Markov Decision Process

Thomas Moerland

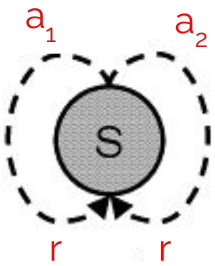
Leiden University

Bandits

One-step decision-making problem

Bandits

One-step decision-making problem

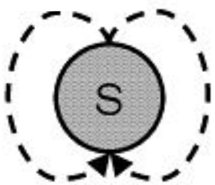


Bandit

(no state/state fixed)

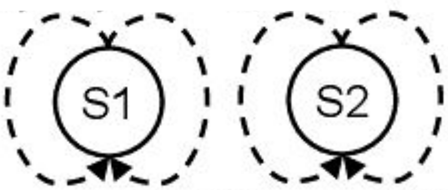
Bandits

One-step decision-making problem



Bandit

(no state/state fixed)

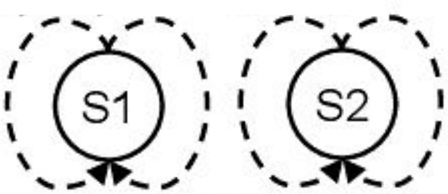


Bandits

One-step decision-making problem

Bandit

(no state/state fixed)



Contextual bandit

(state changes action rewards)

Bandits

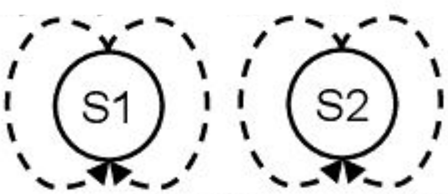
One-step decision-making problem

Markov Decision Process

Sequential decision-making problem

Bandit

(no state/state fixed)

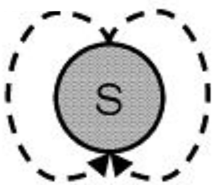


Contextual bandit

(state changes action rewards)

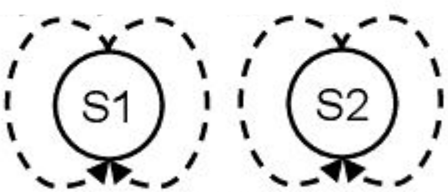
Bandits

One-step decision-making problem



Bandit

(no state/state fixed)

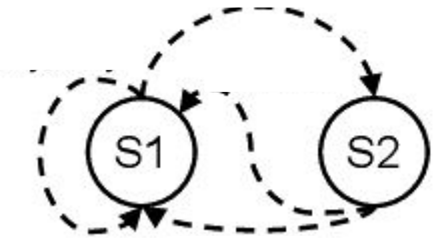


Contextual bandit

(state changes action rewards)

Markov Decision Process

Sequential decision-making problem

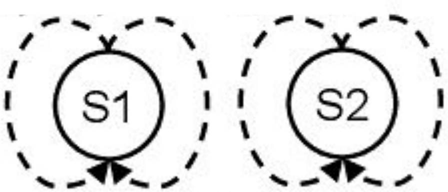


Bandits

One-step decision-making problem

Bandit

(no state/state fixed)

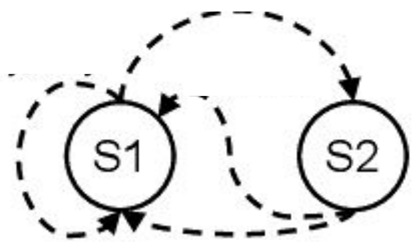


Contextual bandit

(state changes action rewards)

Markov Decision Process

Sequential decision-making problem



Markov Decision Process
(actions influence what next state you see

-

makes the problem *sequential*: we may prefer a low instant reward if it gives us high long-term reward)

Content

Content

1. Sequential Decision Making (relevance)

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)
3. Markov Decision Process (problem definition)

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)
3. Markov Decision Process (problem definition)

Break

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)
3. Markov Decision Process (problem definition)

Break

4. Policy (solution space)

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)
3. Markov Decision Process (problem definition)

Break

4. Policy (solution space)
5. Return

Content

1. Sequential Decision Making (relevance)
2. Conceptual Example (high-level overview)
3. Markov Decision Process (problem definition)

Break

4. Policy (solution space)
5. Return
6. Value (objective)

Content

1.	Sequential Decision Making	(relevance)
2.	Conceptual Example	(high-level overview)
3.	Markov Decision Process	(problem definition)
Break		
4.	Policy	(solution space)
5.	Return	
6.	Value	(objective)
7.	Optimal value & policy	(solution)

Part I:

Sequential Decision Making

Sequential Decision Making

Sequential Decision Making

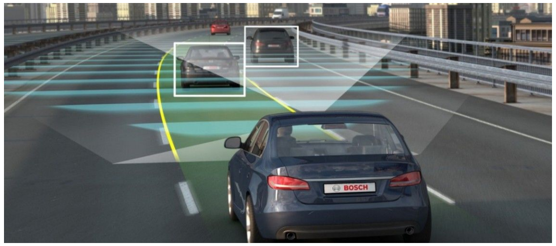
Many key problems in artificial intelligence are naturally sequential
(usually in time)

Sequential Decision Making

Many key problems in artificial intelligence are naturally sequential
(usually in time)

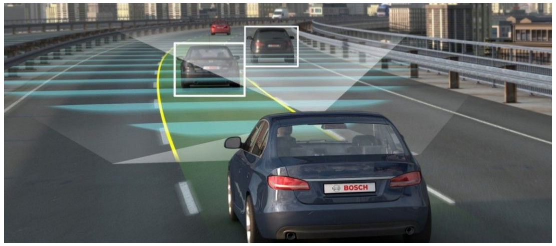
Sequential Decision Making

Many key problems in artificial intelligence are naturally sequential
(usually in time)



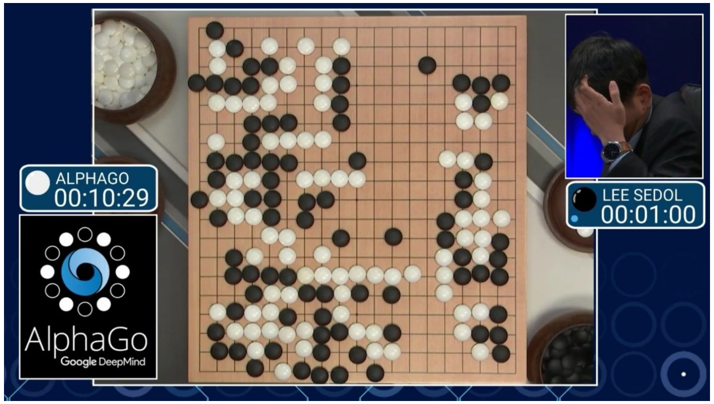
Sequential Decision Making

Many key problems in artificial intelligence are naturally sequential
(usually in time)

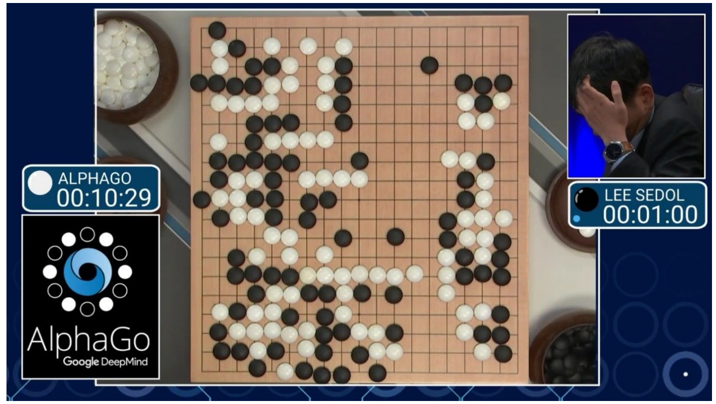


Sequential Decision Making

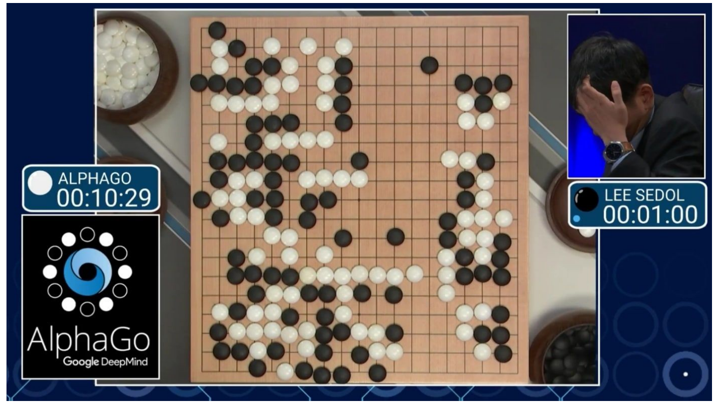
Many key successes of AI use this formulation…

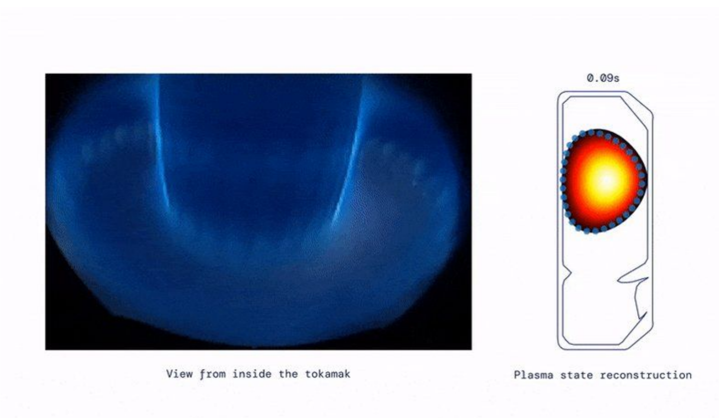

ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00


The international journal of science / 6 October 2022
nature
MATRIX GAMES
Deep reinforcement learning opens route to faster algorithms for matrix multiplication
Protecting Peru
Can technology help Indigenous groups preserve the Amazon?
Invisible touch
How marine clouds are affected by aerosols emitted from shipping
Preferential practice
US universities favour prestige in faculty hiring and retention

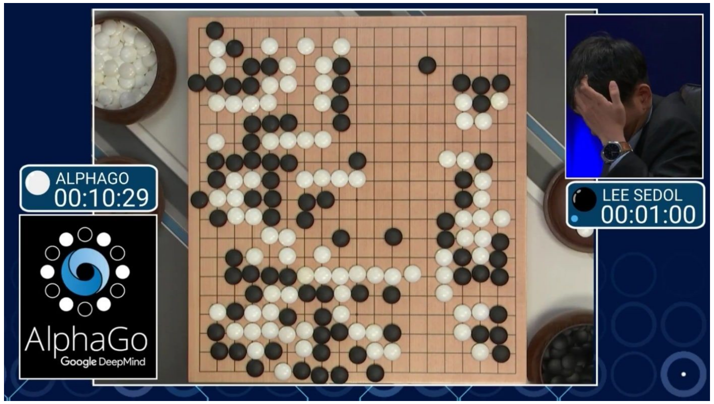

ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

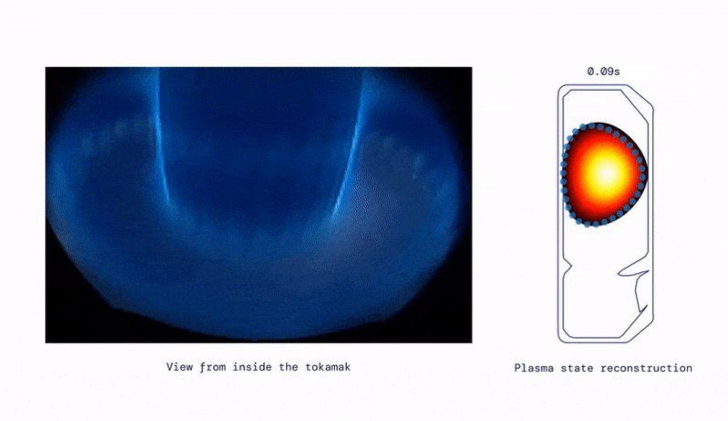

The international journal of science / 6 October 2022
nature
MATRIX GAMES
Deep reinforcement learning opens route to faster algorithms for matrix multiplication
Protecting Peru
Can technology help Indigenous groups preserve the Amazon?
Invisible touch
How marine clouds are affected by aerosols emitted from shipping
Preferential practice
US universities favour prestige in faculty hiring and retention

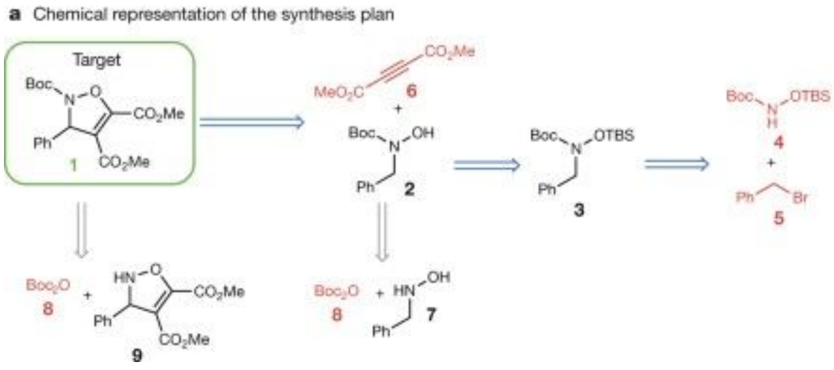

ALPHAGO
00:10:29
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind


0.09s
View from inside the tokamak
Plasma state reconstruction

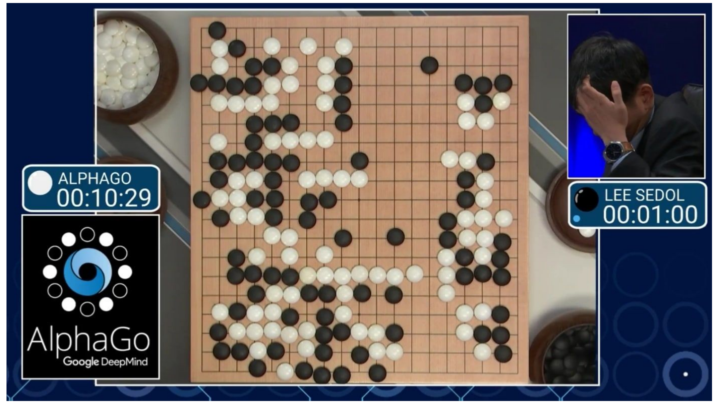

The international journal of science / 6 October 2022
nature
MATRIX GAMES
Deep reinforcement learning opens route to faster algorithms for matrix multiplication
Protecting Peru
Can technology help Indigenous groups preserve the Amazon?
Invisible touch
How marine clouds are affected by aerosols emitted from shipping
Preferential practice
US universities favour prestige in faculty hiring and retention

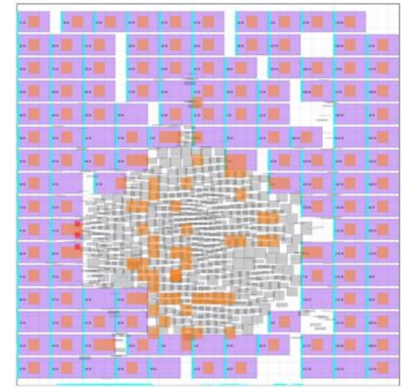

ALPHAGO
00:10:29
AlphaGo
Google DeepMind
LEE SEDOL
00:01:00

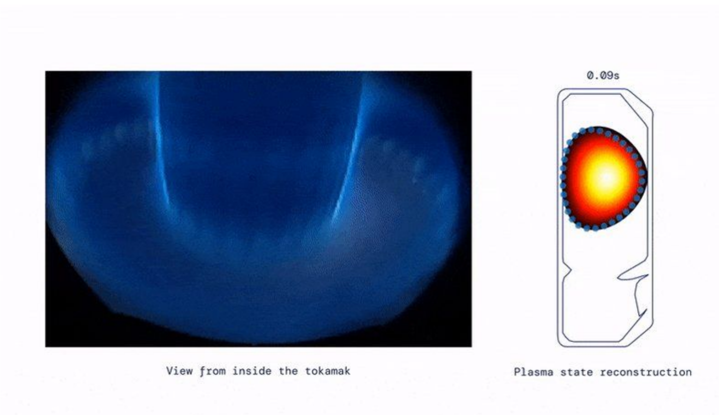

0.09s
View from inside the tokamak
Plasma state reconstruction

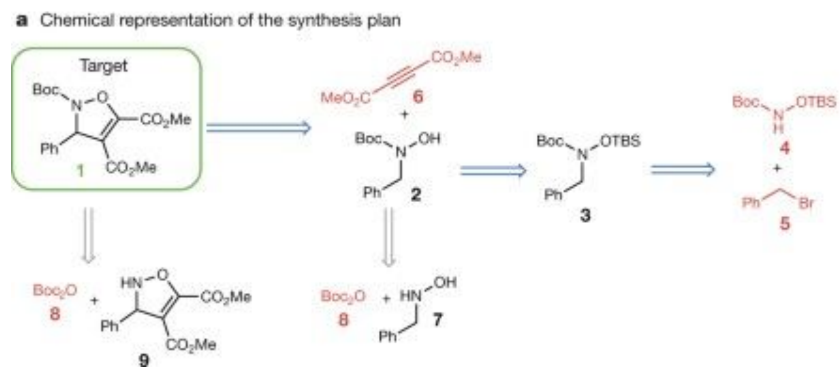

a Chemical representation of the synthesis plan
Target
Boc
CO2Me
Ph
1
6
2
3
4
5
8
9
7
MeO2C
OH
OTBS
Br
Boc2O
HN

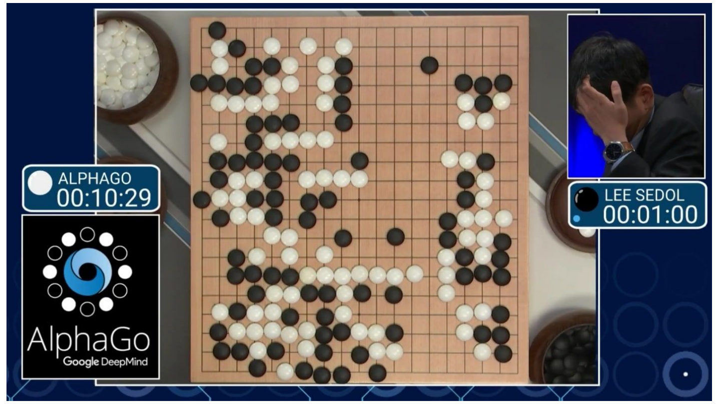

The international journal of science / 6 October 2022
nature
MATRIX GAMES
Deep reinforcement learning opens route to faster algorithms for matrix multiplication
Protecting Peru
Can technology help Indigenous groups preserve the Amazon?
Invisible touch
How marine clouds are affected by aerosols emitted from shipping
Preferential practice
US universities favour prestige in faculty hiring and retention

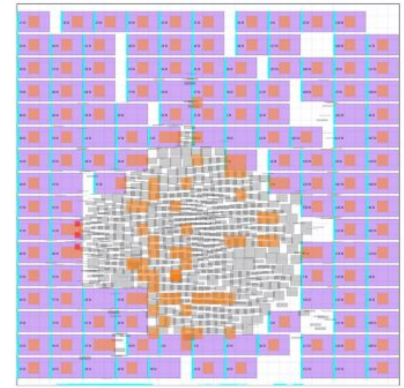

ALPHAGO
00:10:29
LEE SEDOL
00:01:00
AlphaGo
Google DeepMind

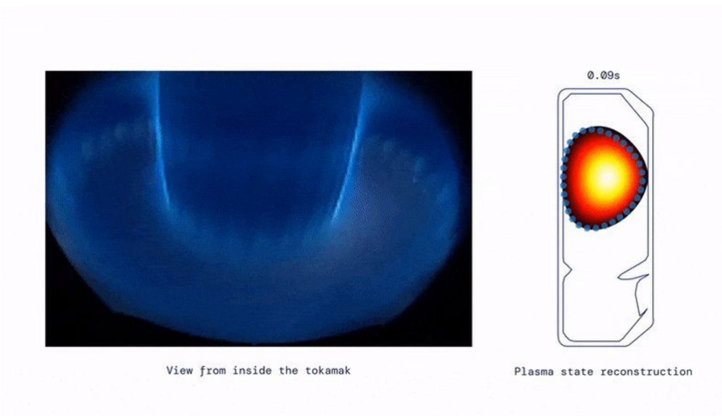


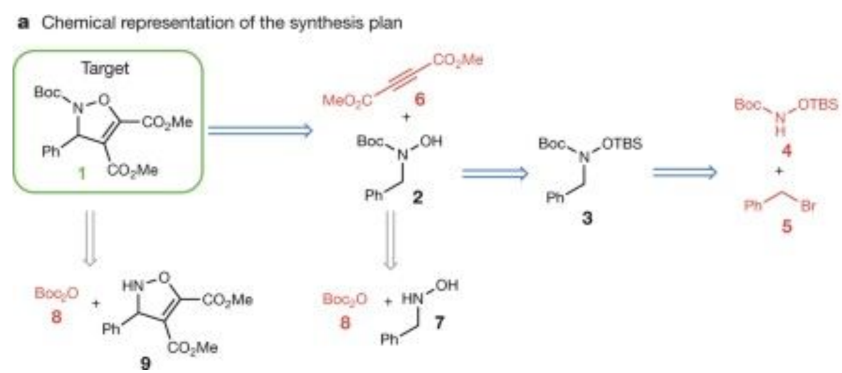

0.09s
View from inside the tokamak
Plasma state reconstruction


a Chemical representation of the synthesis plan
Target
Boc
CO2Me
Ph
1
MeO2C
6
Boc
OH
2
OTBS
3
4
Br
5
Boc2O
8
HN
9
7









Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature* 529.7587 (2016): 484-489.

Fawzi, Alhussein, et al. "Discovering faster matrix multiplication algorithms with reinforcement learning." *Nature* 610.7930 (2022): 47-53.

Degrave, Jonas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning." *Nature* 602.7897 (2022): 414-419.

Segler, Marwin HS et al. "Planning chemical syntheses with deep neural networks and symbolic AI." *Nature* 555.7698 (2018): 604-610.

Mirhoseini, Azalia, et al. "A graph placement methodology for fast chip design." *Nature* 594.7862 (2021): 207-212.

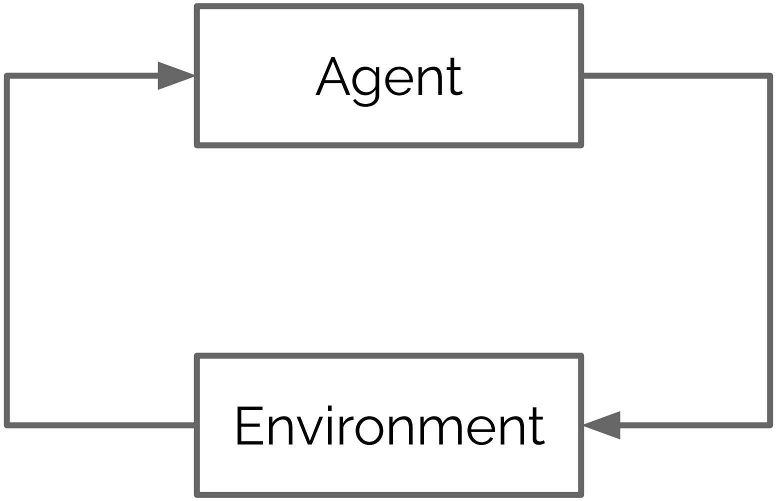
Sequential Decision Making

Many key successes of AI use this formulation…

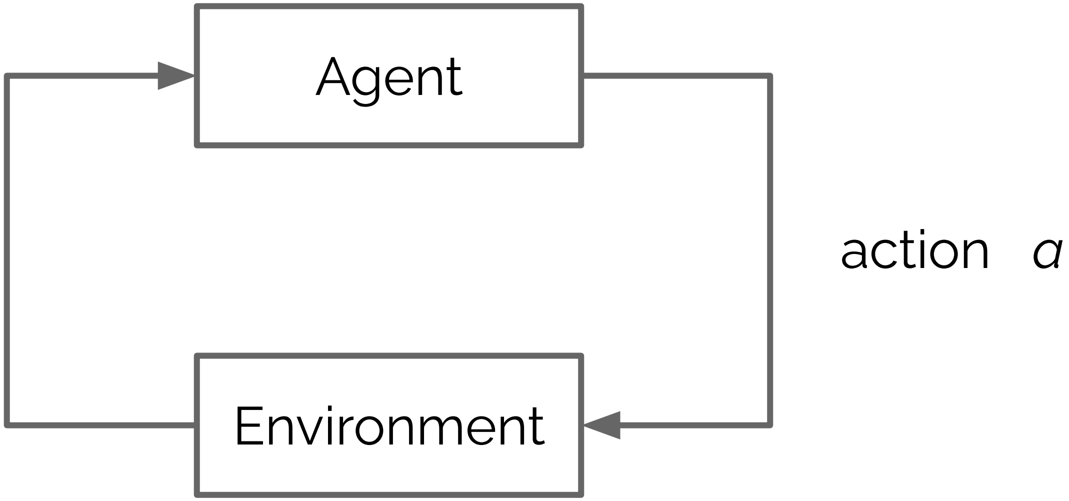
…even ones that don't seem sequential at first!

