\text{Hungry}) = -10+(0.81)U_1(\text{Full})$

$\Rightarrow U_1(\text{Hungry}) = (-10/0.91)+(0.81/0.91)U_1(\text{Full})$

$\Rightarrow U_1(\text{Hungry})=-10.9+(0.89)U_1(\text{Full})$

Substitute $U_1(\text{Hungry})=-10.9+(0.89)U_1(\text{Full})$ into Equation 2

$(0.28)U_1(\text{Full}) - (0.18)[-10.9+(0.89)U_1(\text{Full})]=10$

$\Rightarrow(0.28)U_1(\text{Full}) + 1.96-(0.16)U_1(\text{Full})=10$

$\Rightarrow(0.12)U_1(\text{Full})=8.04$

$\Rightarrow U_1(\text{Full})=67$

$\Rightarrow U_1(\text{Hungry})=-10.9+(0.89)(67)=-10.9+59.63=48.7$

Policy Iteration Example

$$\pi_2(\text{Hungry})$$

$$= \underset{\{\text{Eat, WatchTV}\}}{\text{argmax}} \begin{cases} \text{T(Hungry, Eat, Full)U}_1\text{(Full)} + \\ \quad \text{T(Hungry, Eat, Hungry)U}_1\text{(Hungry)} & \text{[Eat]} \\ \text{T(Hungry, WatchTV, Hungry)U}_1\text{(Hungry)} & \text{[WatchTV]} \end{cases}$$

$$= \underset{\{\text{Eat, WatchTV}\}}{\text{argmax}} \begin{cases} (0.9)\text{U1(Full)} + (0.1)\text{U1(Hungry)} & \text{[Eat]} \\ (1.0)\text{U1(Hungry)} & \text{[WatchTV]} \end{cases}$$

$$= \underset{\{\text{Eat, WatchTV}\}}{\text{argmax}} \begin{cases} (0.9)(67) + (0.1)(48.7) & \text{[Eat]} \\ (1.0)(48.7) & \text{[WatchTV]} \end{cases}$$

$$= \underset{\{\text{Eat, WatchTV}\}}{\text{argmax}} \begin{cases} 65.2 & \text{[Eat]} \\ 48.7 & \text{[Watch]} \end{cases}$$

$$= \text{Eat}$$

Policy Iteration Example

$$\pi_2(\text{Full})$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} T(\text{Full,Exercise,Hungry})U_1(\text{Hungry}) & [\text{Exercise}] \\ T(\text{Full,Sleep,Full})U_1(\text{Full}) + & \\ \quad T(\text{Full,Sleep,Hungry})U_1(\text{Hungry}) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} (1.0)U_1(\text{Hungry}) & [\text{Exercise}] \\ (0.8)U_1(\text{Full}) + (0.2)U_1(\text{Hungry}) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} (1.0)(48.7) & [\text{Exercise}] \\ (0.8)(67) + (0.2)(48.7) & [\text{Sleep}] \end{cases}$$

$$= \underset{\{\text{Exercise,Sleep}\}}{\text{argmax}} \begin{cases} 48.7 & [\text{Exercise}] \\ 63.34 & [\text{Sleep}] \end{cases}$$

$$= \text{Sleep}$$

Policy Iteration Example

- The new policy π_2 after an iteration from π_1
  - π_2(Hungry)->Eat
  - π_2(Full)->Sleep

Summary for Uncertain Environments

- POMDP is very (the most) general model
  - Deal with uncertainty by
    - action models and sensor models
- Incorporate goals -> rewards -> utilities -> policy
  - Utilities and policies can be computed from rewards
    - Systematically (Bellman equations)
    - Iteratively (Bellman's iteration algorithm)
  - Solve a problem by following a good policy
    - One policy for one problem/goal
    - Different policies are needed for different goals