Agent-Environment loop

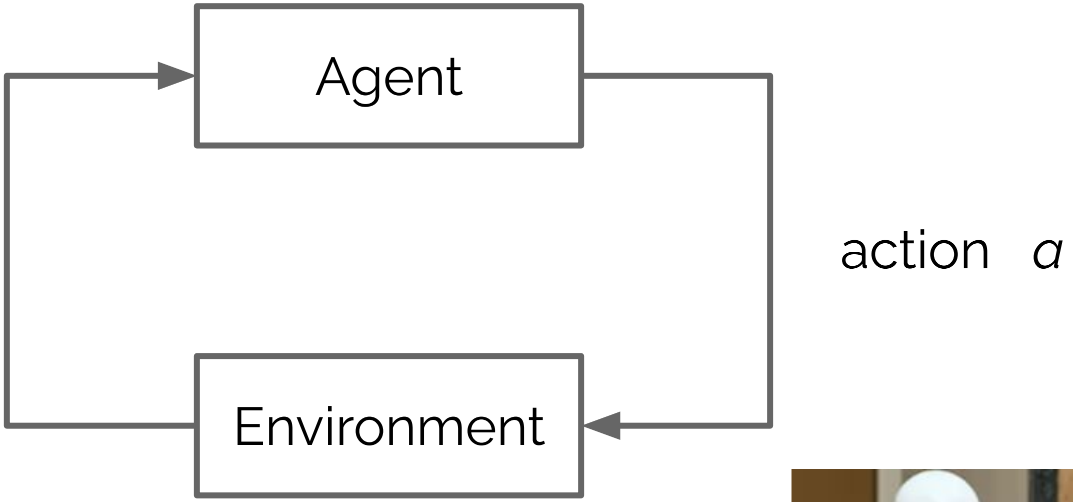
Agent-Environment loop



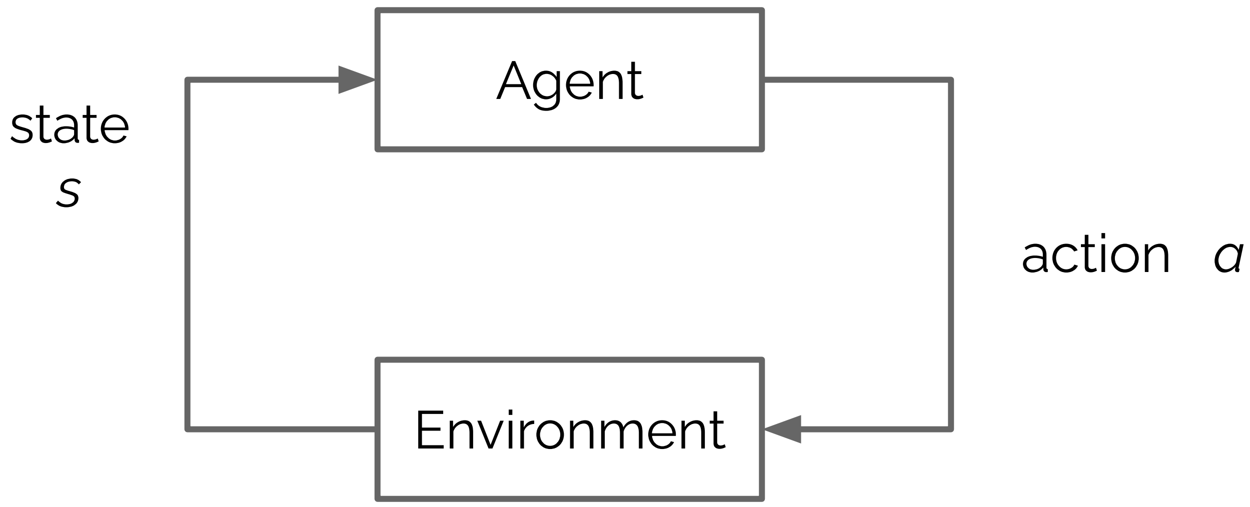
Agent-Environment loop



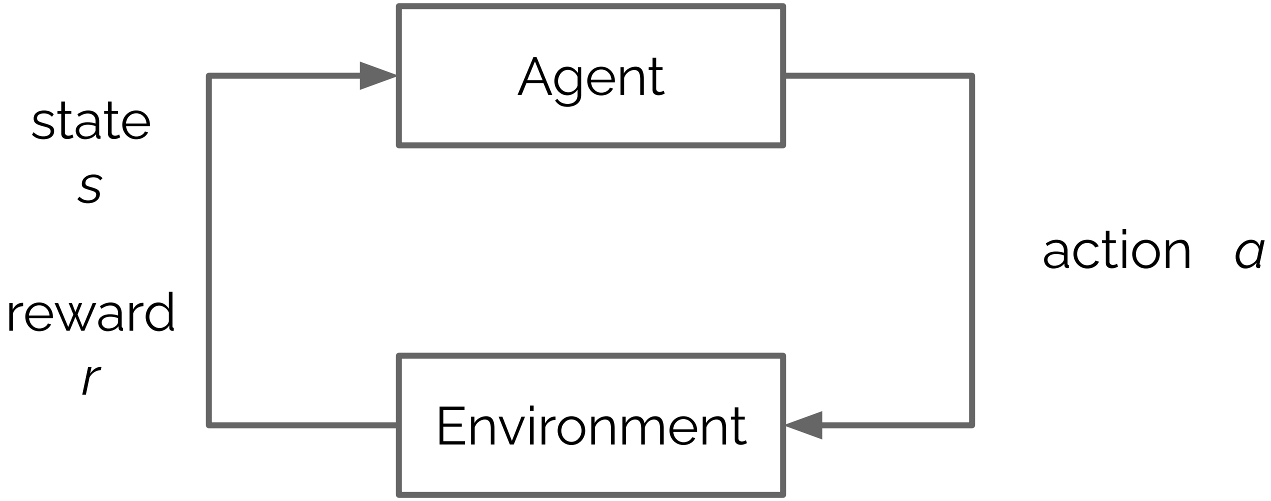
Agent-Environment loop



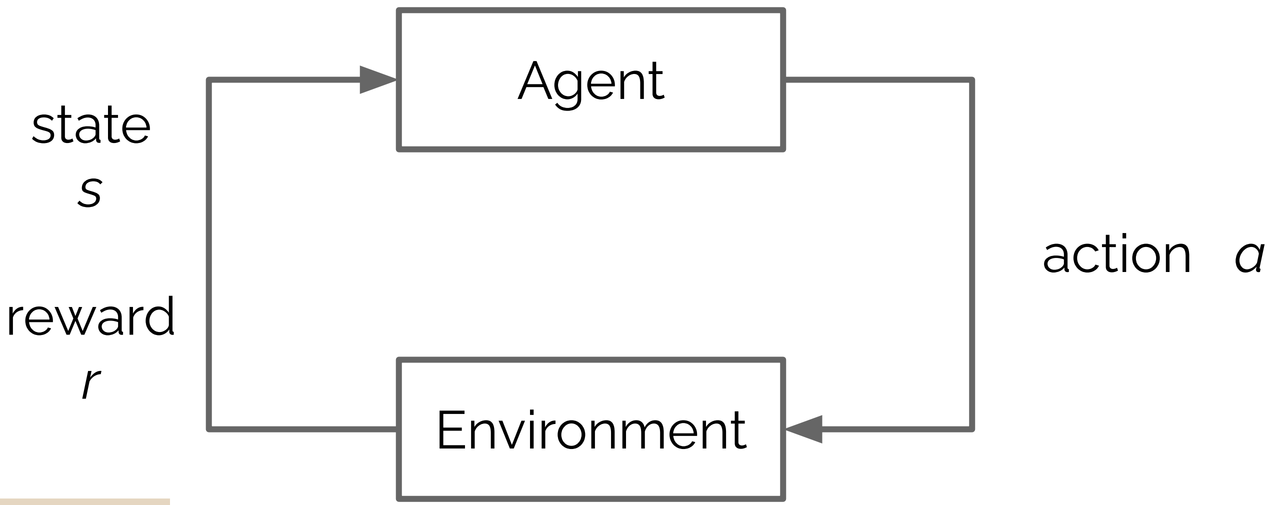
Agent-Environment loop



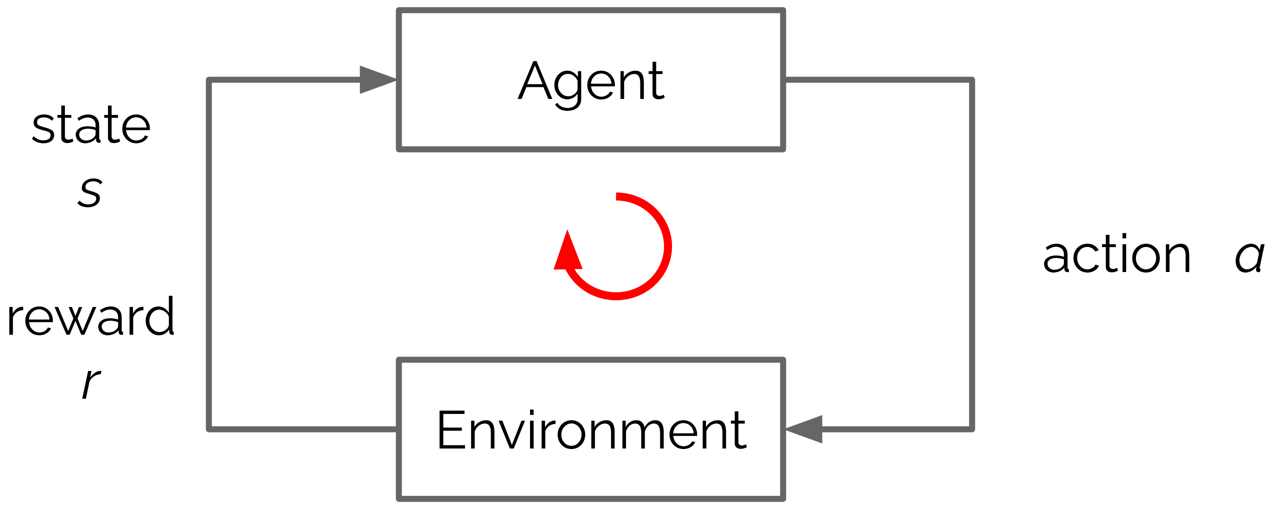
Agent-Environment loop



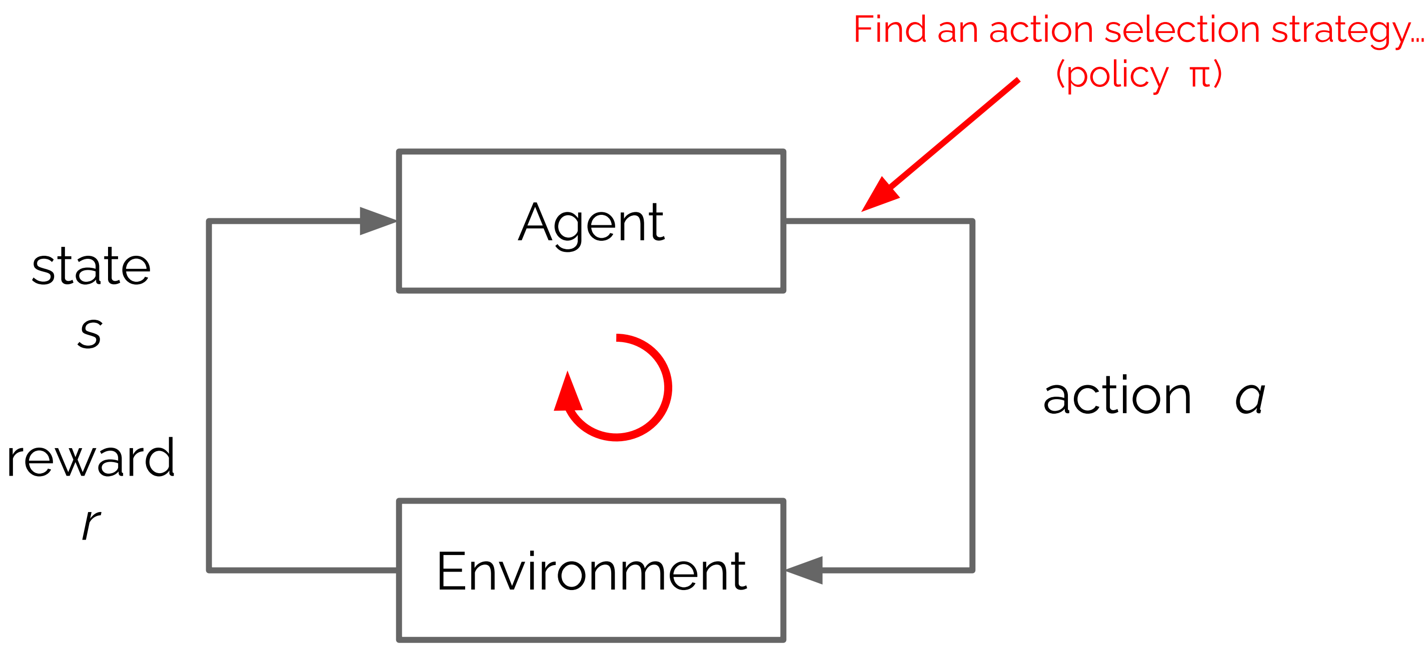
Agent-Environment loop



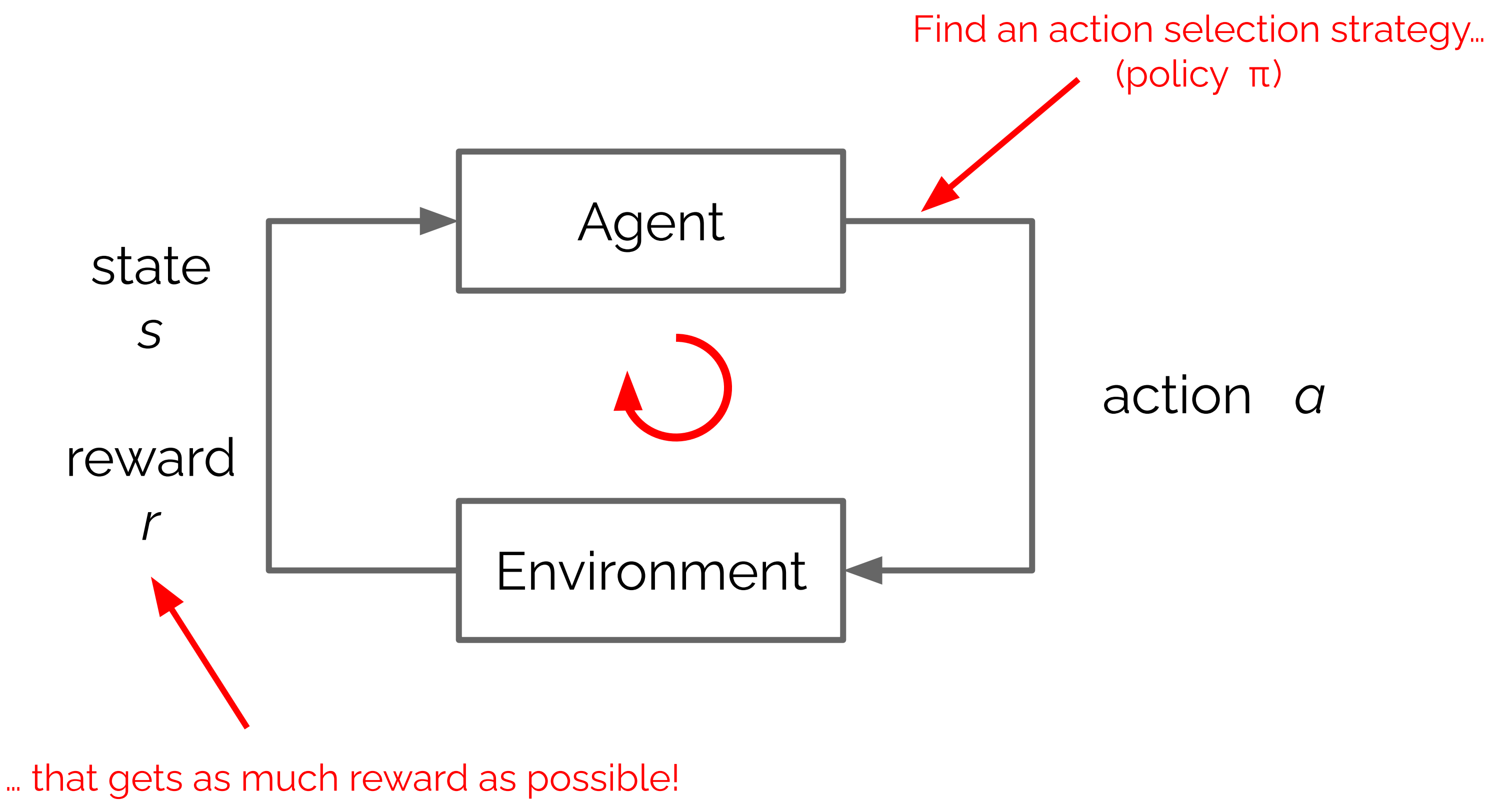
Agent-Environment loop



Agent-Environment loop



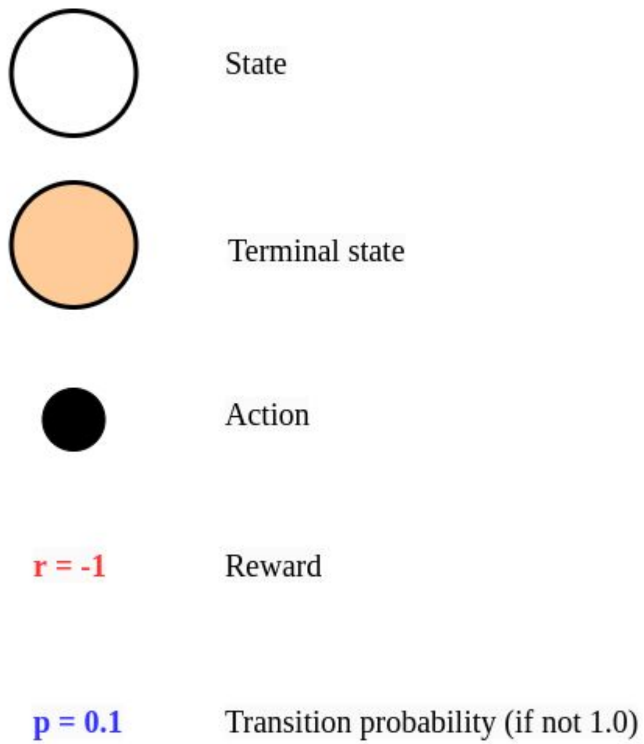
Agent-Environment loop



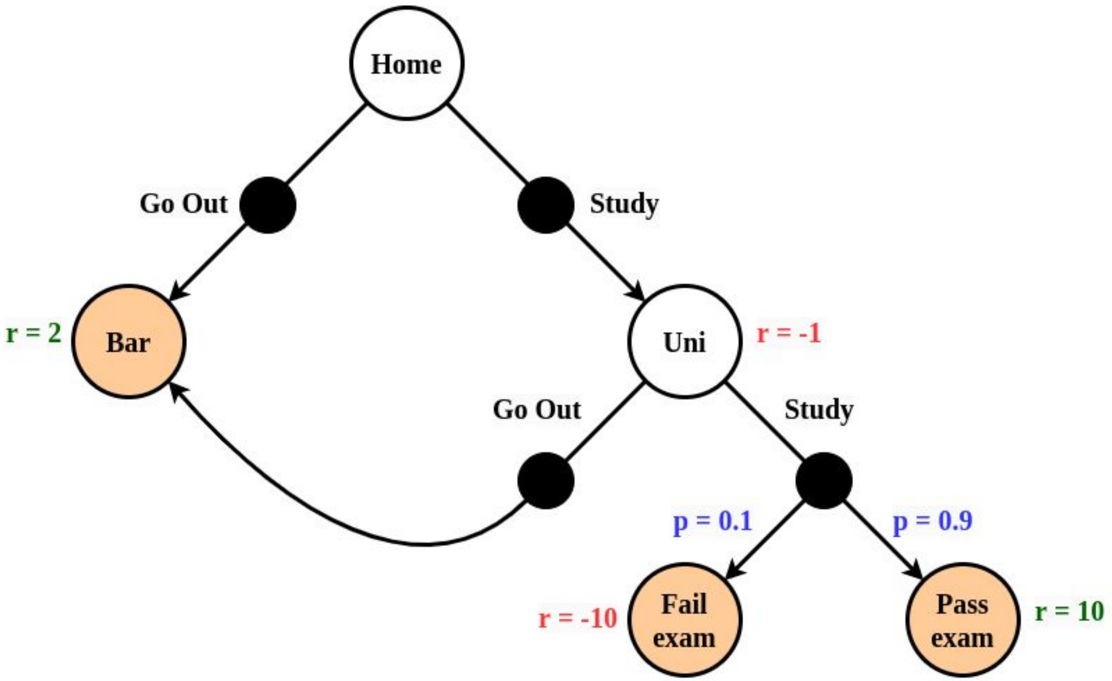
Part II

Conceptual Example
(High-level Overview)

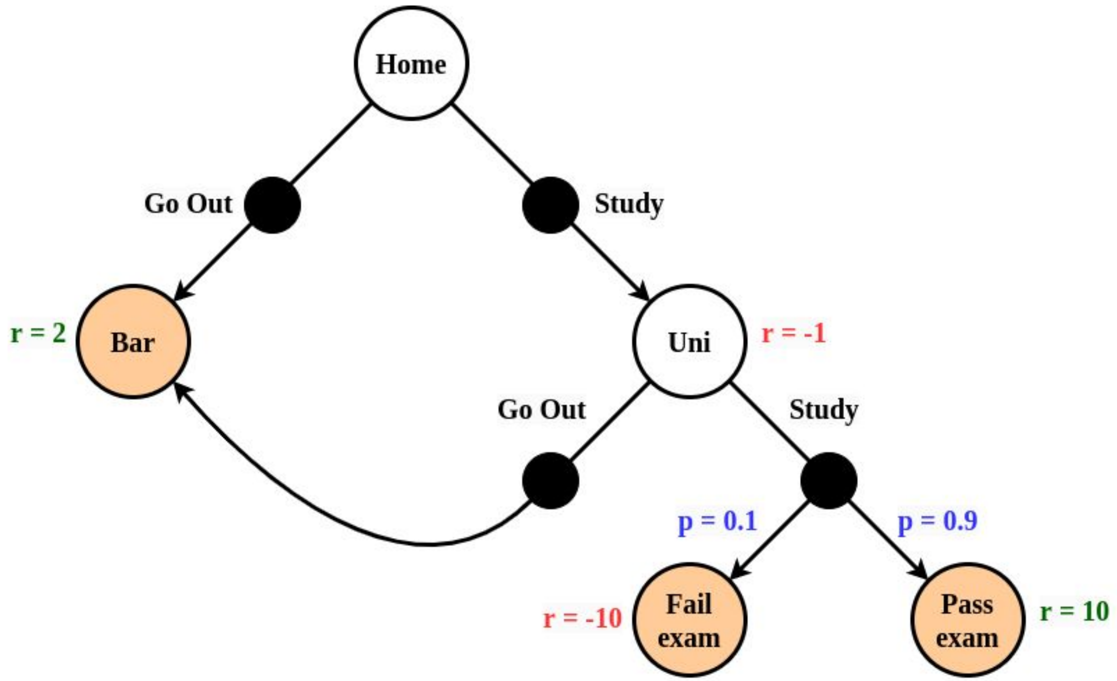
Example: Notation



Example: The Study MDP

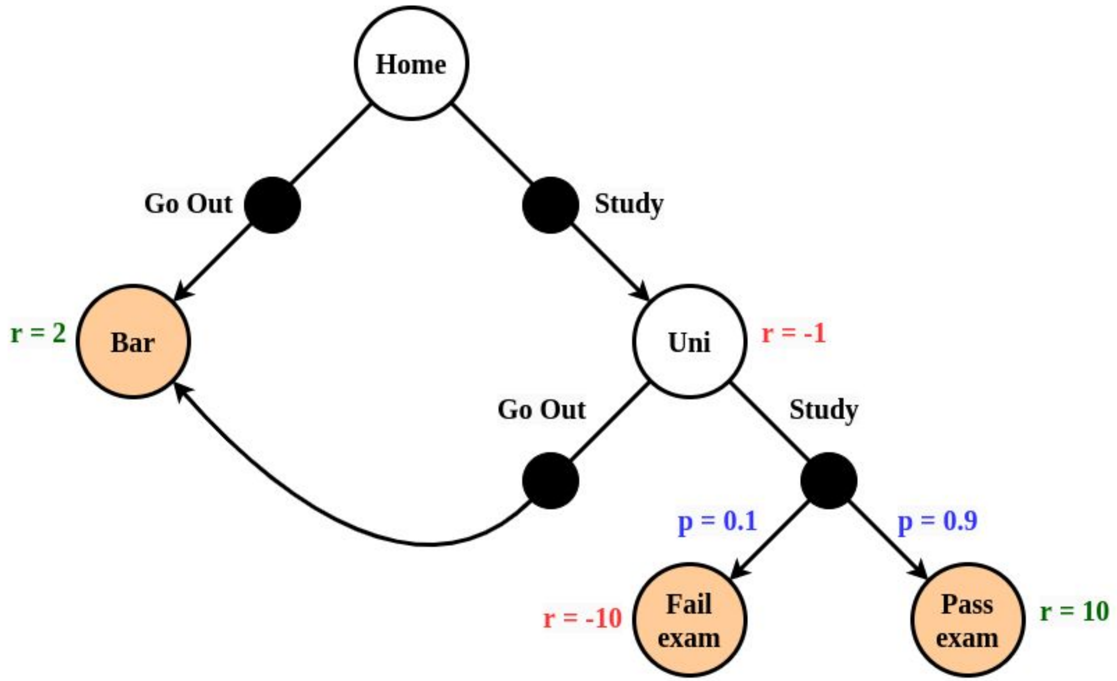


Example: The Study MDP



We are interested in the **optimal value of a state** (v^*) and the **optimal value of an action** (q^*)

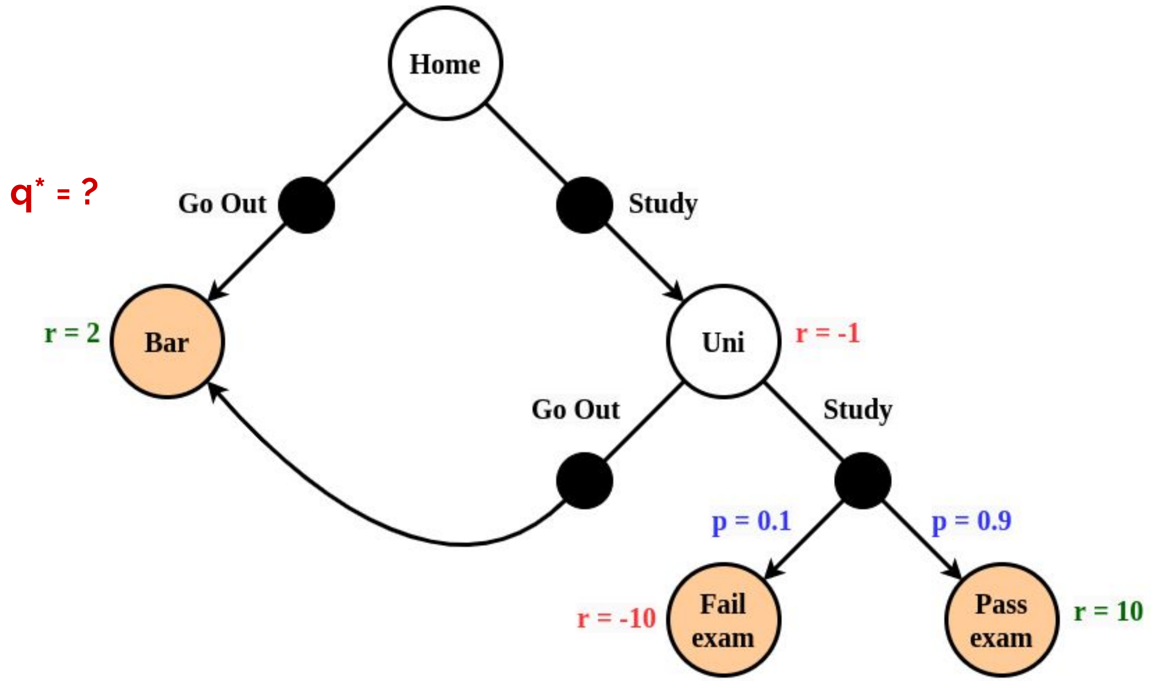
Example: The Study MDP



We are interested in the **optimal value of a state** (v^*) and the **optimal value of an action** (q^*)

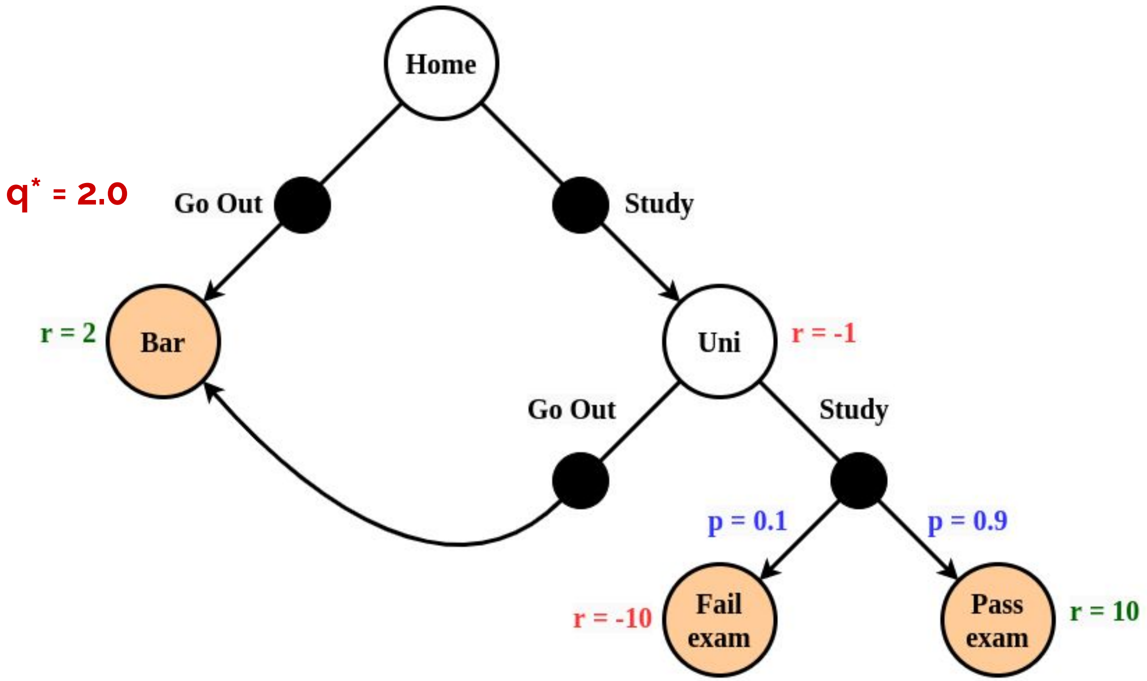
"How much reward can we at best get from that state or action"

Example: The Study MDP



Question: What is q*(Home, Go Out)?

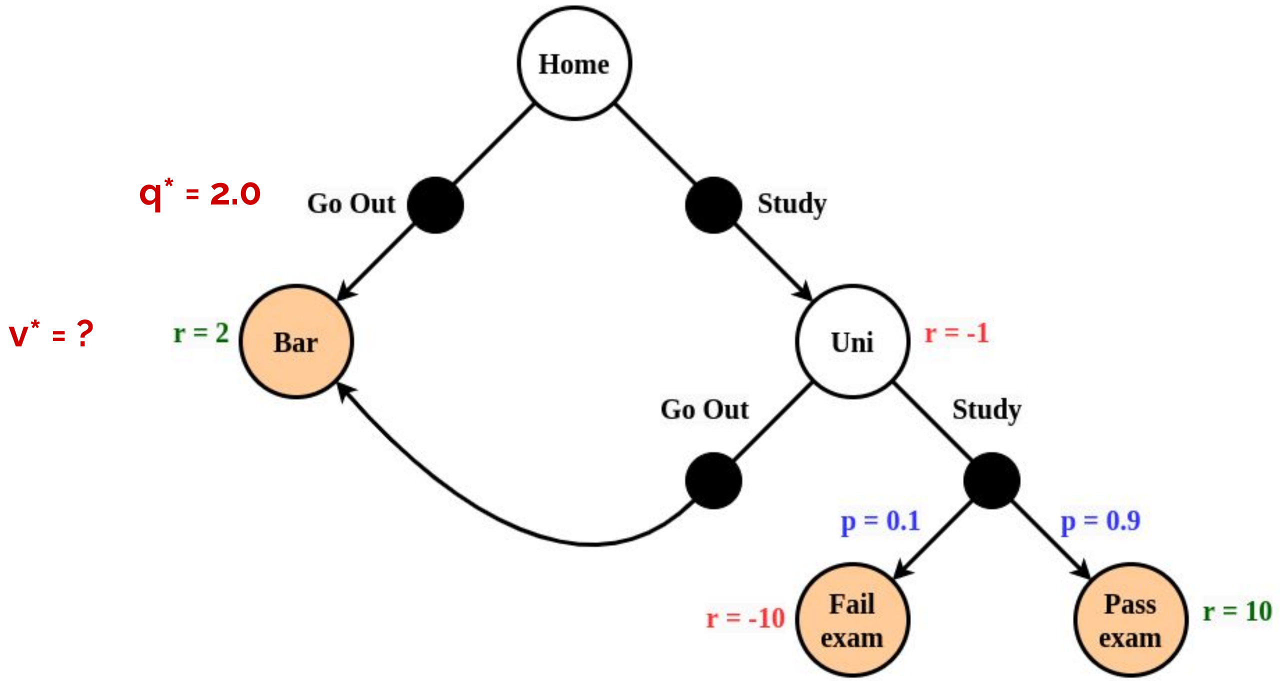
Example: The Study MDP



Question: What is q*(Home, Go Out)?

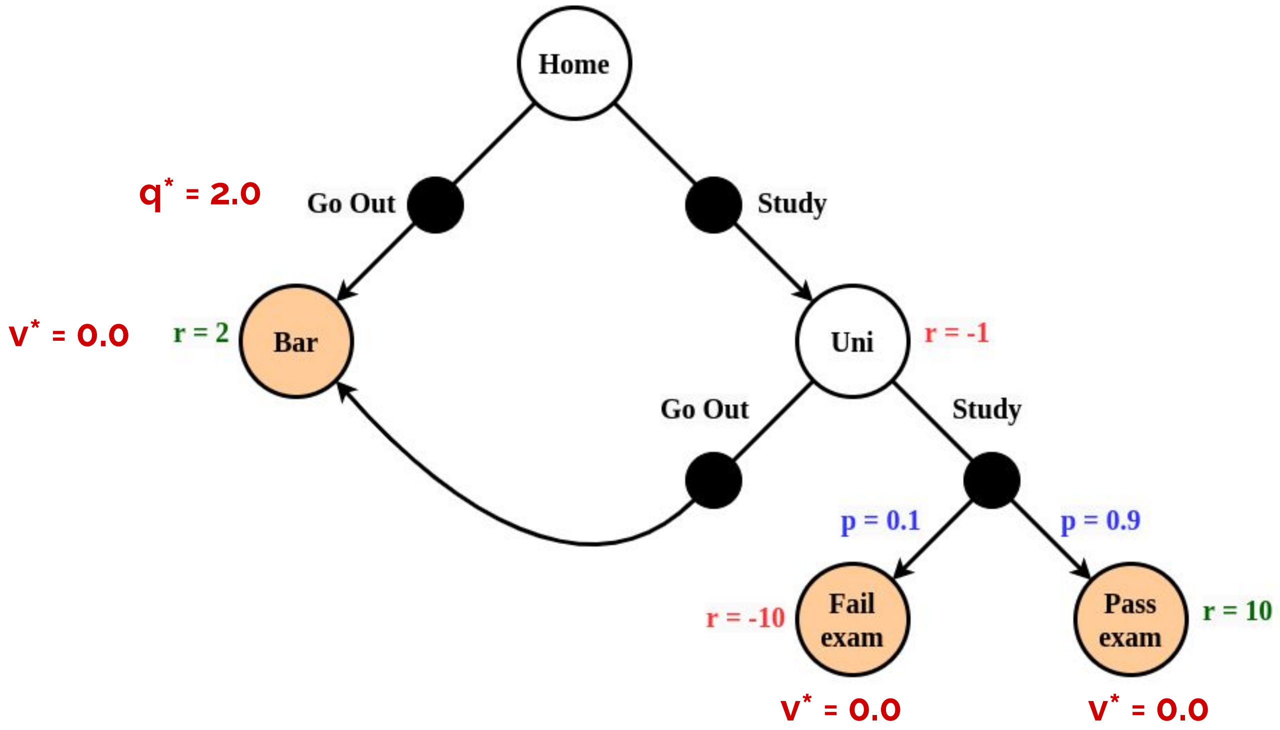
Answer: 2.0
(we always reach the Bar for reward of 2.0, and then terminate)

Example: The Study MDP



Question: What is v^*(Bar)?

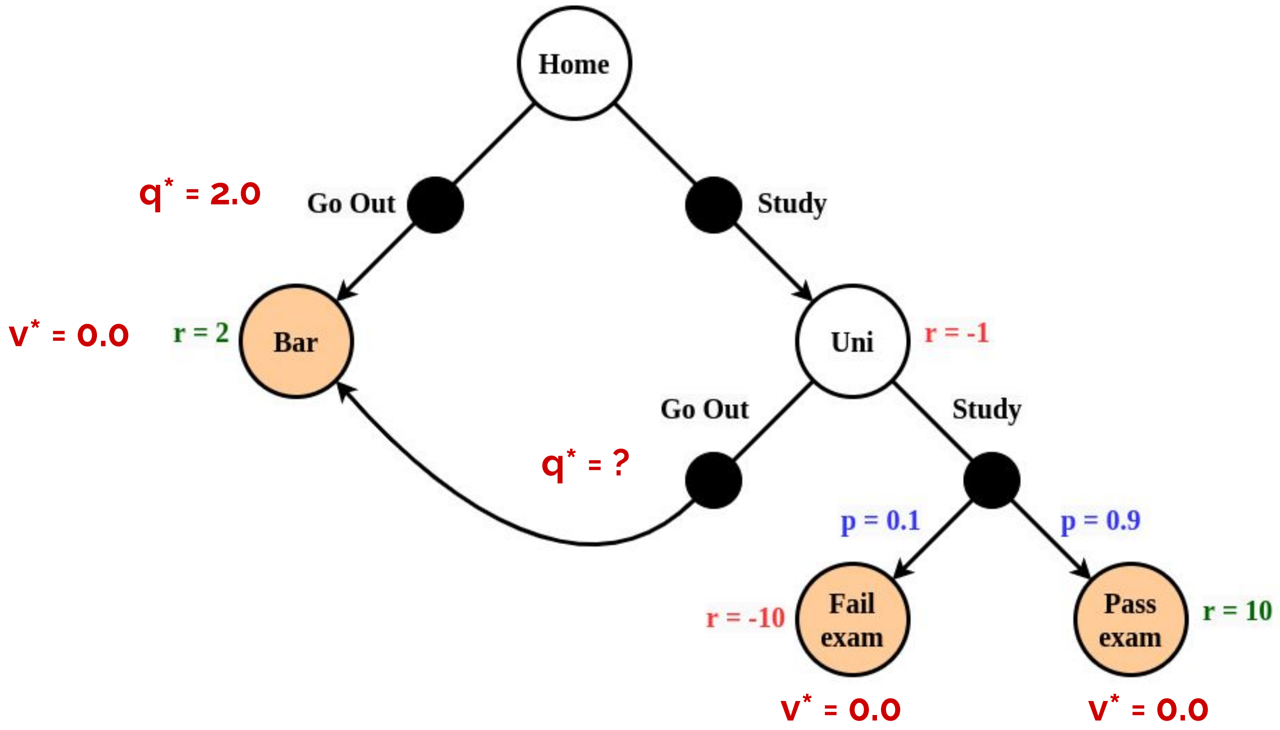
Example: The Study MDP



Question: What is v*(Bar)?

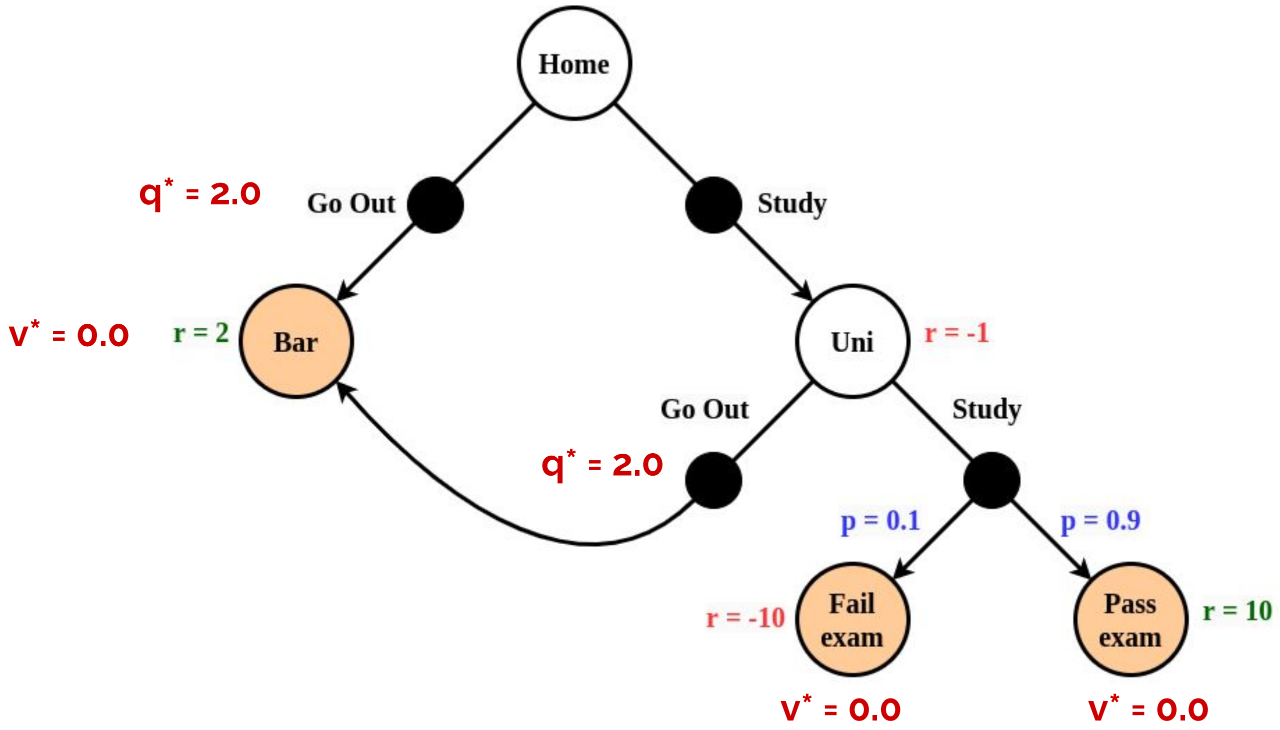
Answer: 0.0
(terminal state so we can never get any additional reward)

Example: The Study MDP



Question: What is q*(Uni, Go Out)?

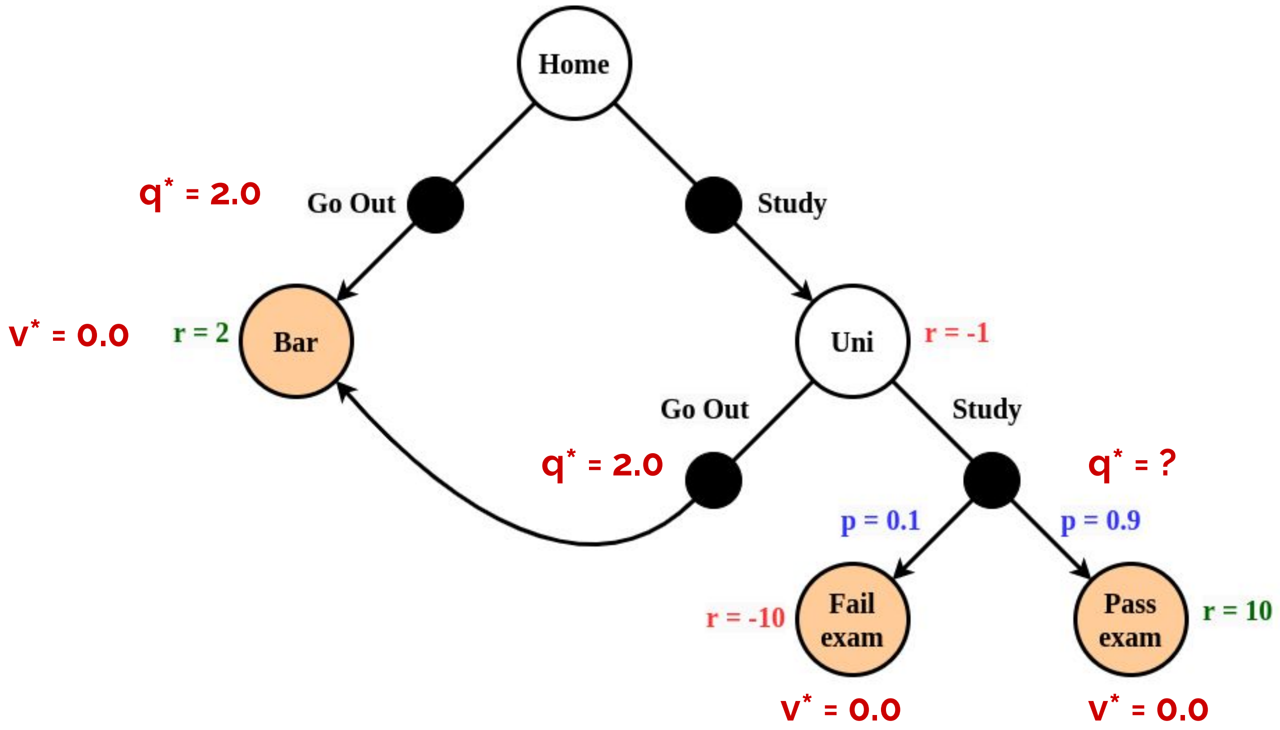
Example: The Study MDP



Question: What is q*(Uni, Go Out)?

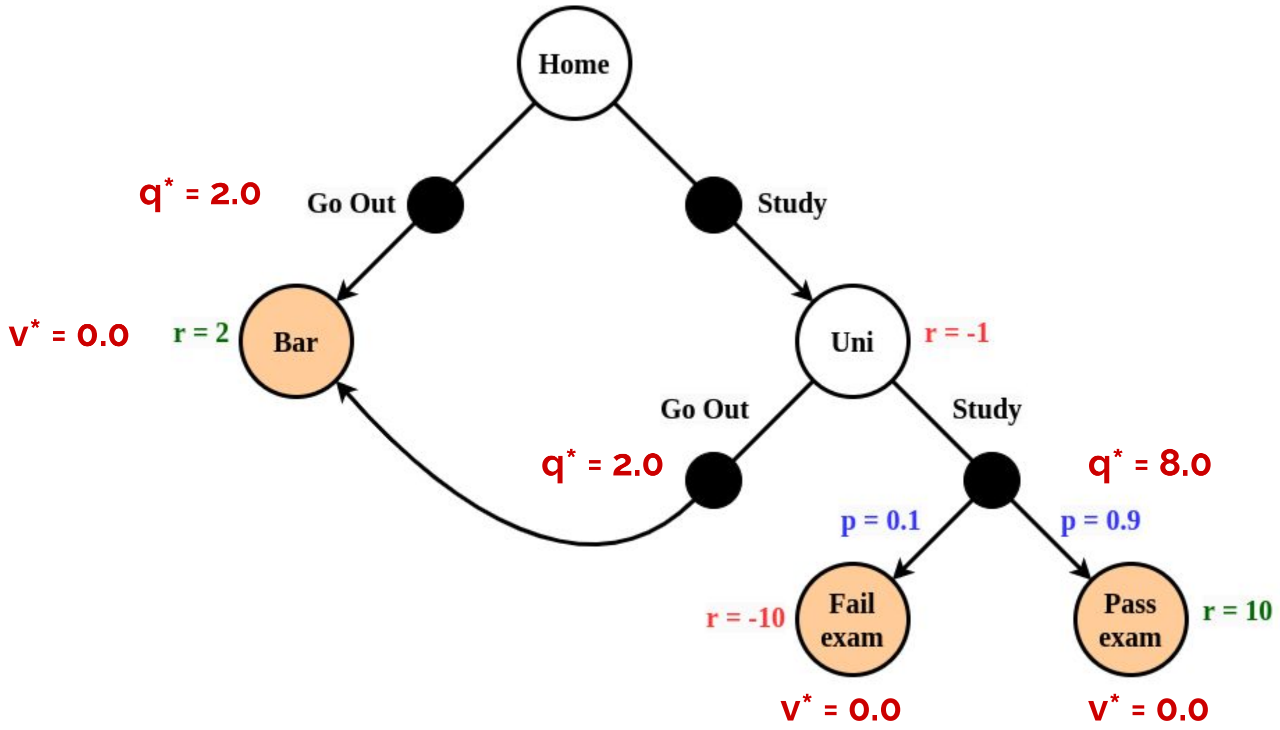
Answer: 2.0
(we always reach the Bar for reward of 2.0, and then terminate)

Example: The Study MDP



Question: What is q^*(Uni, Study)? (Stochastic!)

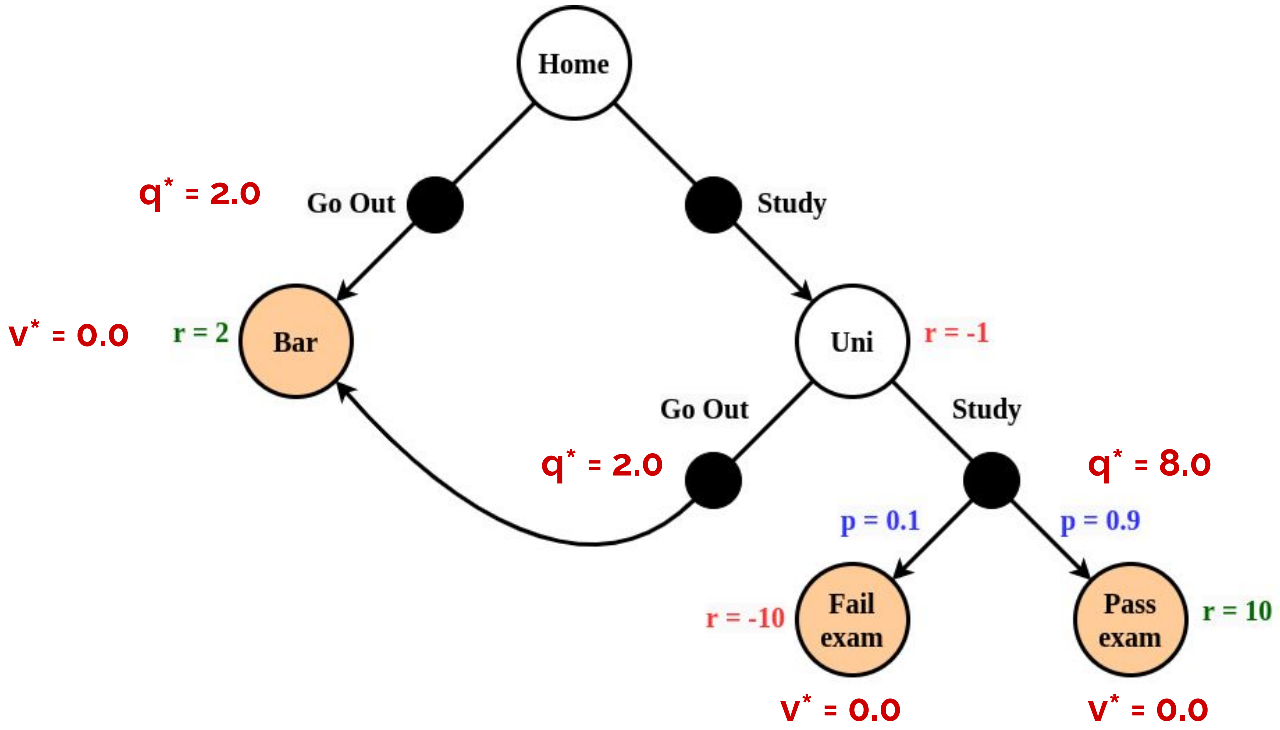
Example: The Study MDP



Question: What is q*(Uni, Study)? (Stochastic!)

Answer: 0.9*10 + 0.1 * (-10) = 8.0
(90% we pass the exam for reward +10, but 10% we fail and get reward -10)

Example: The Study MDP

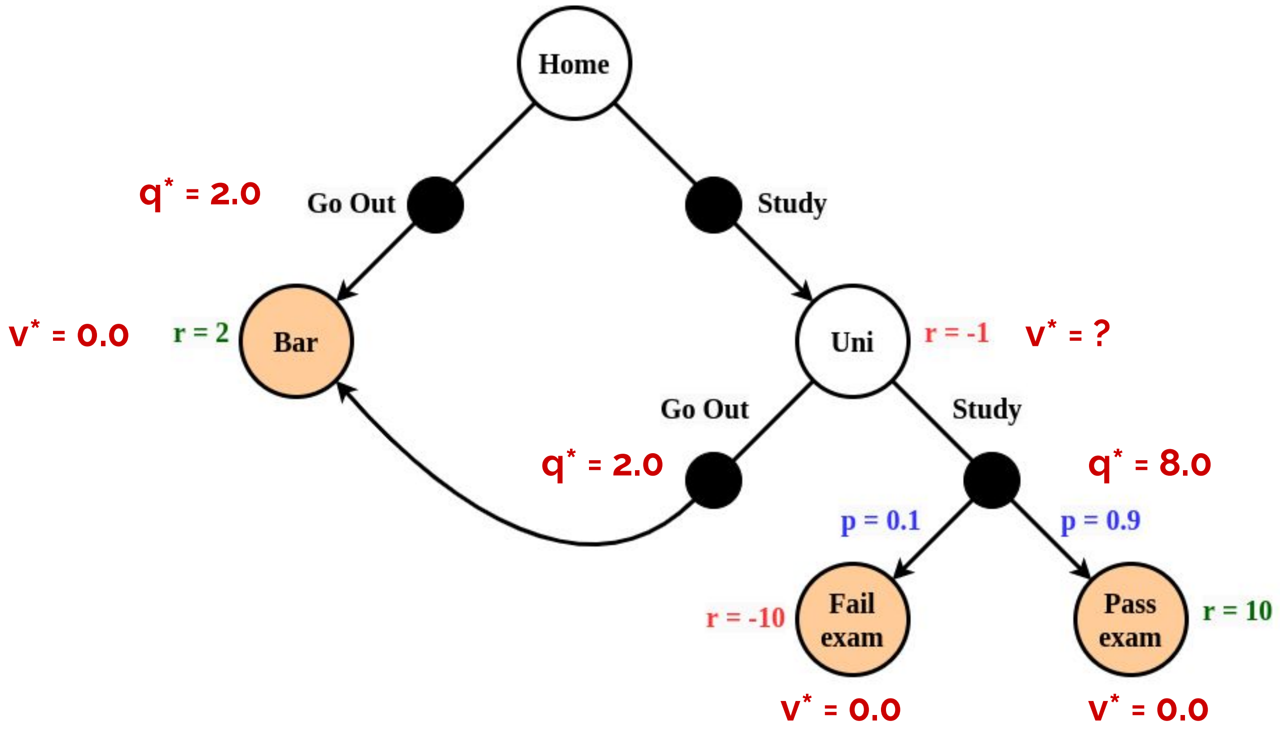


We **average** over the environment dynamics

Question: What is q*(Uni, Study)? (Stochastic!)

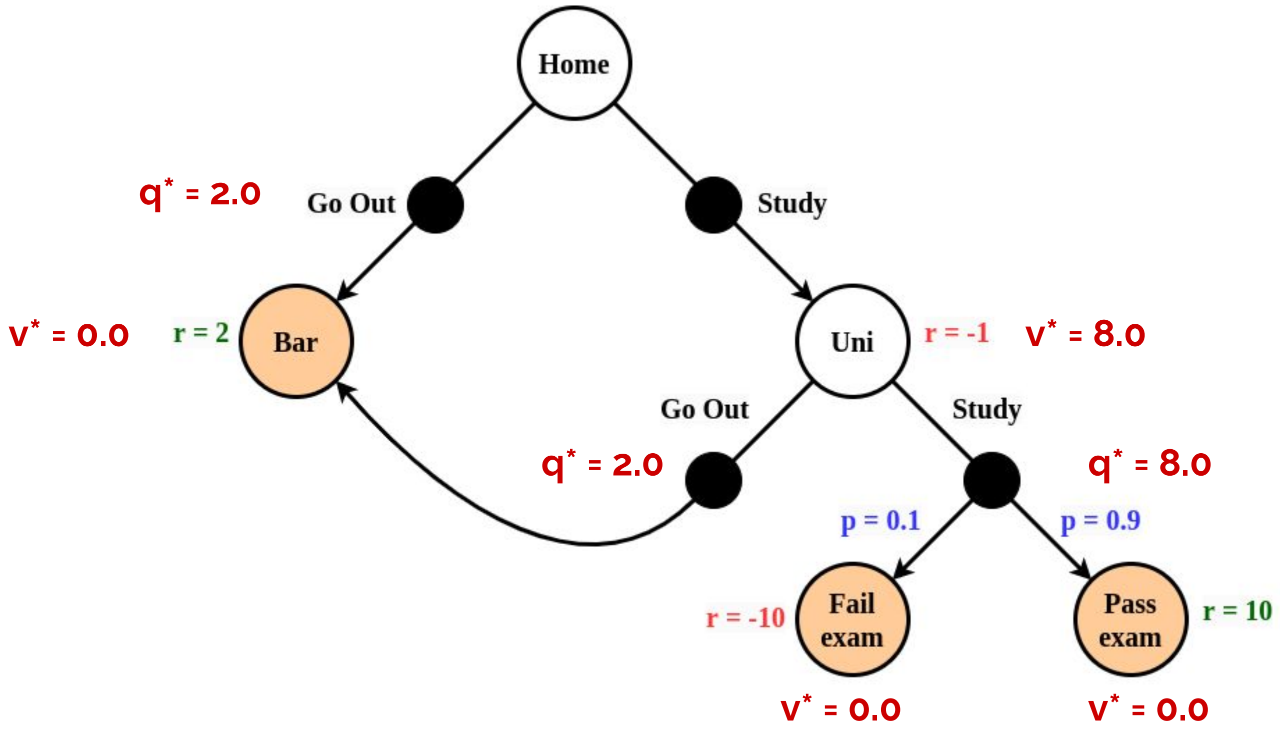
Answer: 0.9*10 + 0.1 * (-10) = 8.0
(90% we pass the exam for reward +10, but 10% we fail and get reward -10)

Example: The Study MDP



Question: What is v*(Uni)?

Example: The Study MDP

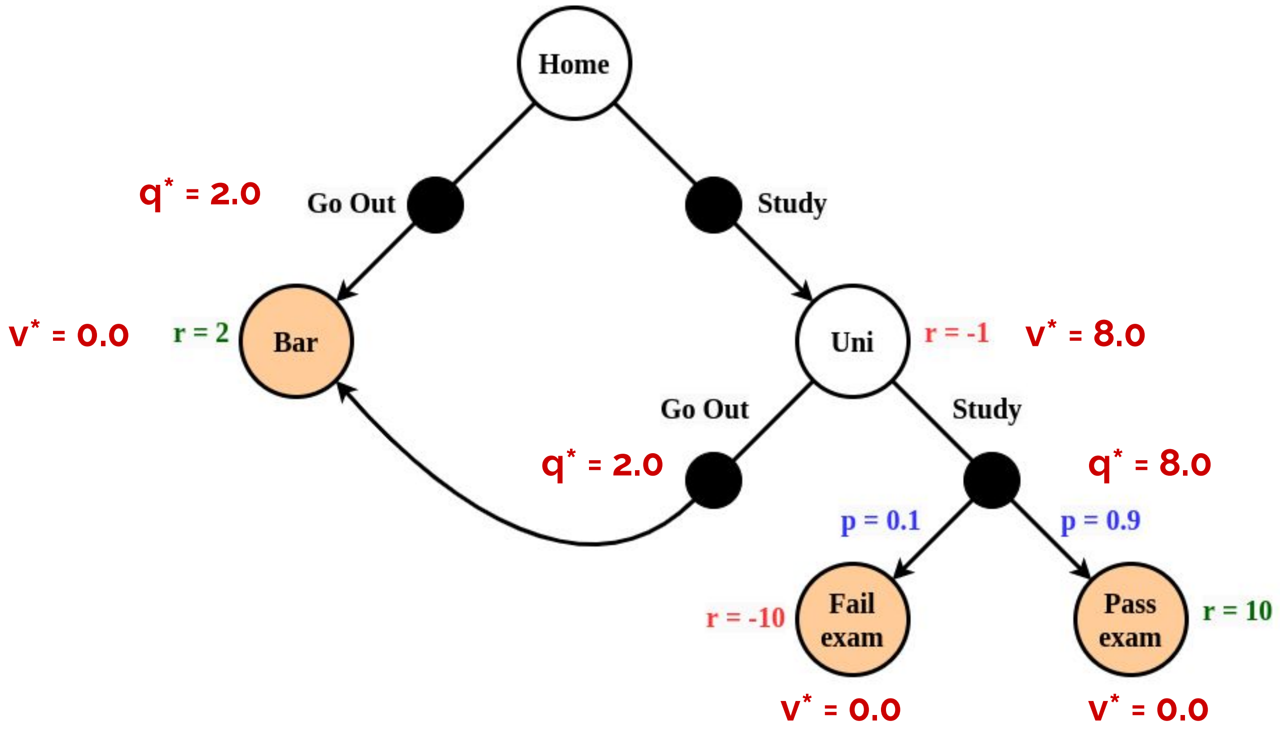


Question: What is v*(Uni)?

Answer: 8.0

(The best choice is to Study from Uni, which we already know has value 8.0)

Example: The Study MDP



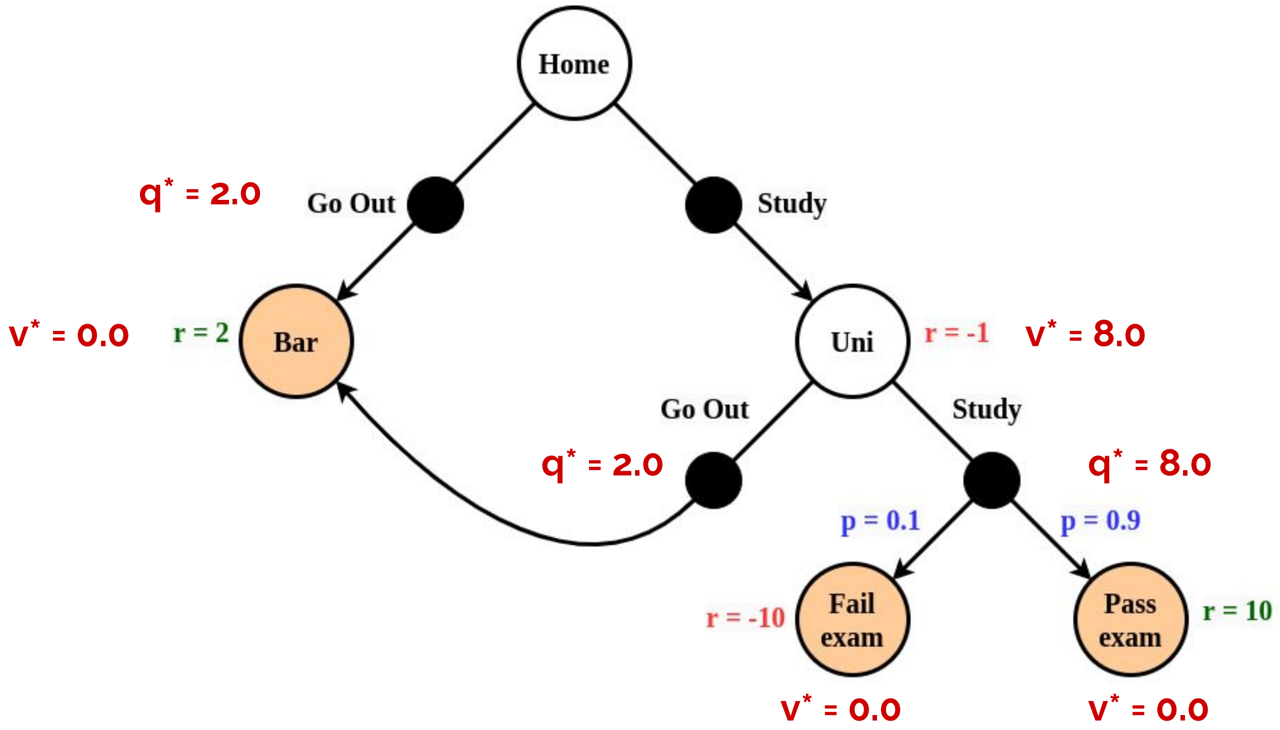
We **maximize** over our own decisions

Question: What is v*(Uni)?

Answer: 8.0

(The best choice is to Study from Uni, which we already know has value 8.0)

Example: The Study MDP



Markov Decision Process=

MAX

EXP

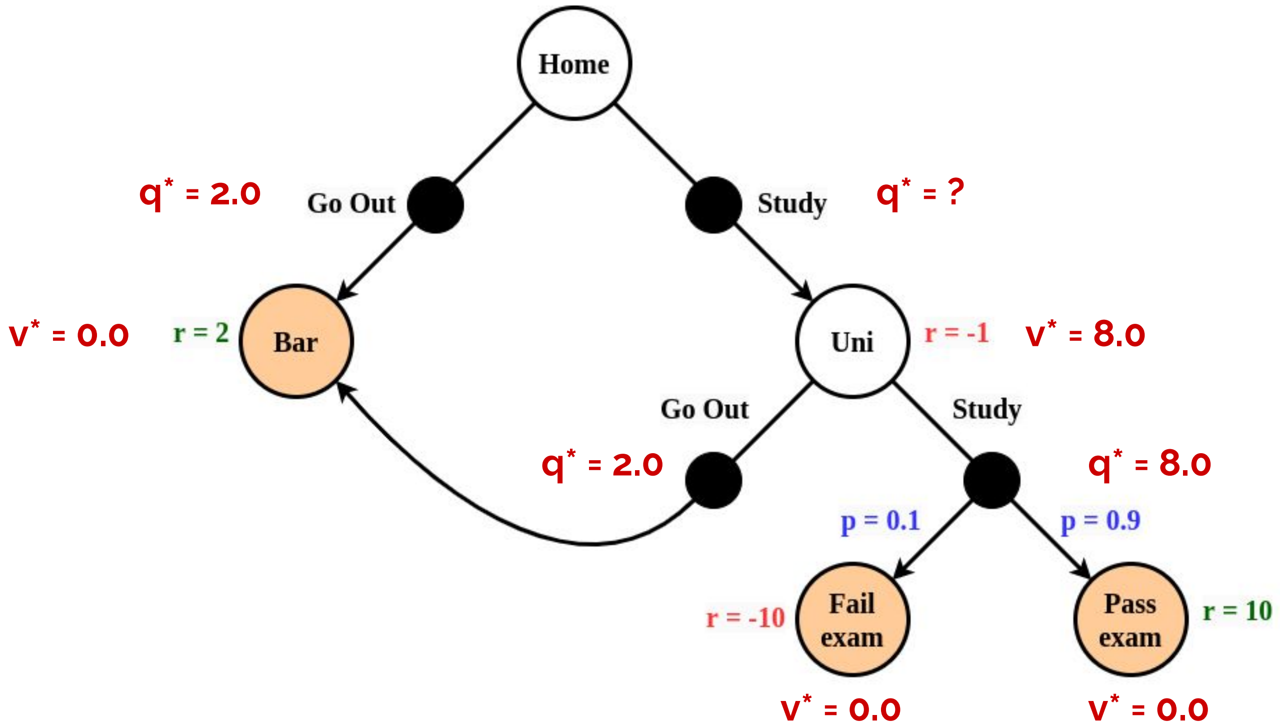
graph

Question: What is v*(Uni)?

Answer: 8.0

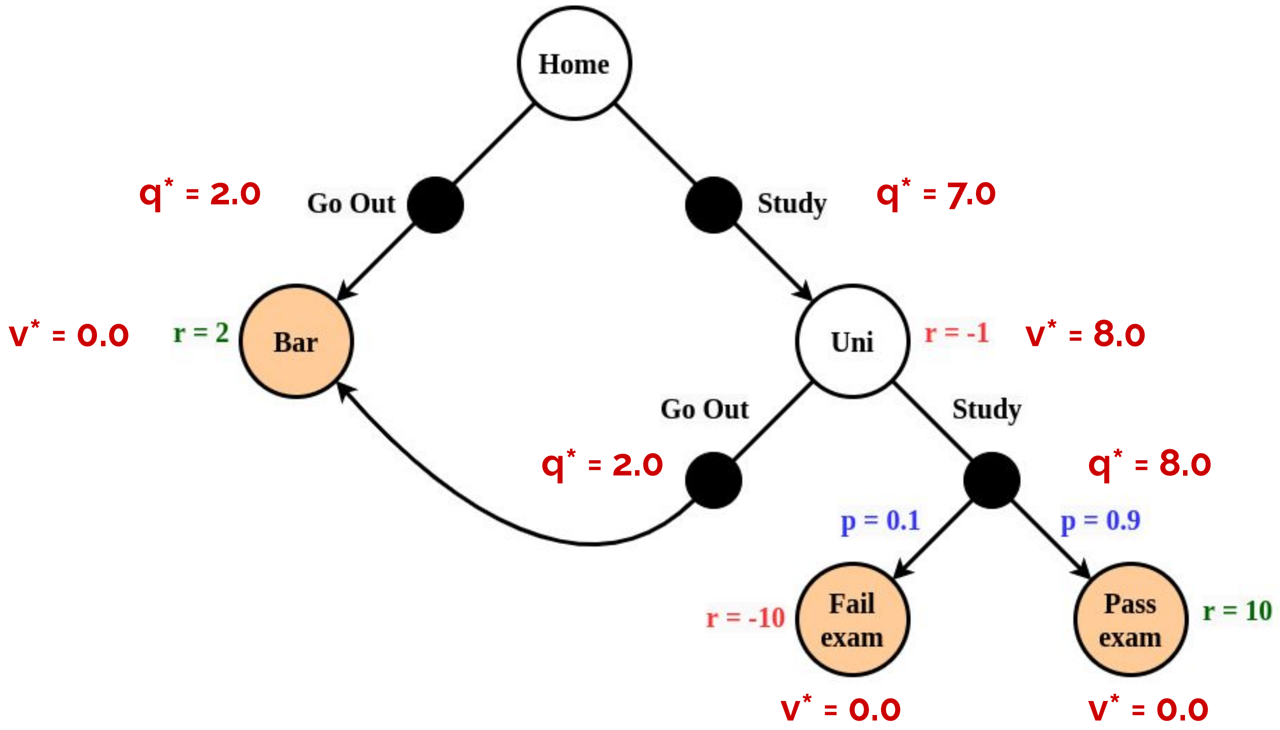
(The best choice is to Study from Uni, which we already know has value 8.0)

Example: The Study MDP



Question: What is q*(Home,Study)?

Example: The Study MDP

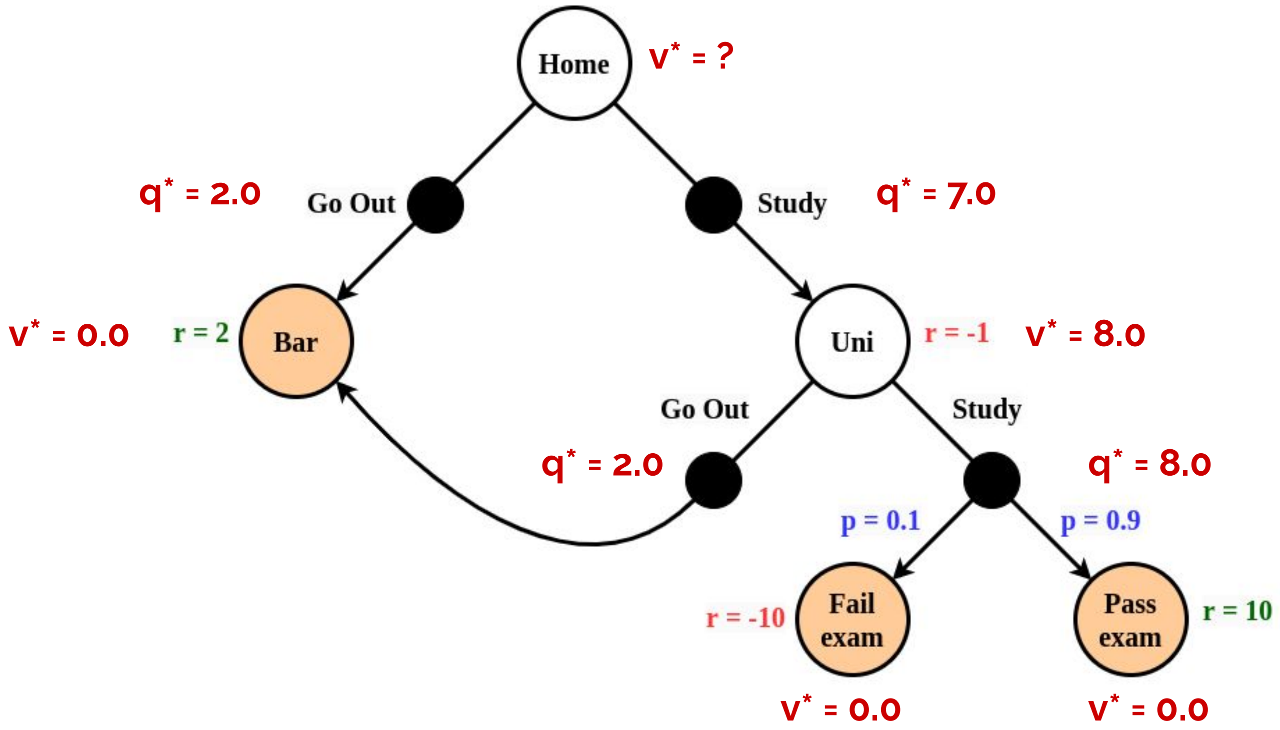


Question: What is q*(Home,Study)?

Answer: -1.0 + 8.0 = 7.0

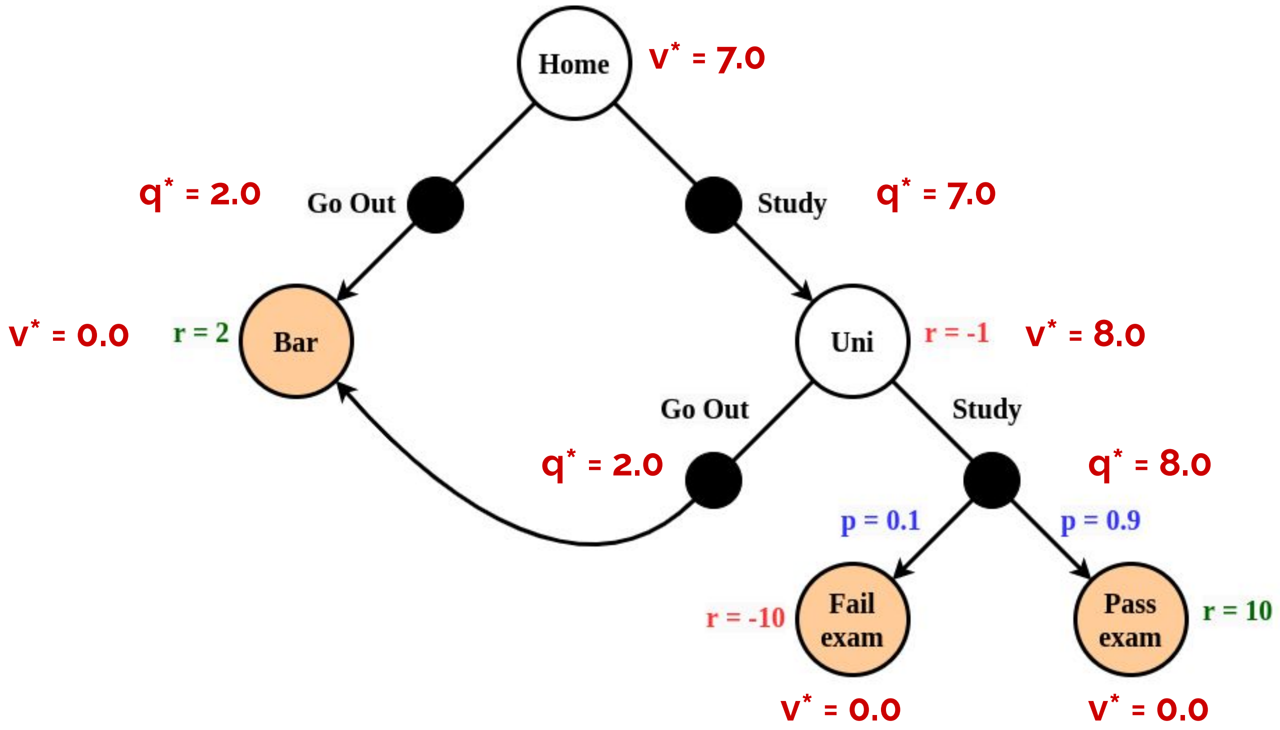
(We get -1.0 for reaching the Uni, and can then at best get 8.0 afterwards)

Example: The Study MDP



Question: What is v*(Home)?

Example: The Study MDP

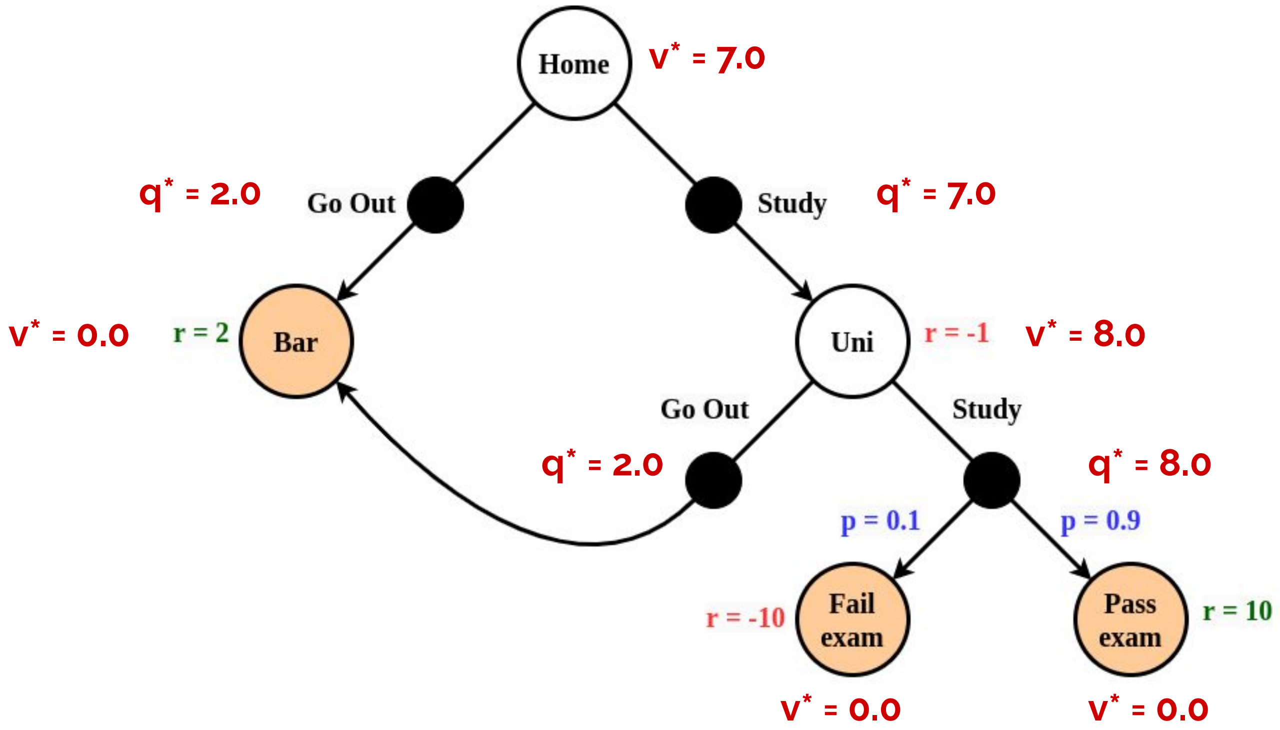


Question: What is v*(Home)?

Answer: 7.0

(We can choose to Study, which will give an average total reward of 7.0)

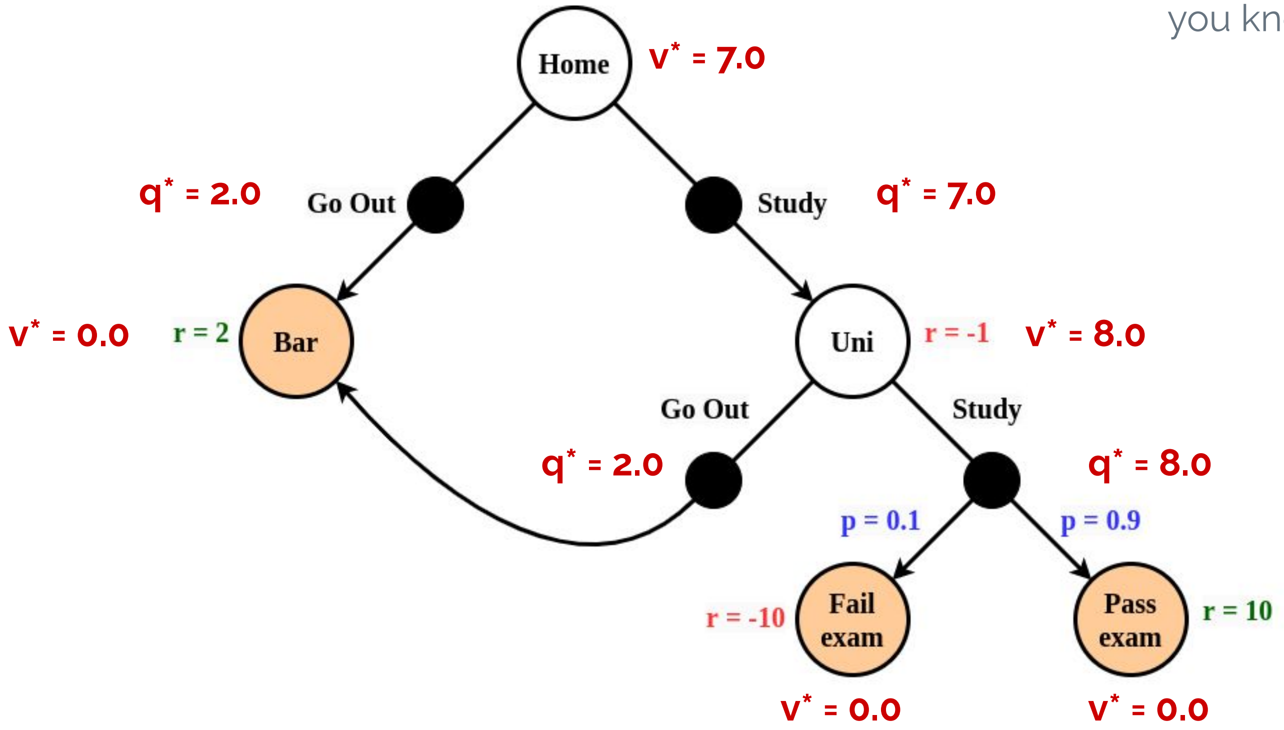
Example: The Study MDP



Question: So what should you do at Home?

Example: The Study MDP

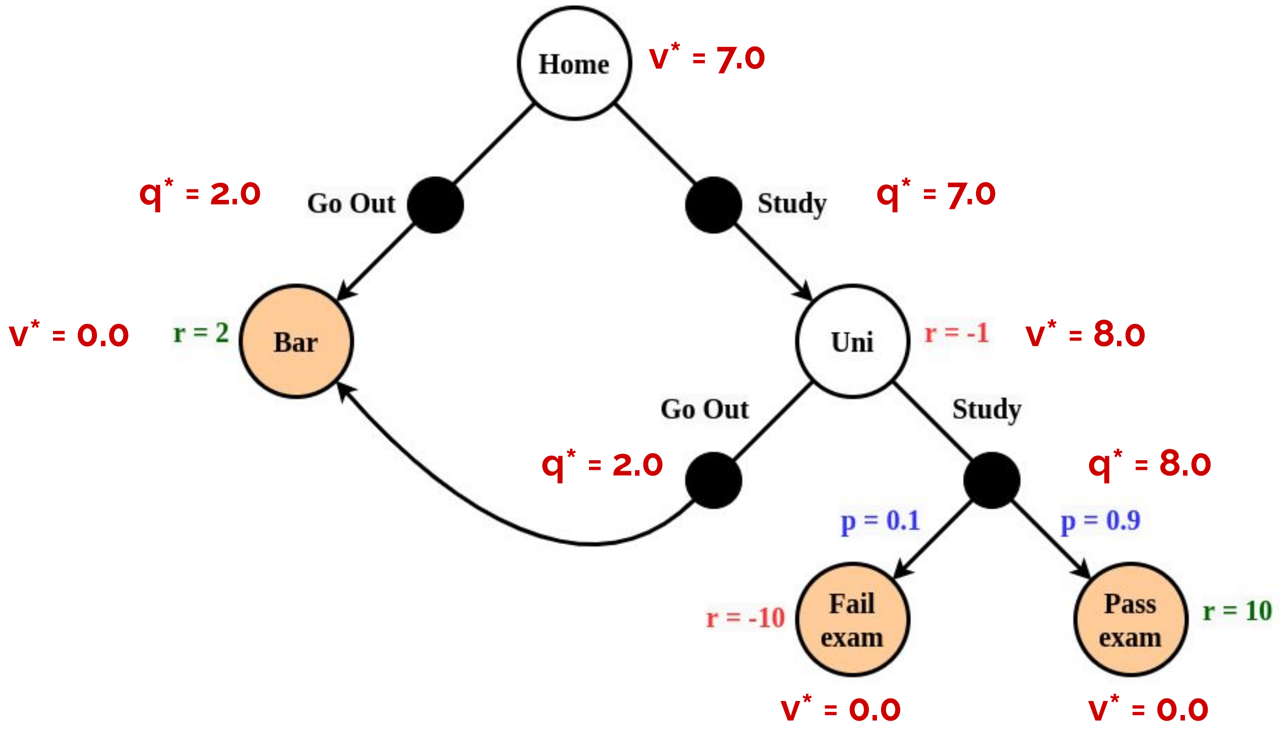
Given the optimal q* values you know how to act



Question: So what should you do at Home?

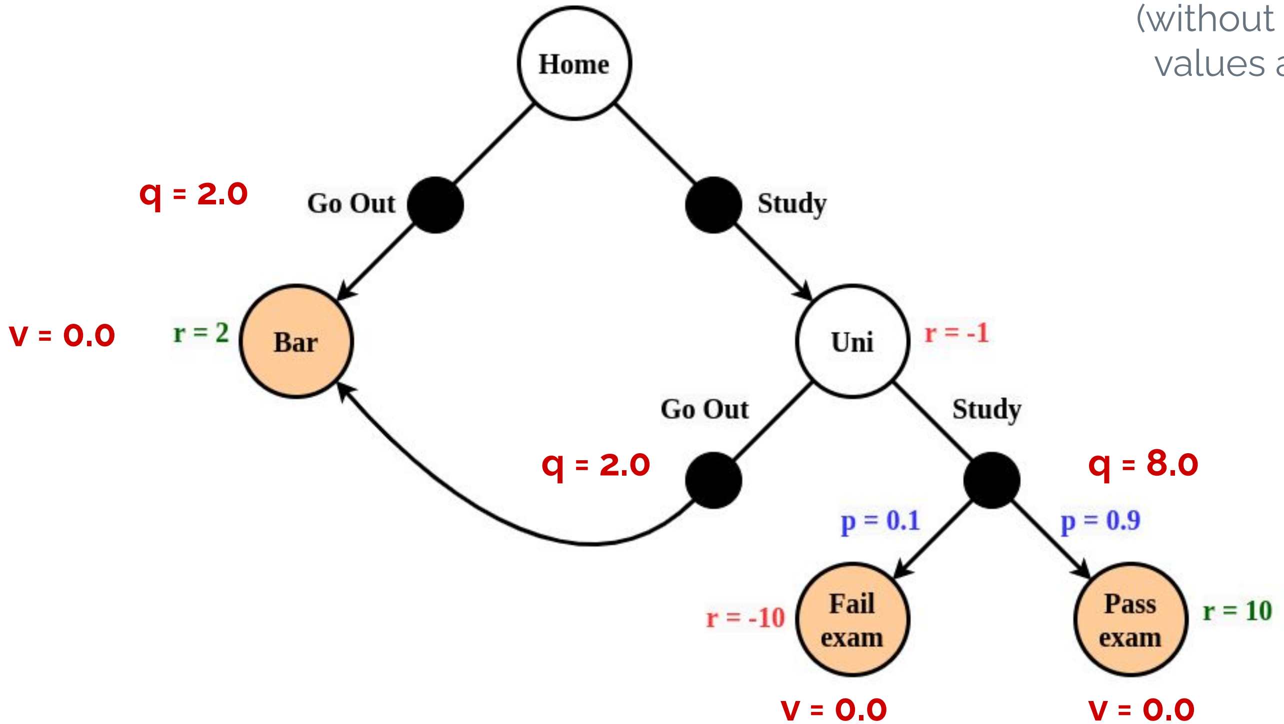
Answer: Come to university!

Example: The Study MDP



Question: Imagine we act *randomly* instead of *optimally*. Which values will stay the same?

Example: The Study MDP

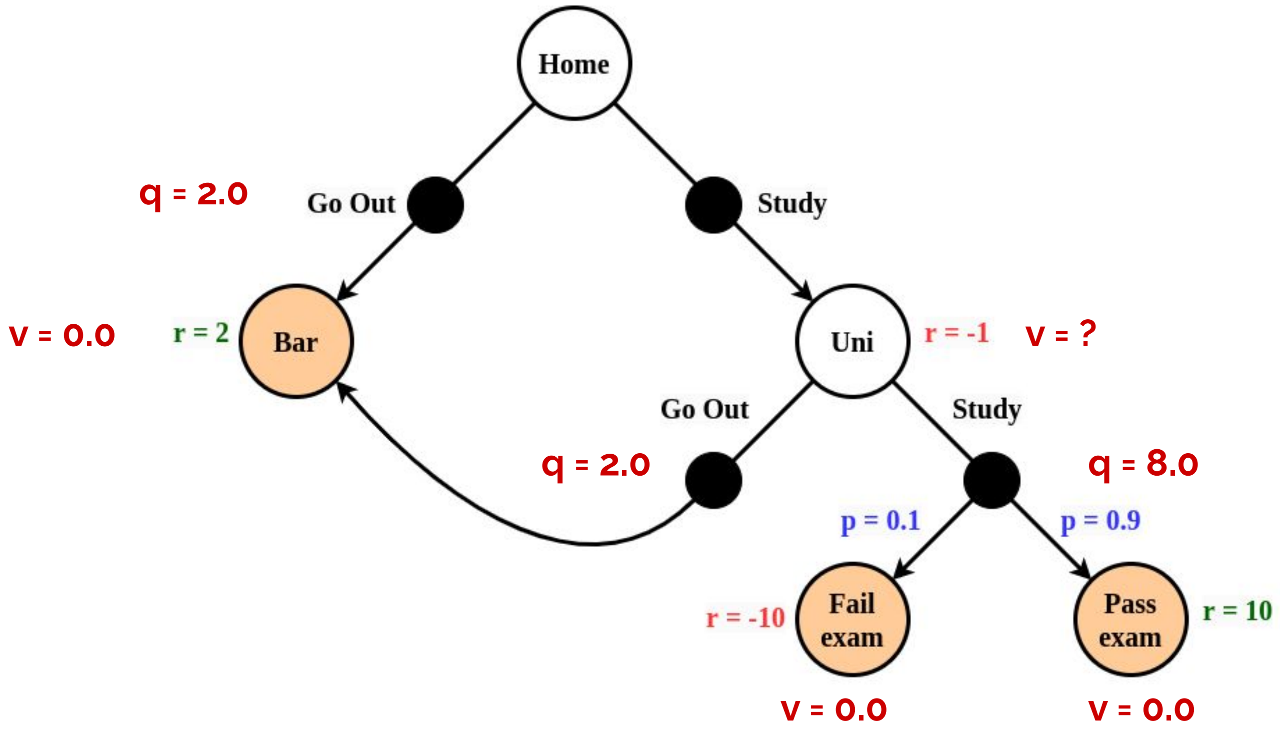


We now write v and q (without a star) since the values are not optimal

Question: Imagine we act *randomly* instead of *optimally*. Which values will stay the same?

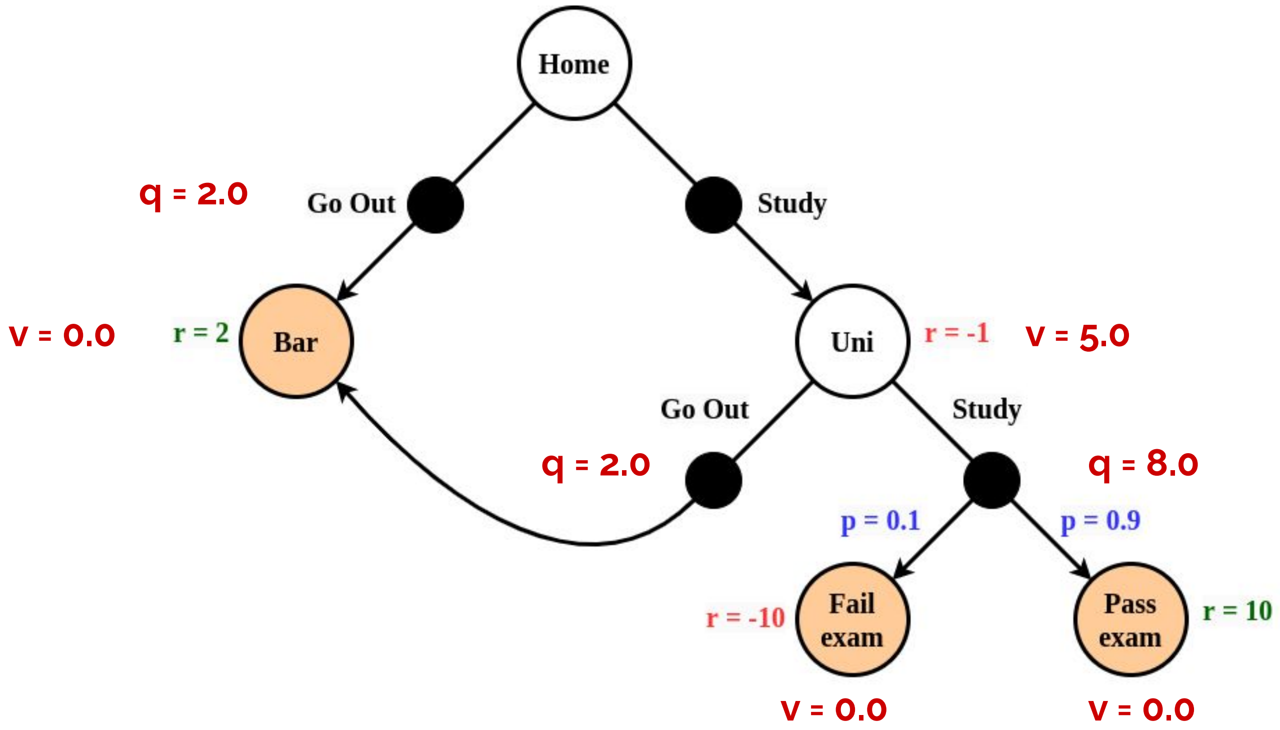
Answer: Terminal states and actions leading to terminal states - don't depend on our policy.

Example: The Study MDP



Question: But what is v(Uni) under the *random* policy?

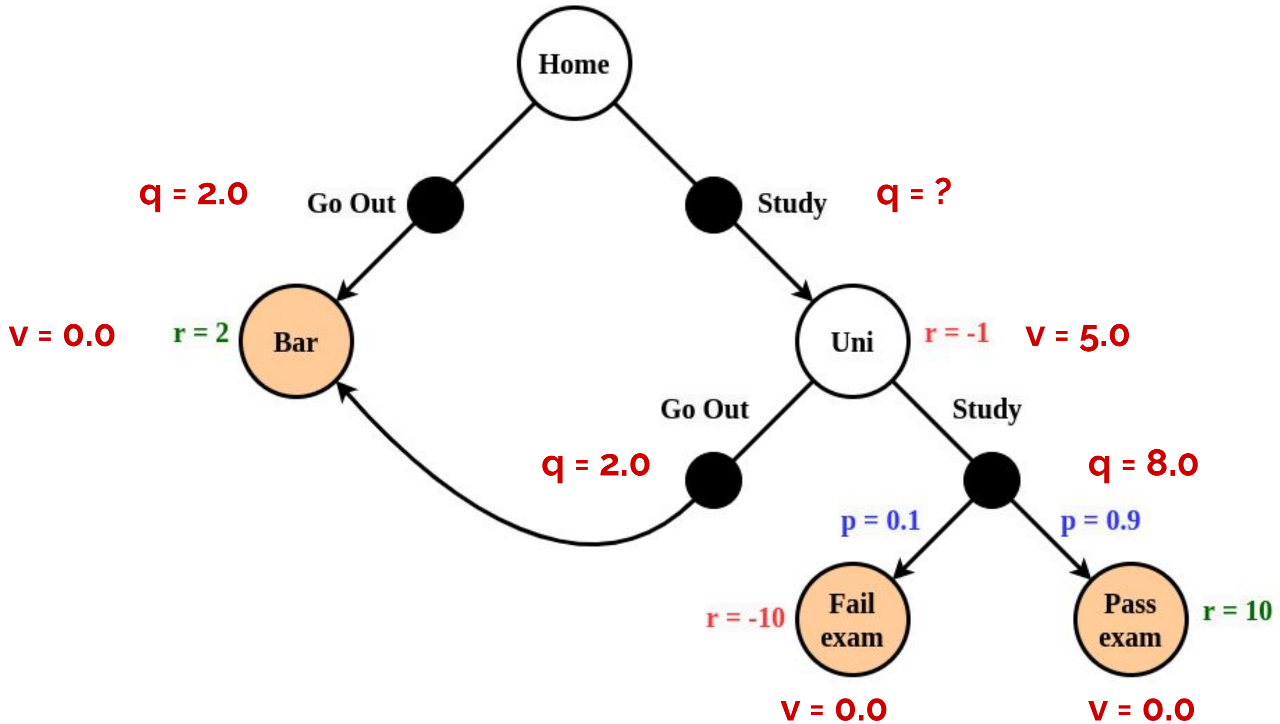
Example: The Study MDP



Question: But what is v(Uni) under the *random* policy?

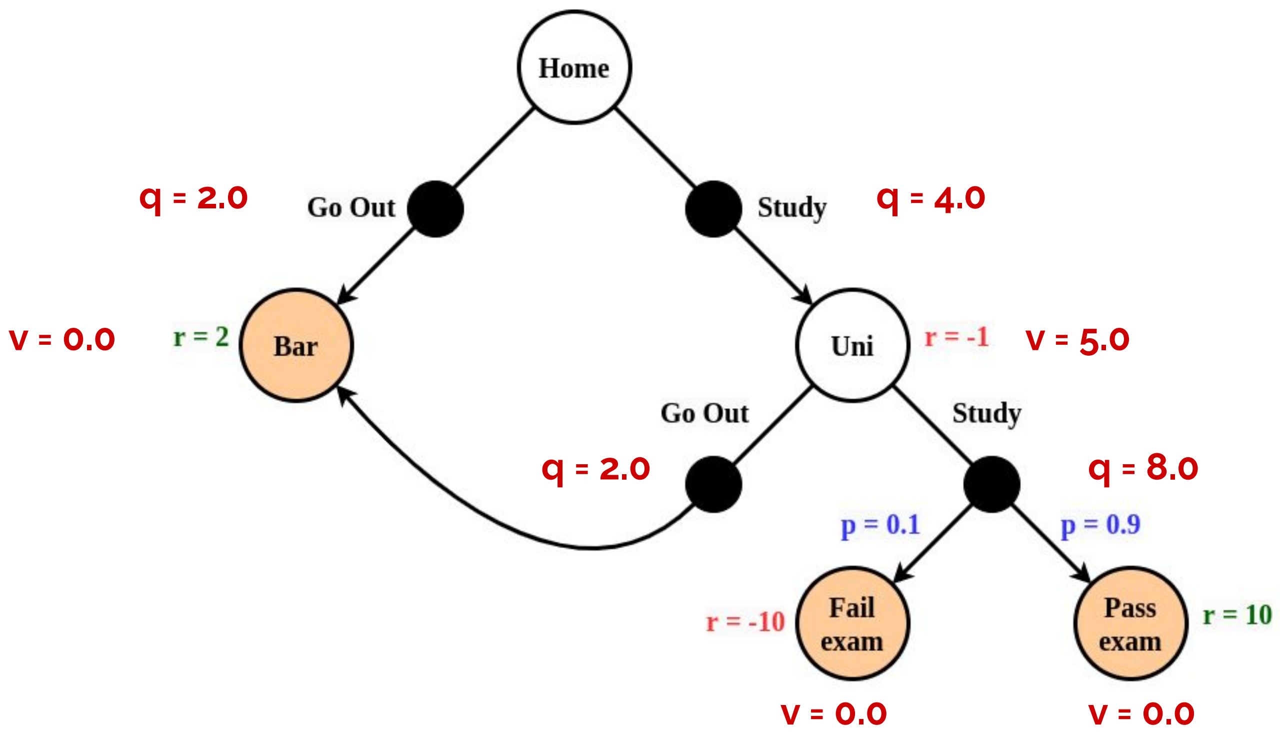
Answer: 5.0
(Act random, so 50% of times Go Out for q=2.0, and 50% Study for q=8.0)

Example: The Study MDP



Question: And q(Home, Study) under the *random* policy?

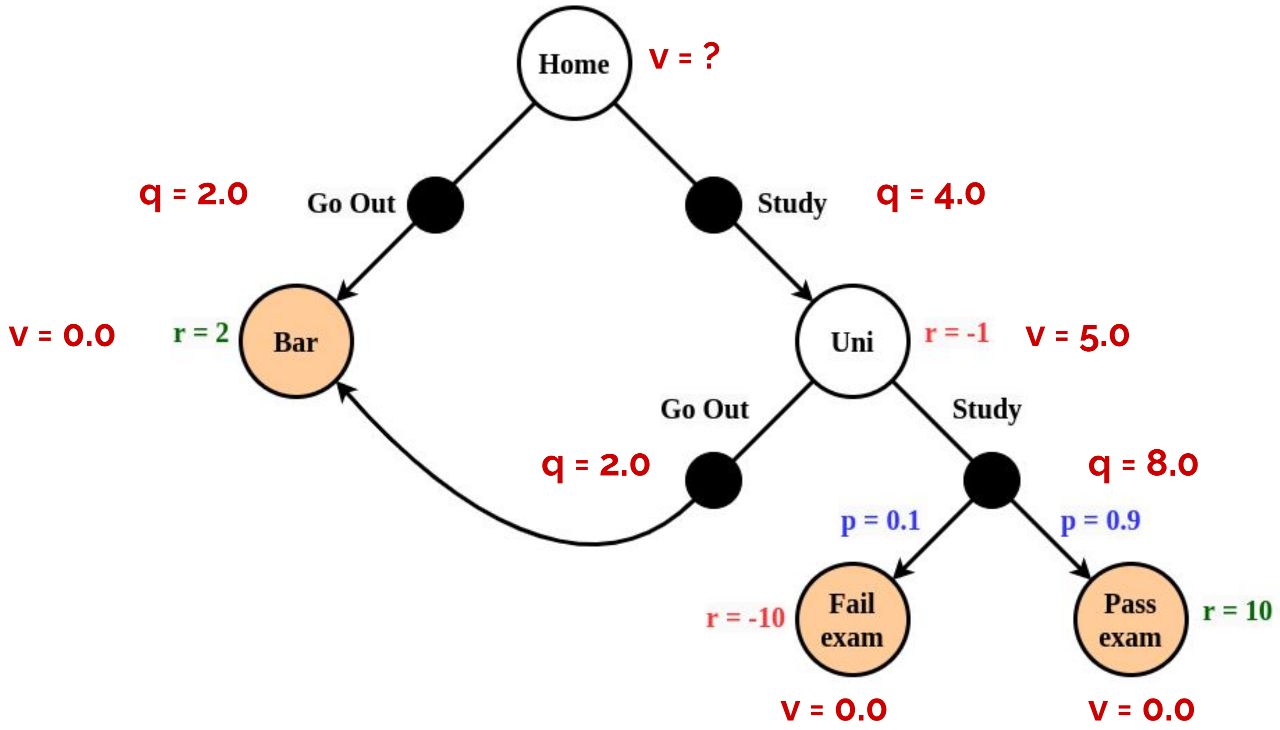
Example: The Study MDP



Question: And q(Home, Study) under the *random* policy?

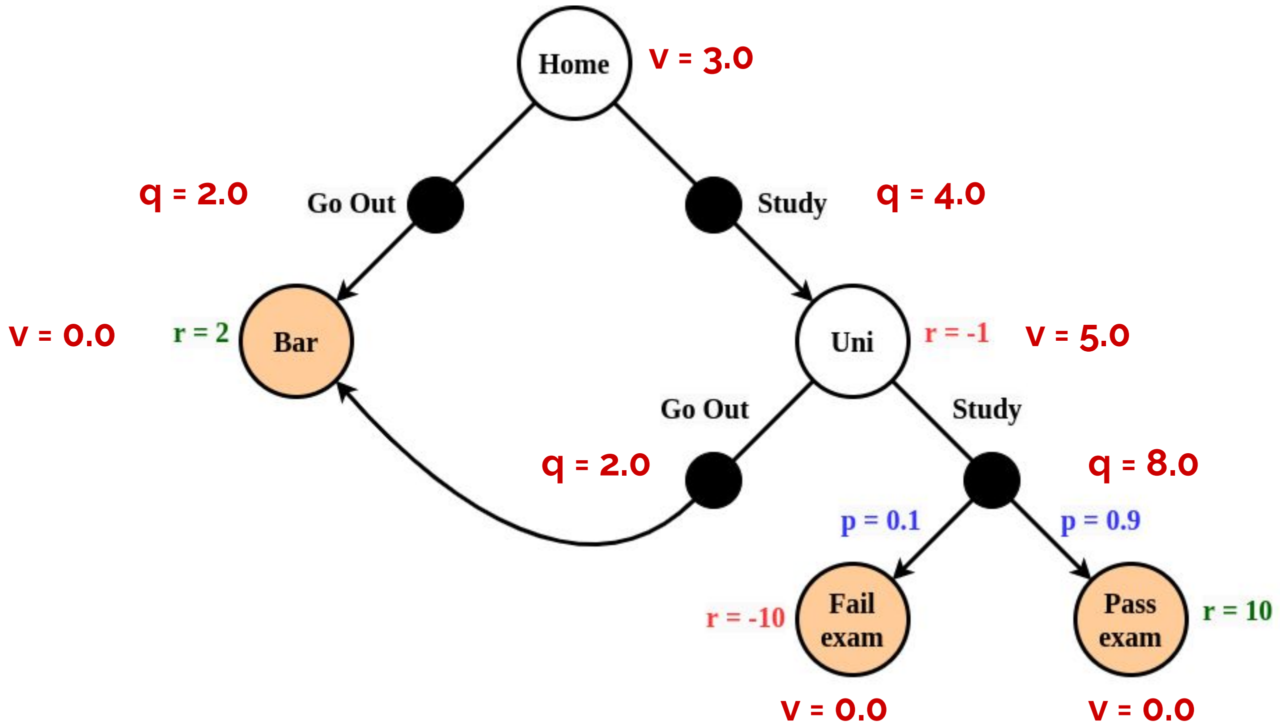
Answer: 4.0
(get -1.0 for going to Uni, and then 5.0 from there)

Example: The Study MDP



Question: So what is v(Home) under the *random* policy?

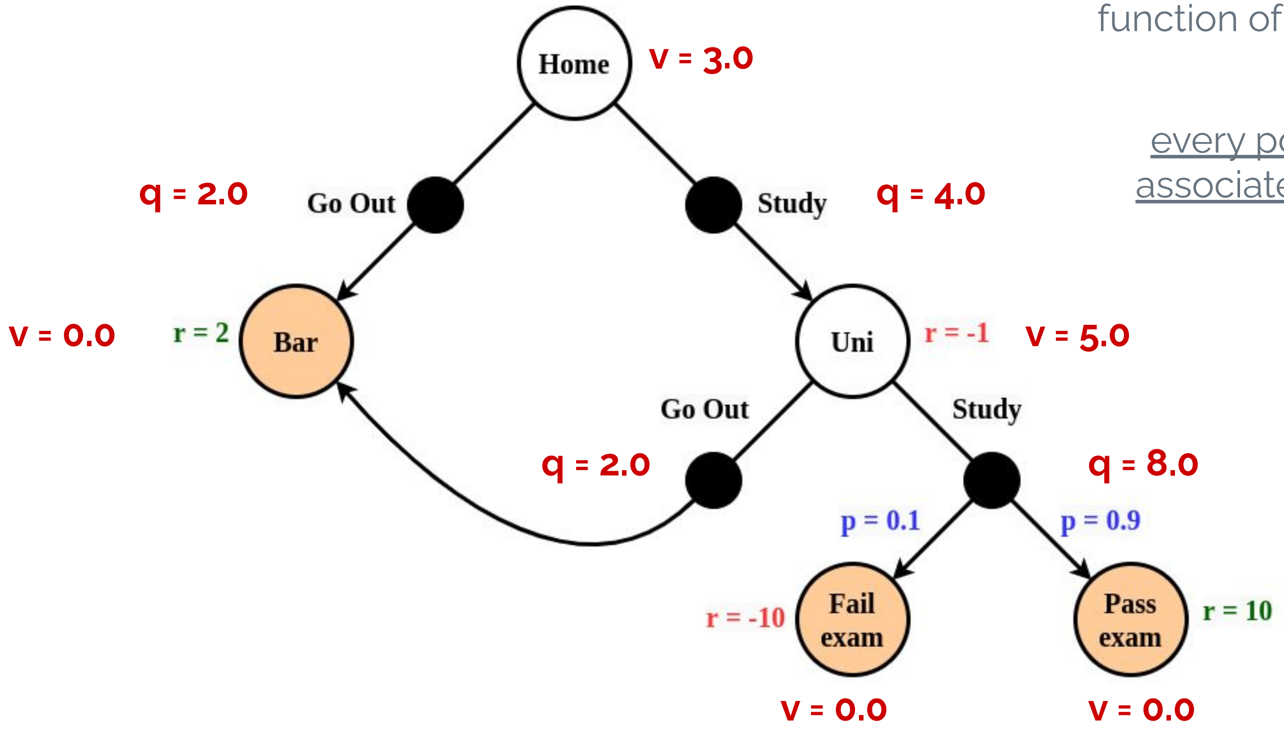
Example: The Study MDP



Question: So what is v(Home) under the *random* policy?

Answer: 3.0
(50% of times Go Out for 2.0, 50% of times Study for 4.0)

Example: The Study MDP



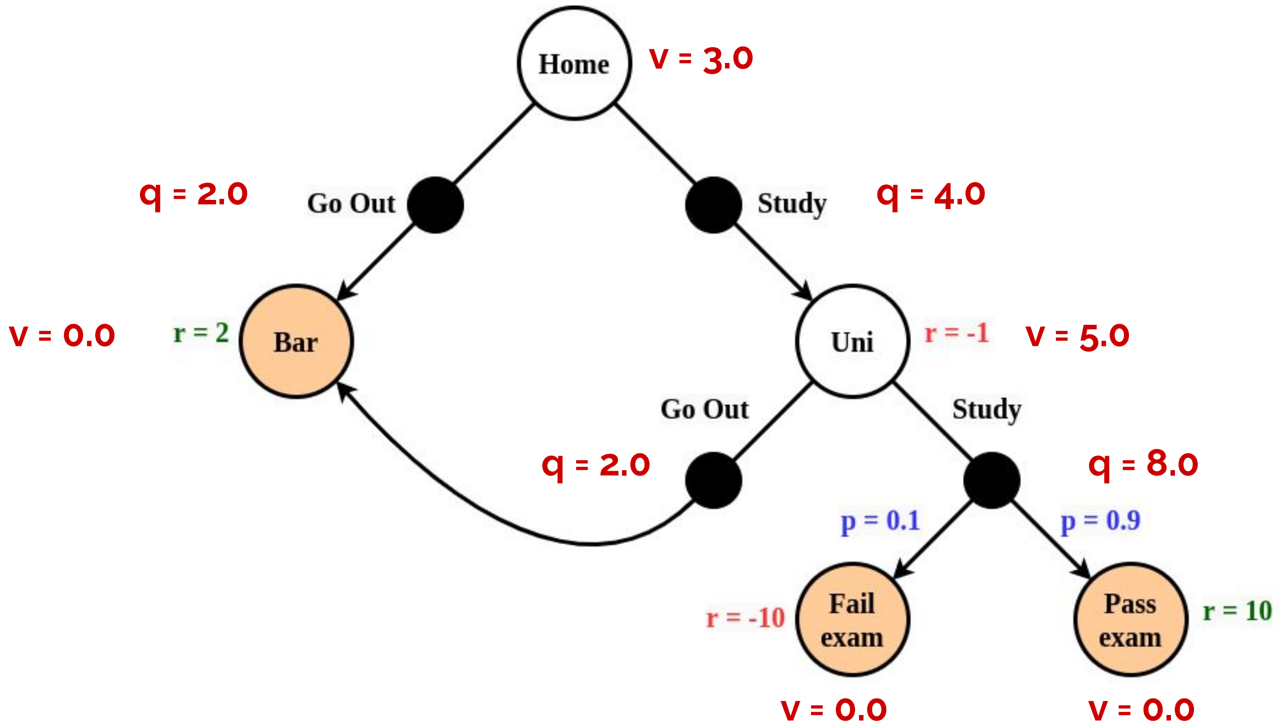
We now computed the value function of the random policy:

<u>every policy has its own associated value function</u>

Question: So what is v(Home) under the *random* policy?

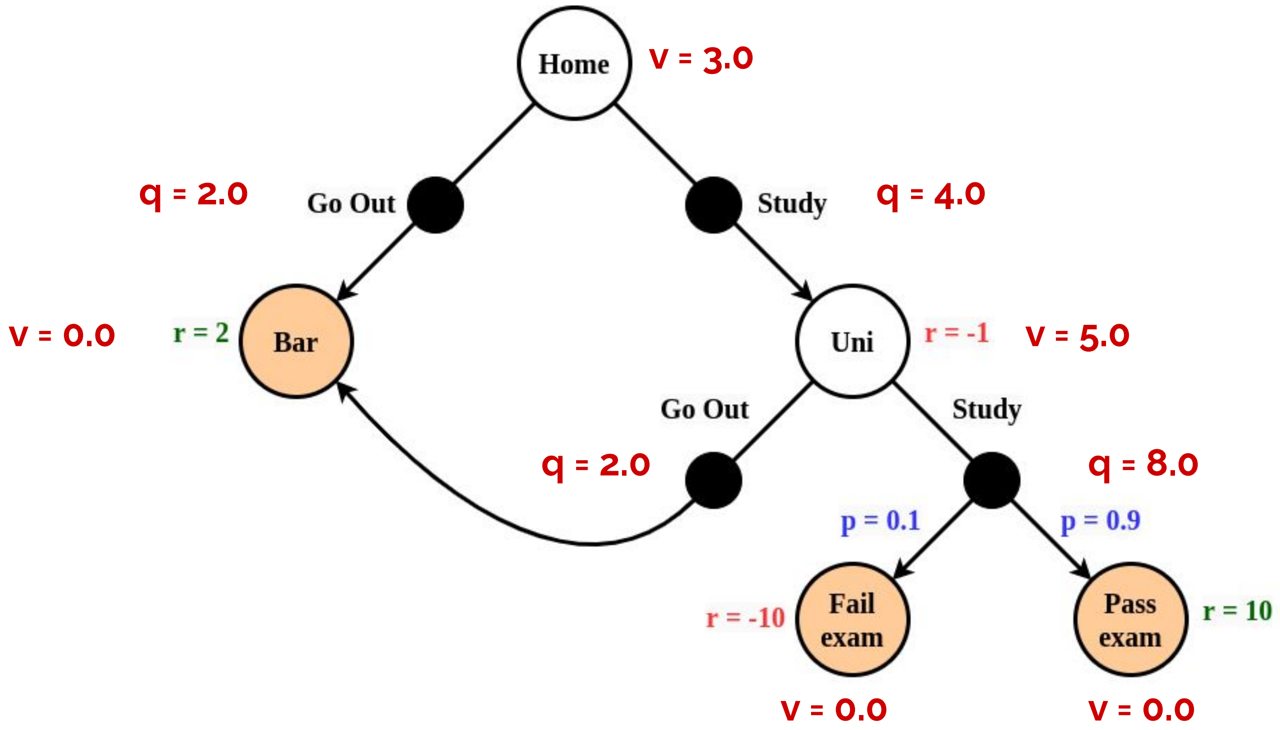
Answer: 3.0
(50% of times Go Out for 2.0, 50% of times Study for 4.0)

Example: The Study MDP



Question: Is it smart to act randomly from Home?

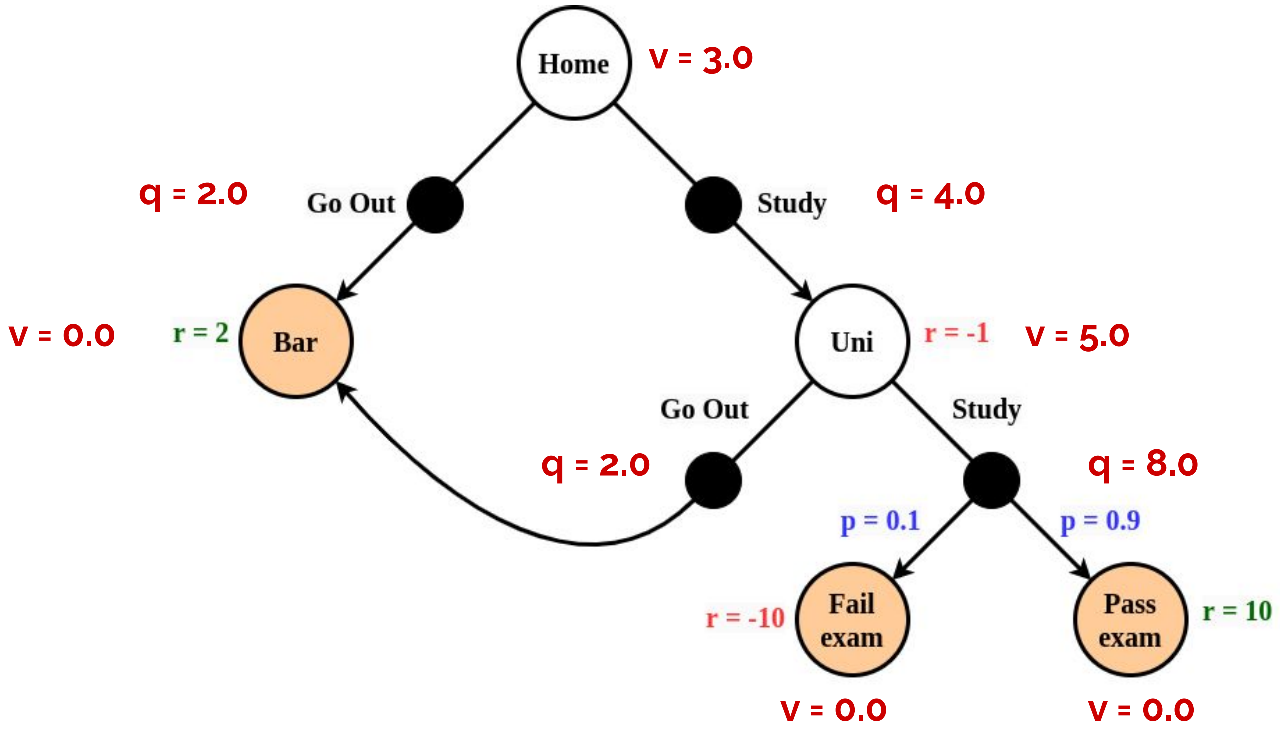
Example: The Study MDP



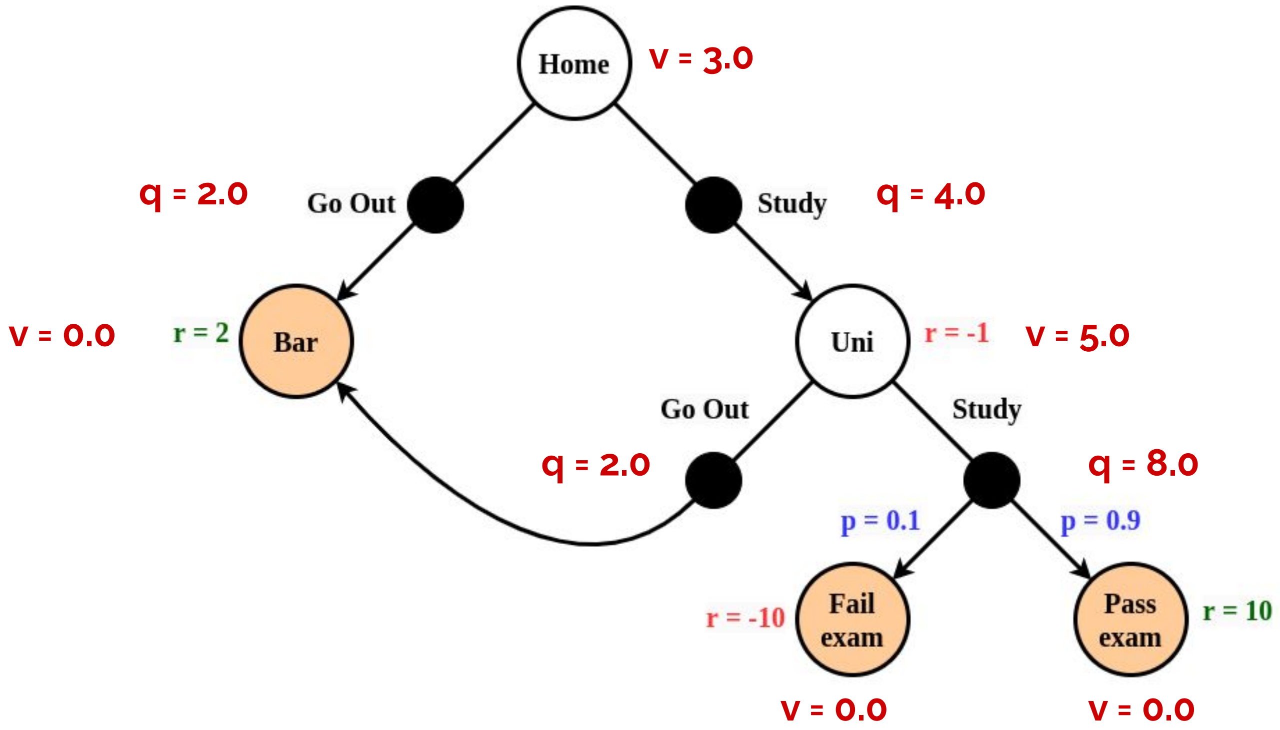
Question: Is it smart to act randomly from Home?

Answer: No!
We could optimally get 7.0 on average from Home, but random policy gives just 3.0!

Example: The Study MDP

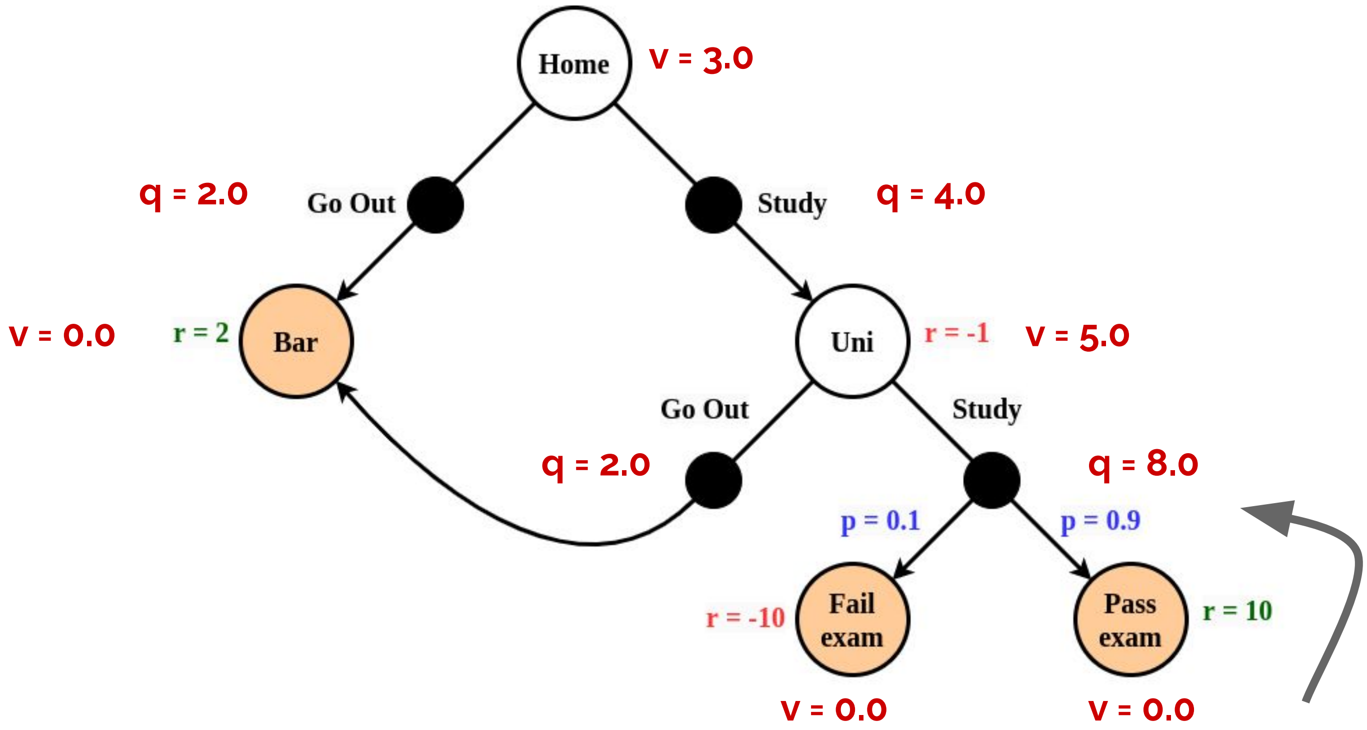


Example: The Study MDP



There is a recursive relation between the value estimates of states and actions

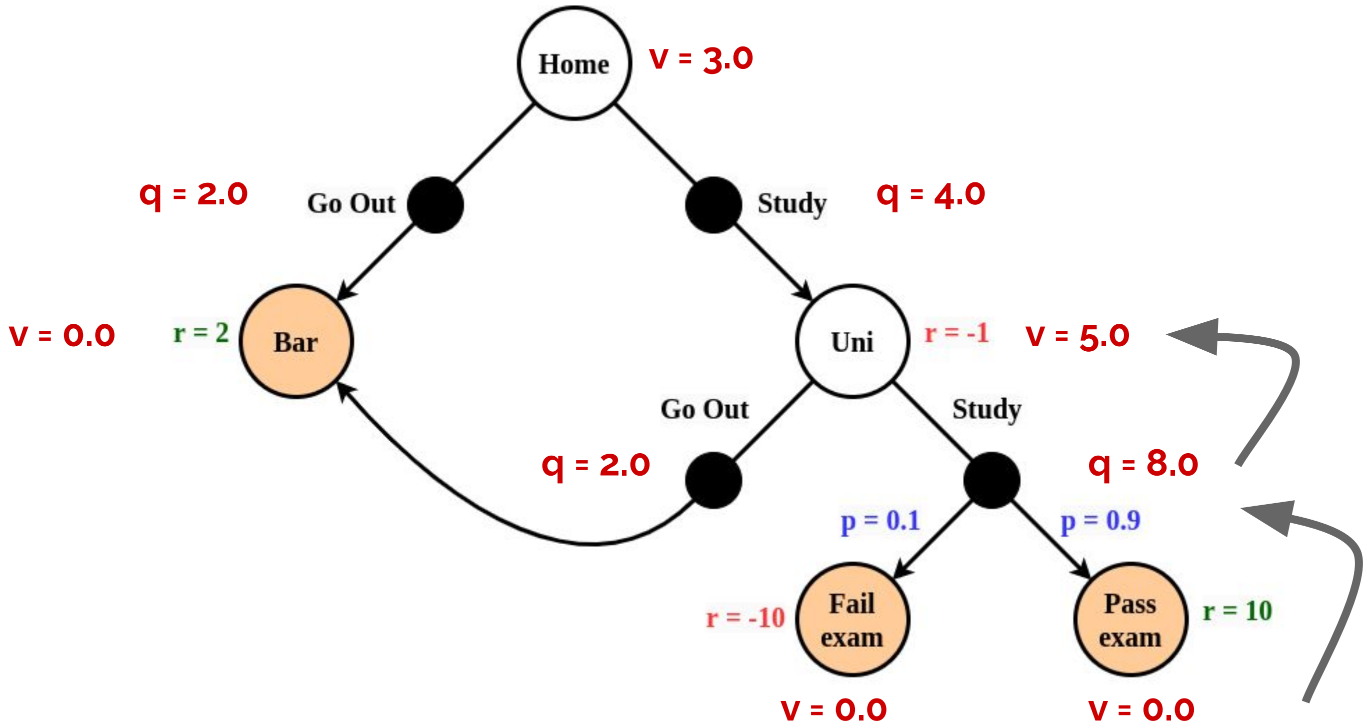
Example: The Study MDP



There is a recursive relation between the value estimates of states and actions

- Compute q from v and rewards by averaging over the environment stochasticity (EXP)

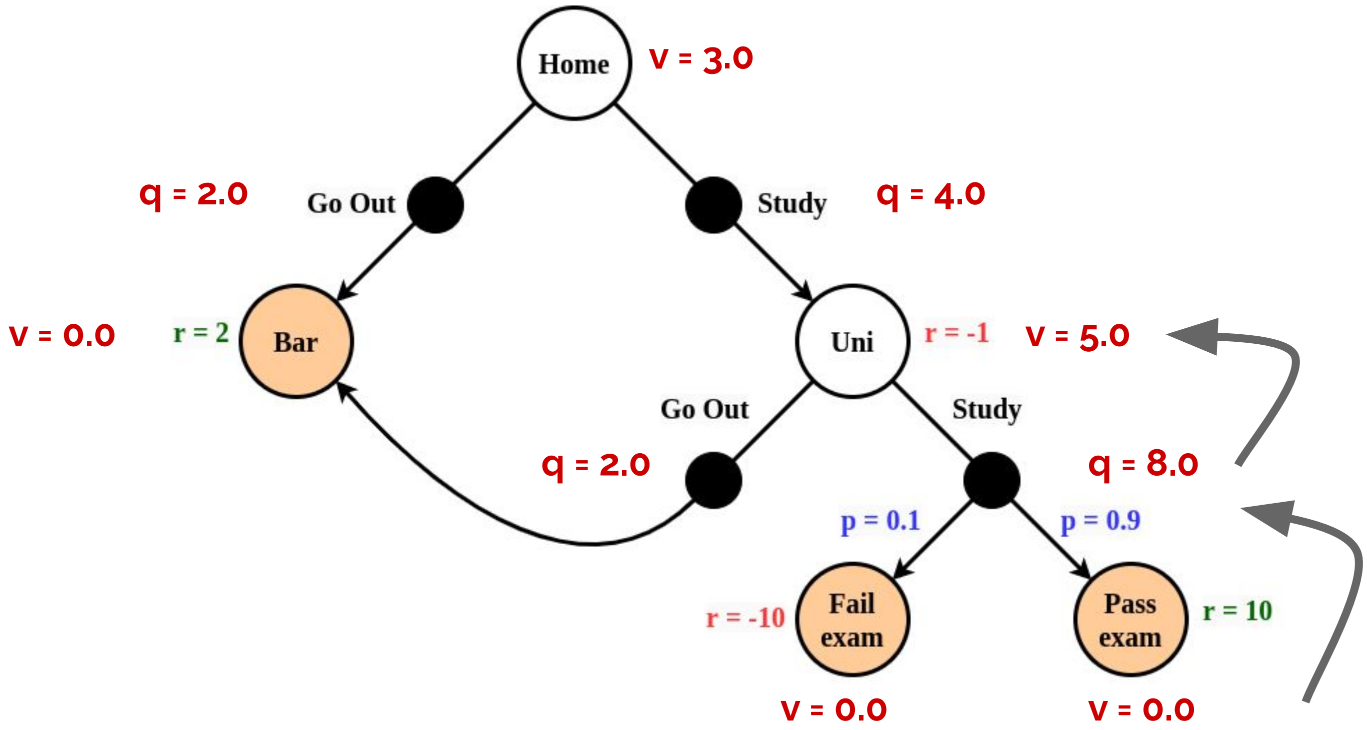
Example: The Study MDP



There is a recursive relation between the value estimates of states and actions

- Compute q from v and rewards by averaging over the environment stochasticity (EXP)
- Compute v from q depending on our own behaviour (for optimal: MAX)

Example: The Study MDP



There is a recursive relation between the value estimates of states and actions

- Compute q from v and rewards by averaging over the environment stochasticity (EXP)
- Compute v from q depending on our own behaviour (for optimal: MAX)

MDP = MAX-EXP graph problem

In the remainder of this lecture we will formalize these ideas,

In the remainder of this lecture we will formalize these ideas,

with the eventual goal to compute the optimal value function q^*/v^* for a given MDP,

In the remainder of this lecture we will formalize these ideas,

with the eventual goal to compute the optimal value function q^*/v^* for a given MDP,

since we then directly know how to act optimally.

Part III:

Markov Decision Process

Markov Decision Process (MDP)

Markov Decision Process (MDP)

Generic way to formally define a sequential decision-making problem.

Markov Decision Process (MDP)

Generic way to formally define a sequential decision-making problem.

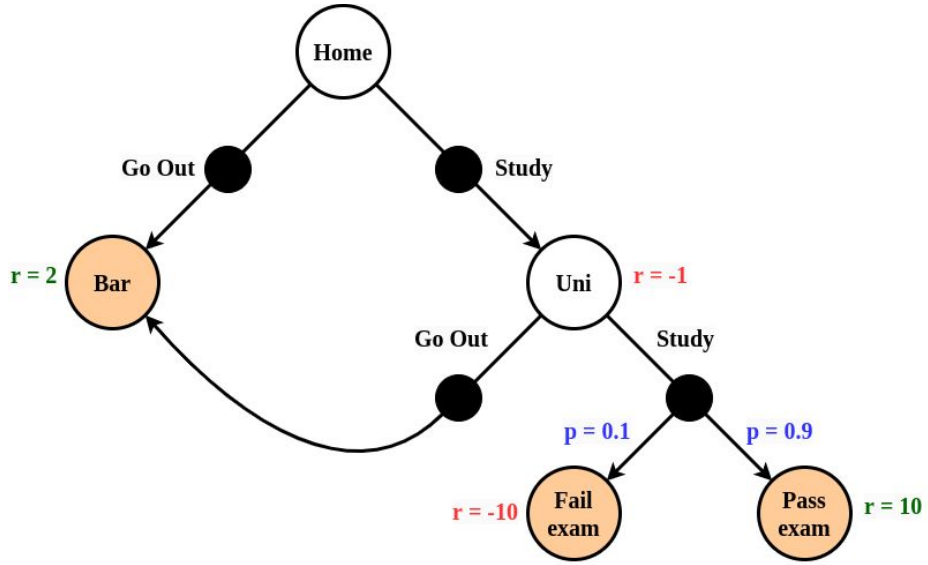
- Can handle *stochastic* environments (through a probabilistic transition function)

Markov Decision Process (MDP)

Generic way to formally define a sequential decision-making problem.

- Can handle *stochastic* environments (through a probabilistic transition function)
- Can trade-off *multiple goals* (through a reward function)

Example: The Study MDP



Let's formulate our problem as an MDP!

Markov Decision Process definition

An MDP consists of 5 elements

1. State space
2. Action space
3. Transition function
4. Reward function
5. Discount parameter

1. State space

Intuition:

Type:

Notation:

1. State space

Intuition: What observations are possible

Type:

Notation:

1. State space

Intuition: What observations are possible

Type: A discrete or continuous set/space

Notation:

1. State space

Intuition: What observations are possible

Type: A discrete or continuous set/space

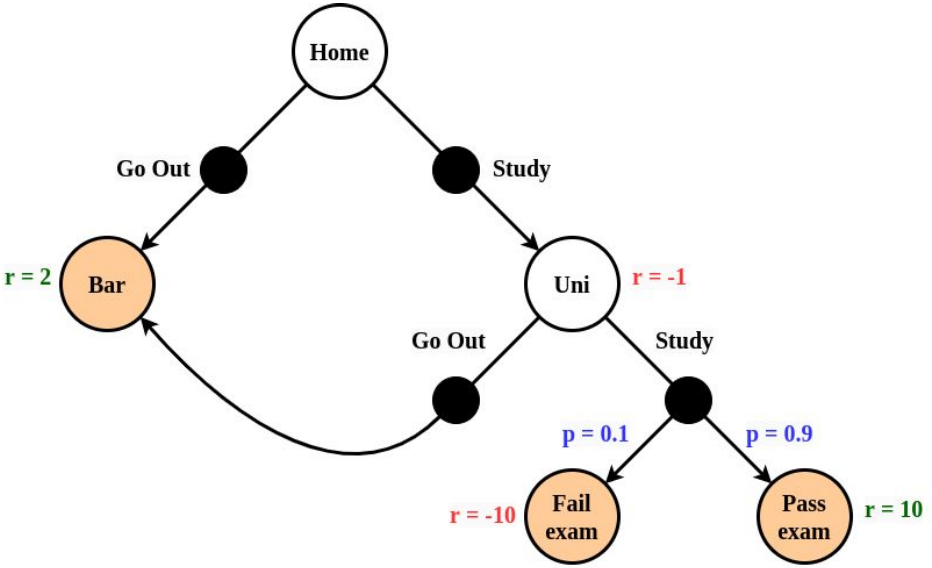
Notation: $\mathcal{S}$

1. State space

Intuition: What observations are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{S}$



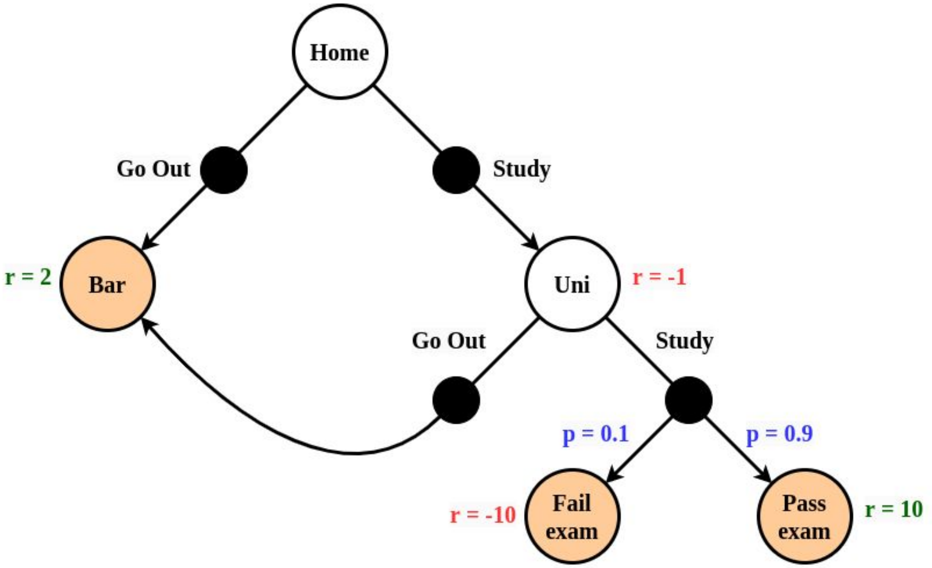
1. State space

Intuition: What observations are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{S}$

Q: What is the state space of this MDP?



1. State space

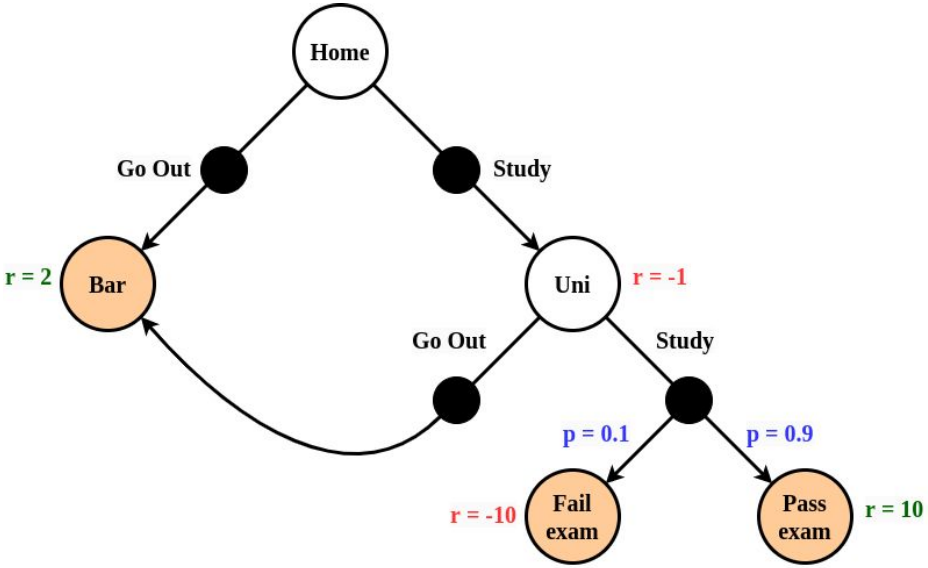
Intuition: What observations are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{S}$

Q: What is the state space of this MDP?

A: {Home, Bar, Uni, Fail exam, Pass exam}
(a discrete set of size 5)



Atomic versus factorized states

Atomic versus factorized states

Atomic

- Each state is a unique element

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states

Atomic versus factorized states

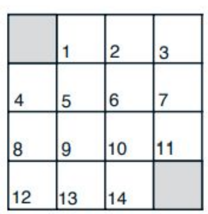
Atomic

- Each state is a unique element
- No relation between states
- Example: $s = 1$

Atomic versus factorized states

Atomic

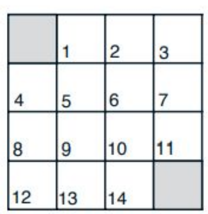
- Each state is a unique element
- No relation between states
- Example: s = 1



Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



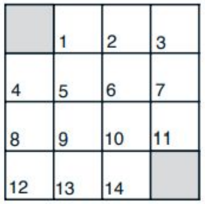
Factorized

- State is a vector/matrix of numbers

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



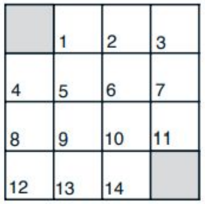
Factorized

- State is a vector/matrix of numbers
- Relation/overlap between states

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



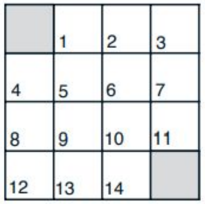
Factorized

- State is a vector/matrix of numbers
- Relation/overlap between states
- Example: s = (64,0,3,1)

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



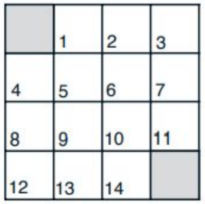
Factorized

- State is a vector/matrix of numbers
- Relation/overlap between states
- Example: s = (64,0,3,1)

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



Main focus of this course

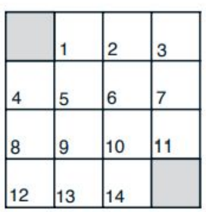
Factorized

- State is a vector/matrix of numbers
- Relation/overlap between states
- Example: s = (64,0,3,1)

Atomic versus factorized states

Atomic

- Each state is a unique element
- No relation between states
- Example: s = 1



Main focus of this course

Factorized

- State is a vector/matrix of numbers
- Relation/overlap between states
- Example: s = (64,0,3,1)

Allows for approximation & generalisation (e.g. deep learning)

Curse of Dimensionality

Curse of Dimensionality

The cardinality of a state space grows *exponentially* in the dimensionality of the space

Curse of Dimensionality

The cardinality of a state space grows *exponentially* in the dimensionality of the space

↑

The total number of possible unique states

(e.g., **[0,0,0]**, **[0,0,1]**, **[0,0,2]**, **[0,0,3]**, **[0,1,0]** etc.)

Curse of Dimensionality

The cardinality of a state space grows *exponentially* in the dimensionality of the space

The amount of elements in a single state
(e.g., **s=[2, 8,-4]** has dimensionality **3**)

Curse of Dimensionality

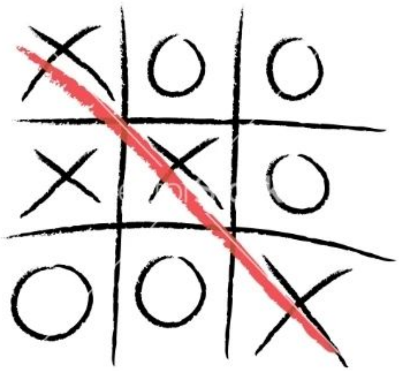
The cardinality of a state space grows *exponentially* in the dimensionality of the space

↑

Grows extremely fast!

Curse of Dimensionality: Illustration

Curse of Dimensionality: Illustration

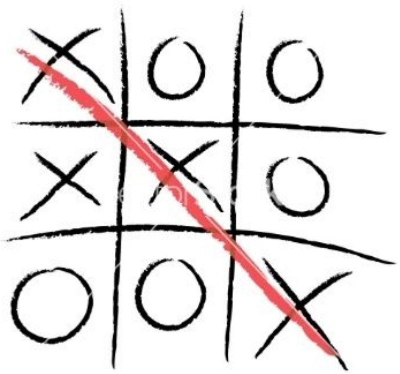


Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board

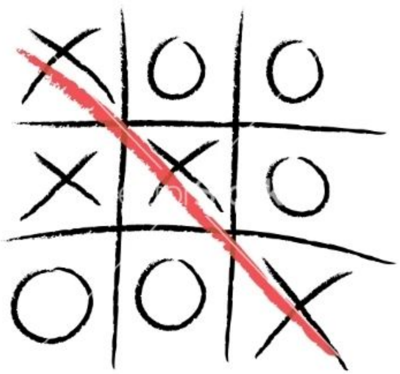
- Each matrix element in (X,O, Empty)



Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

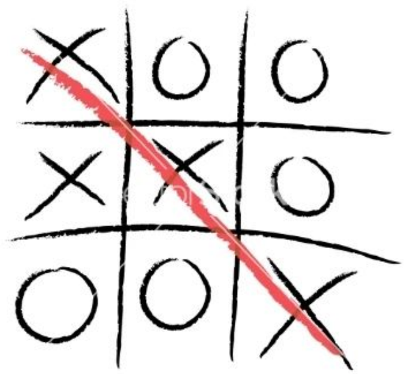


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

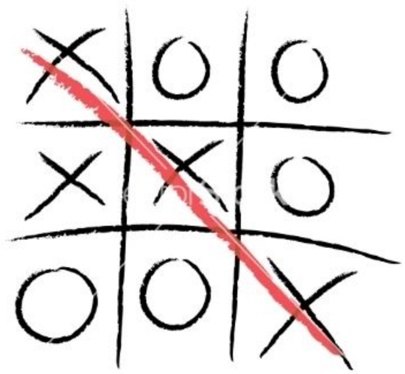


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3			

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

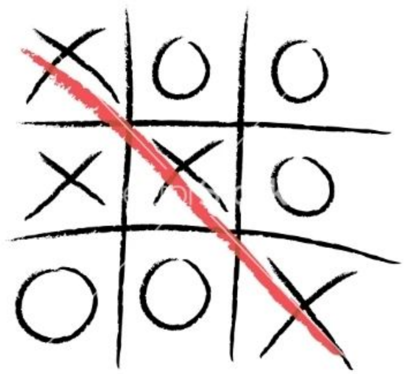


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9		

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

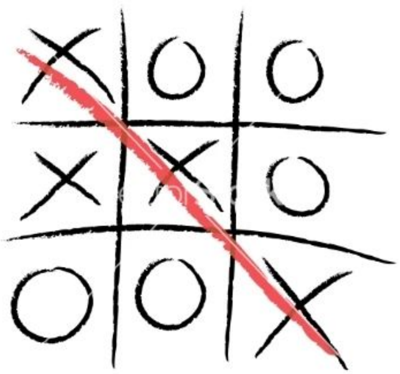


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	

Curse of Dimensionality: Illustration

<u>Tic-Tac-Toe state</u>:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

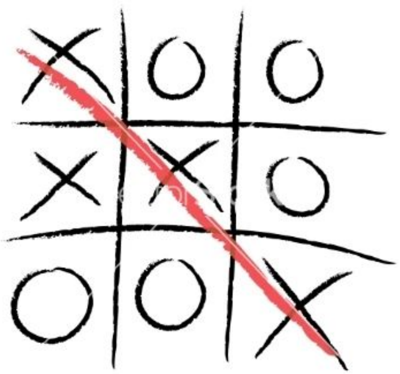


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

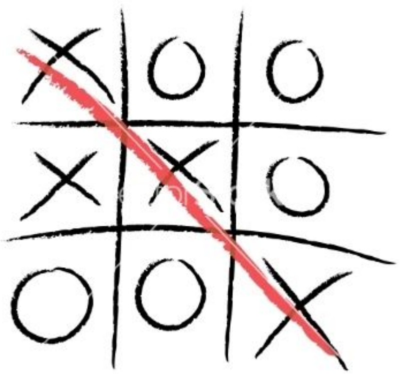


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4			

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

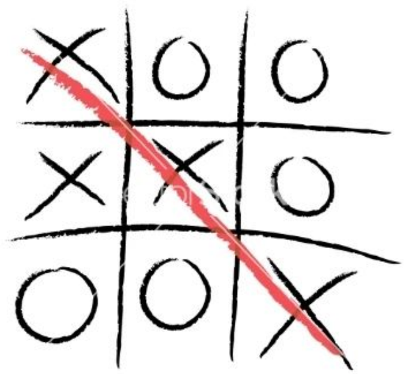


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4	16	3^{16} (~43 million)	164 MB

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

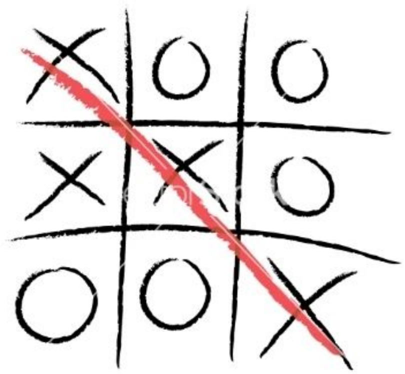


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4	16	3^{16} (~43 million)	164 MB
5 by 5			

Curse of Dimensionality: Illustration

<u>Tic-Tac-Toe state</u>:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)

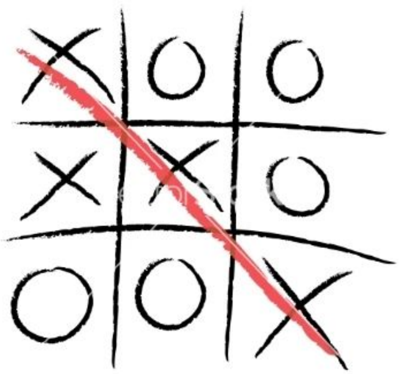


Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4	16	3^{16} (~43 million)	164 MB
5 by 5	25	3^{25} (~847 billion)	789 TB **(!)**

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)



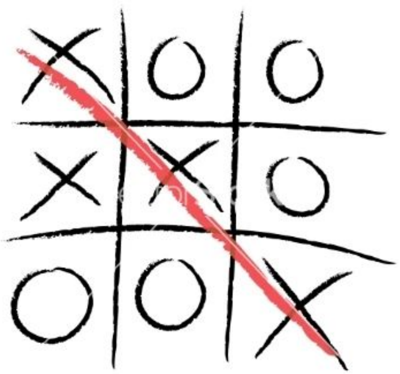
Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4	16	3^{16} (~43 million)	164 MB
5 by 5	25	3^{25} (~847 billion)	789 TB **(!)**

The total size of a state space grows very fast when its dimensionality increases

Curse of Dimensionality: Illustration

Tic-Tac-Toe state:

- Matrix representation of the board
- Each matrix element in (X,O, Empty)



Tic-Tac-Toe shape	Dimensionality	Cardinality	Memory (float-32)
3 by 3	9	3^9 (=19.683)	77 KB
4 by 4	16	3^{16} (~43 million)	164 MB
5 by 5	25	3^{25} (~847 billion)	789 TB **(!)**

The total size of a state space grows very fast when its dimensionality increases
i.e., tabular/atomic solutions only feasible in smaller problems

2. Action space

Intuition:

Type:

Notation:

2. Action space

Intuition: What actions are possible

Type:

Notation:

2. Action space

Intuition: What actions are possible

Type: A discrete or continuous set/space

Notation:

2. Action space

Intuition: What actions are possible

Type: A discrete or continuous set/space

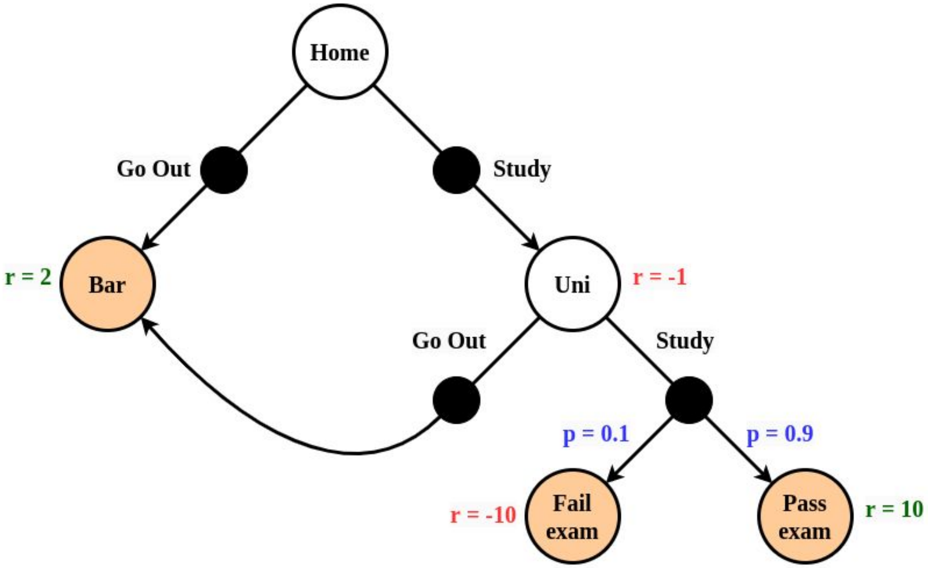
Notation: $\mathcal{A}$

2. Action space

Intuition: What actions are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{A}$



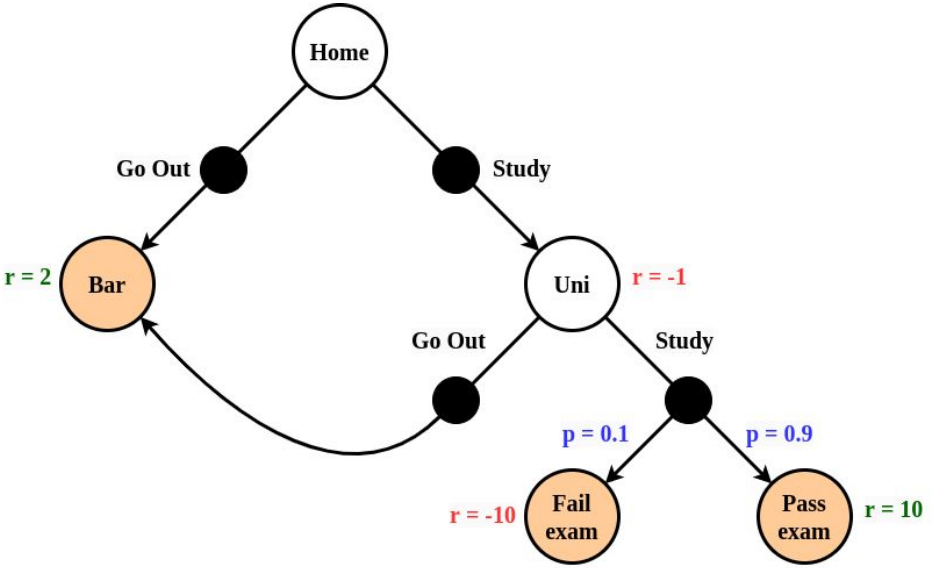
2. Action space

Intuition: What actions are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{A}$

Q: What is the action space of our Study MDP?

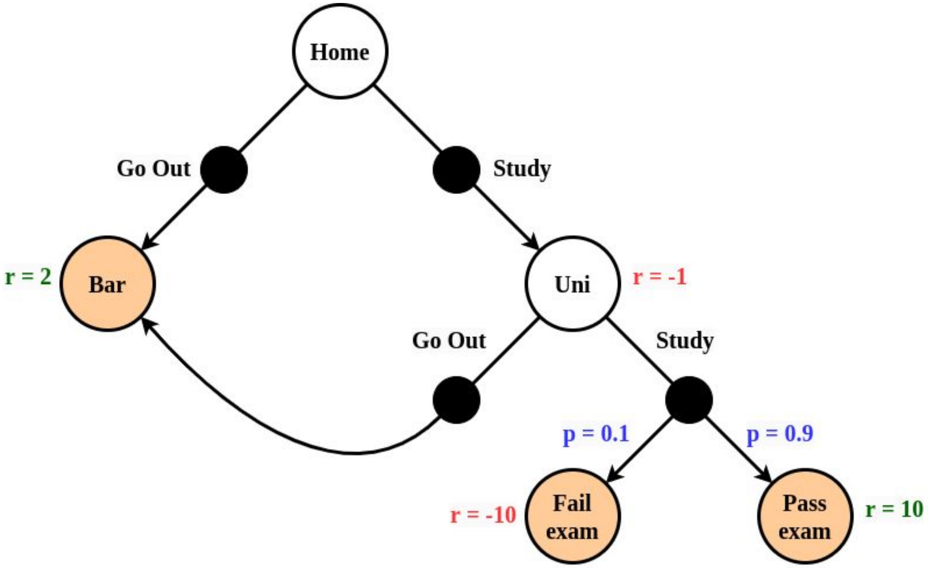


2. Action space

Intuition: What actions are possible

Type: A discrete or continuous set/space

Notation: $\mathcal{A}$



Q: What is the action space of our Study MDP?

A: {Go Out, Study}
(a discrete set of size 2)

3. Transition function

Intuition:

Type:

Notation:

3. Transition function

Intuition: What is the effect of an action in a certain situation

Type:

Notation:

3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation:

3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s, a)$

3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s,a)$

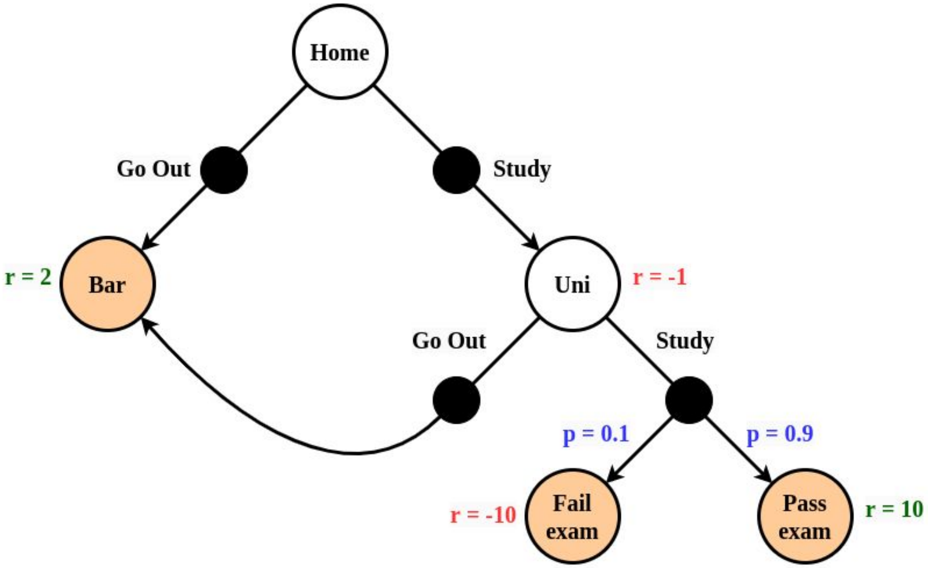
We always write **s'** to denote the next state after taking action **a** in state **s**

3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s, a)$



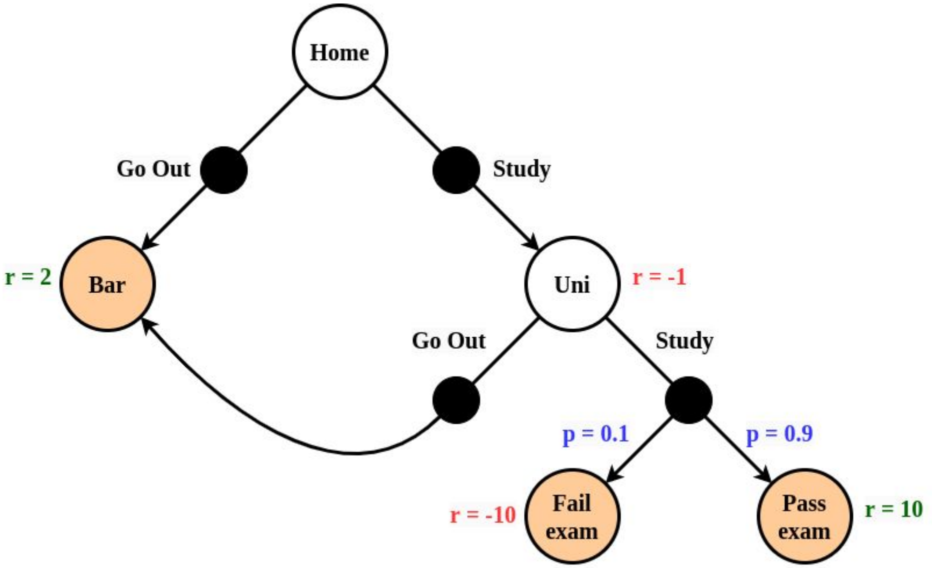
3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s,a)$

Q: What is the p(Uni | Home, Study)?



3. Transition function

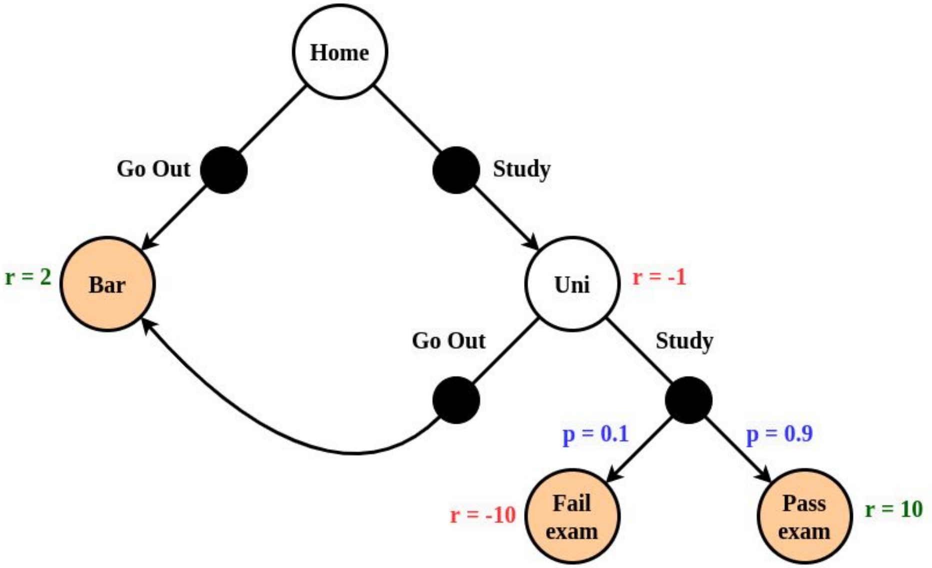
Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s,a)$

Q: What is the p(Uni | Home, Study)?

A: 1.0



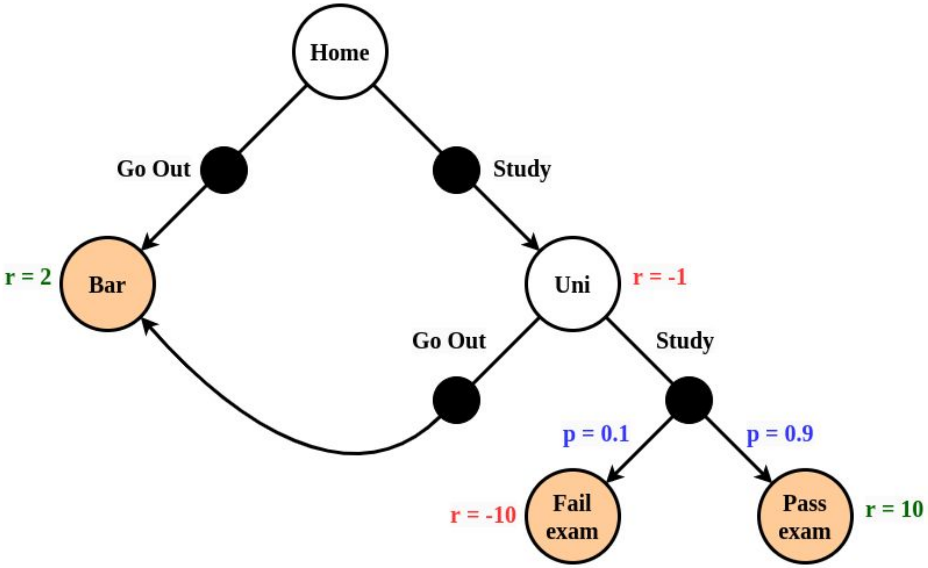
3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s, a)$

Q: What is the p(Uni | Home, Go Out)?

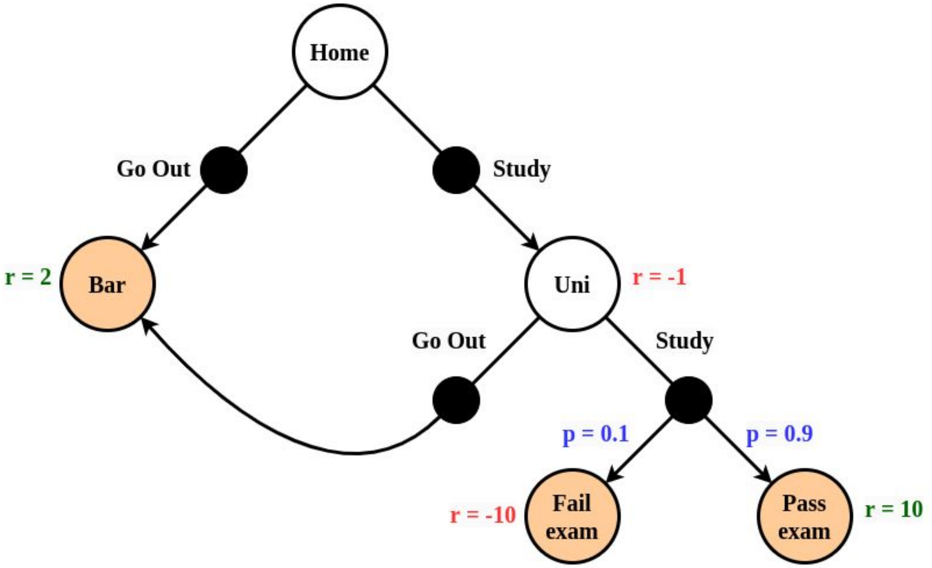


3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s, a)$



Q: What is the p(Uni | Home, Go Out)?

A: 0.0 (impossible)

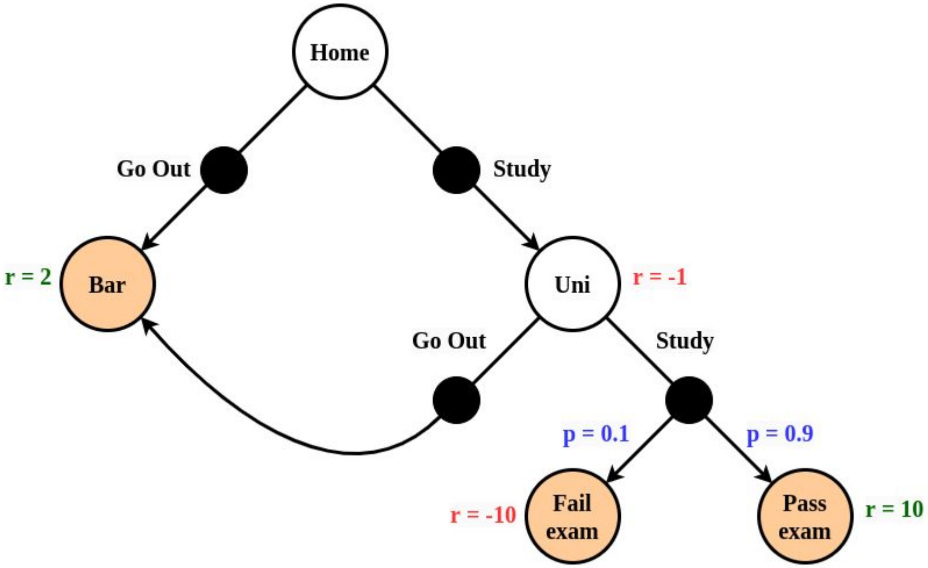
3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s, a)$

Q: What is the p(Pass exam | Uni, Study)?

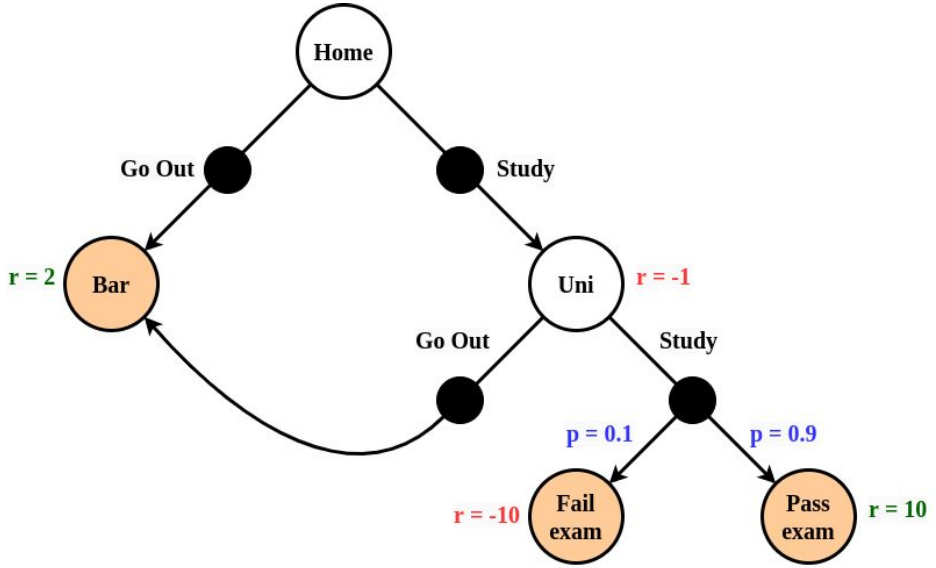


3. Transition function

Intuition: What is the effect of an action in a certain situation

Type: A conditional probability distribution

Notation: $p(s'|s,a)$



Q: What is the p(Pass exam | Uni, Study)?

A: 0.9 (stochastic dynamics!)

3. Transition function

For atomic state and action spaces, the transition function can be stored as an array of size

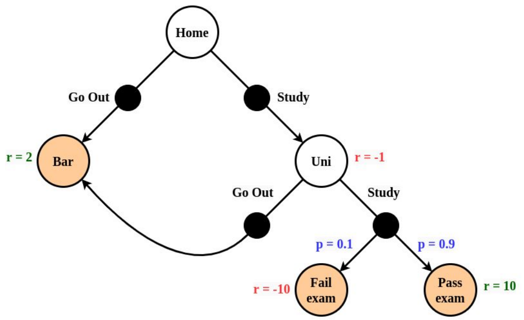
….

3. Transition function

For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

3. Transition function

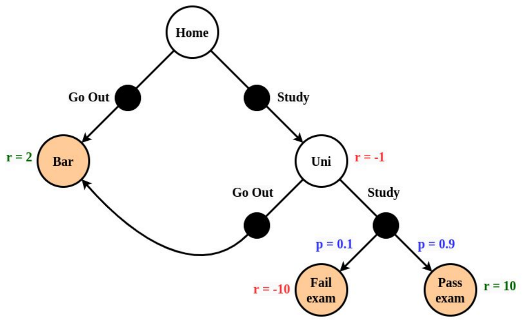


For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

s	**a**	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9

3. Transition function



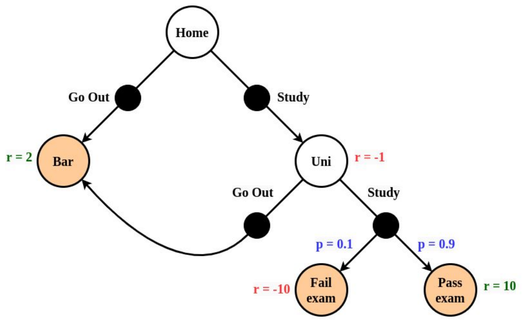
For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

s	**a**	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9

When we are at home and go out, we always end up in the bar

3. Transition function



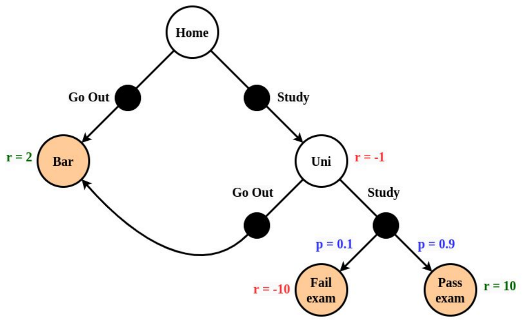
For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

s	**a**	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9

When we are at uni and go out, we also always end up in the bar

3. Transition function



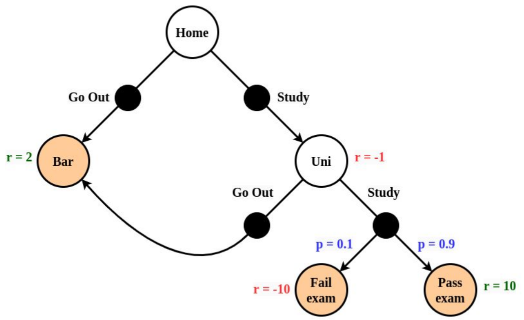
For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

s	a	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9

When we are at uni and study, we 10% fail the exam, and 90% pass the exam (*stochastic* transition)

3. Transition function



For atomic state and action spaces, the transition function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

s	**a**	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9

But what about the transitions from the terminal states?

Terminal states: two valid perspectives

Terminal states: two valid perspectives

1) No available actions (and therefore transition function undefined = previous slide)

Terminal states: two valid perspectives

1) No available actions (and therefore transition function undefined = previous slide)

2) All actions lead back to the same state with a reward of 0

Terminal states: two valid perspectives

1) No available actions (and therefore transition function undefined = previous slide)

2) All actions lead back to the same state with a reward of 0

s	a	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9
Bar	Go Out	0.0	1.0	0.0	0.0	0.0
Bar	Study	0.0	1.0	0.0	0.0	0.0
Fail exam	Go Out	0.0	0.0	0.0	1.0	0.0
Fail exam	Study	0.0	0.0	0.0	1.0	0.0
Pass exam	Go Out	0.0	0.0	0.0	0.0	1.0
Pass exam	Study	0.0	0.0	0.0	0.0	1.0

Terminal states: two valid perspectives

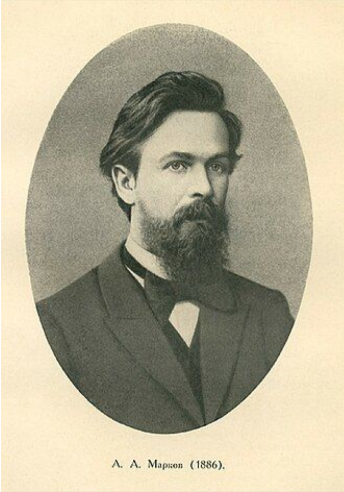
1) No available actions (and therefore transition function undefined = previous slide)

2) All actions lead back to the same state with a reward of 0

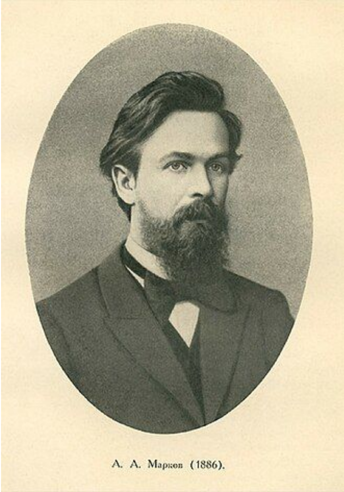
s	a	Home	Bar	Uni	Fail Exam	Pass exam
Home	Go Out	0.0	1.0	0.0	0.0	0.0
Home	Study	0.0	0.0	1.0	0.0	0.0
Uni	Go Out	0.0	1.0	0.0	0.0	0.0
Uni	Study	0.0	0.0	0.0	0.1	0.9
Bar	Go Out	0.0	1.0	0.0	0.0	0.0
Bar	Study	0.0	1.0	0.0	0.0	0.0
Fail exam	Go Out	0.0	0.0	0.0	1.0	0.0
Fail exam	Study	0.0	0.0	0.0	1.0	0.0
Pass exam	Go Out	0.0	0.0	0.0	0.0	1.0
Pass exam	Study	0.0	0.0	0.0	0.0	1.0

Markov Property

Markov Property



Markov Property



Andrey Markov
(1865 - 1922)

Markov Property

‘The future only depends on the present and not on past history’

Markov Property

'The future only depends on the present and not on past history'

$$p(s'|s, a)$$

Markov Property

'The future only depends on the present and not on past history'

$$p(s'|s, a)$$

Markov Property

‘The future only depends on the present and not on past history’

$$p(s'|s, a)$$

Fundamental assumption of the Markov Decision Formulation

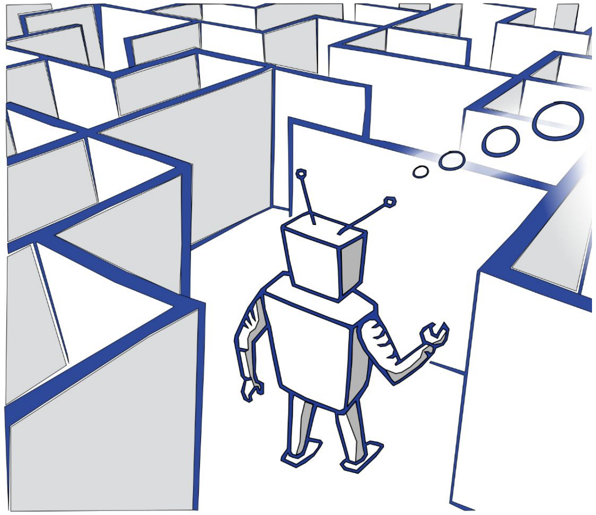
Partial Observability

Partial Observability

Real-world tasks are actually usually not Markovian, they suffer from *partial observability*

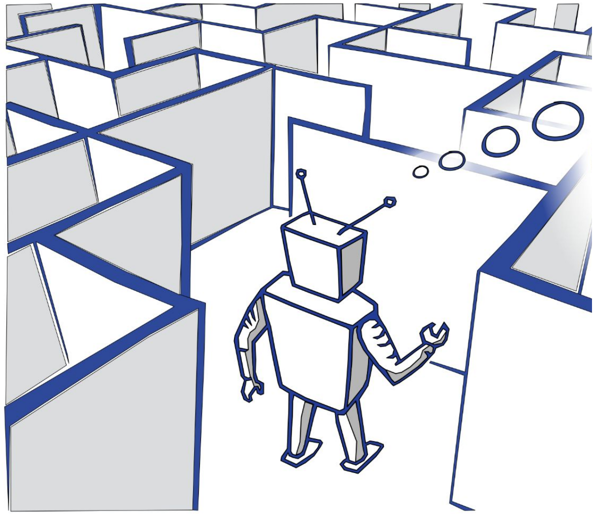
Partial Observability

Real-world tasks are actually usually not Markovian, they suffer from *partial observability*



Partial Observability

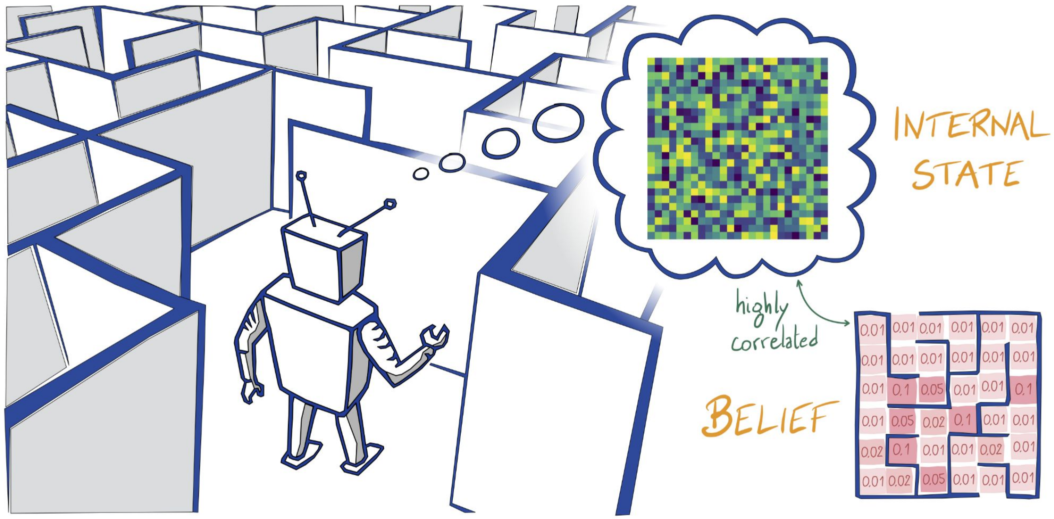
Real-world tasks are actually usually not Markovian, they suffer from *partial observability*



'Partially Observable Markov Decision Process' (POMDP)

Partial Observability

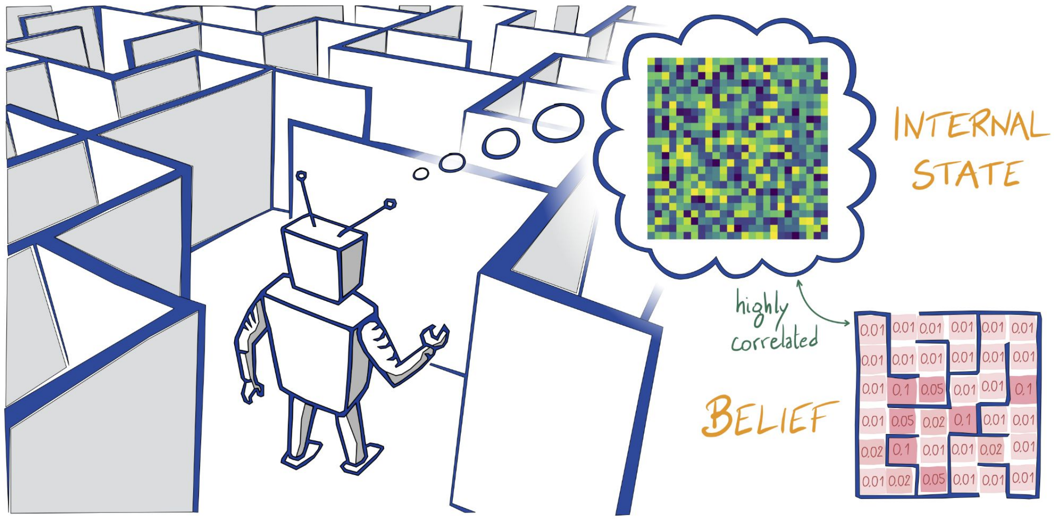
Real-world tasks are actually usually not Markovian, they suffer from *partial observability*



Solution requires some form of memory

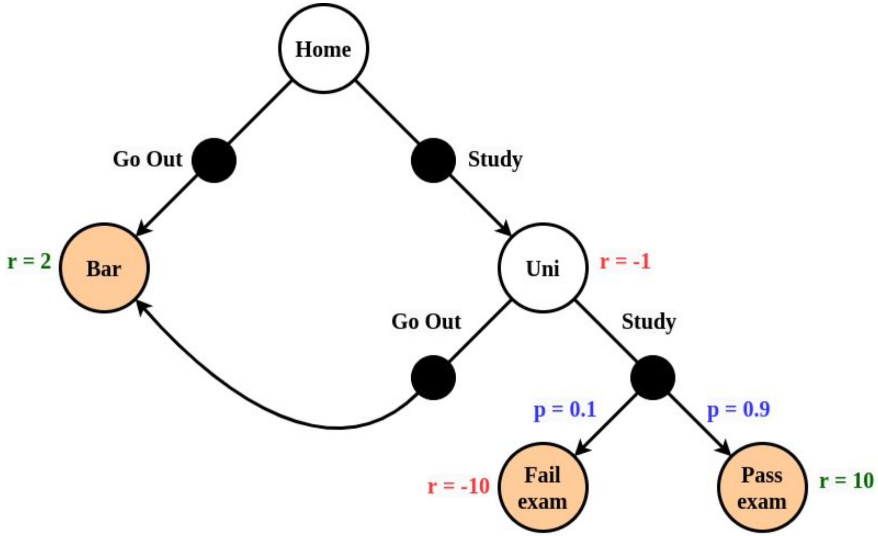
Partial Observability

Real-world tasks are actually usually not Markovian, they suffer from *partial observability*



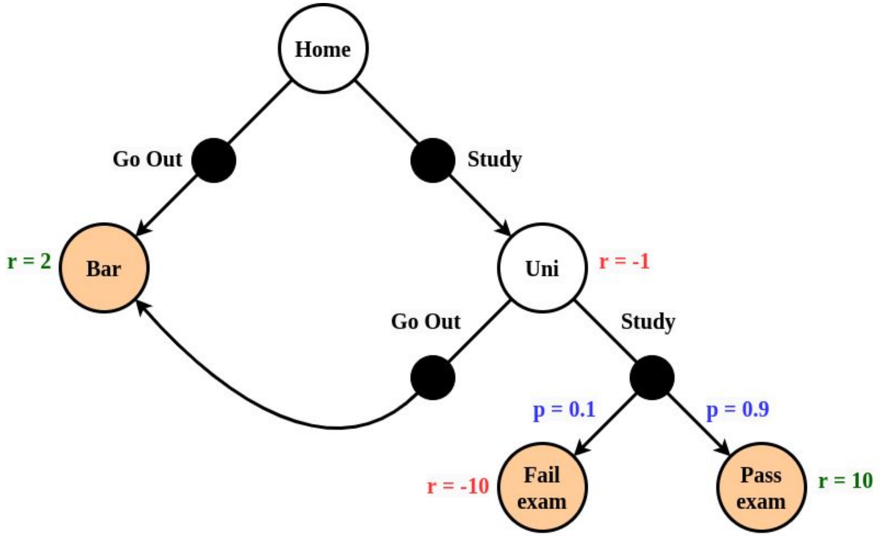
Solution requires some form of memory
(we will skip this topic for now, and assume full observability / Markovianity)

Loops



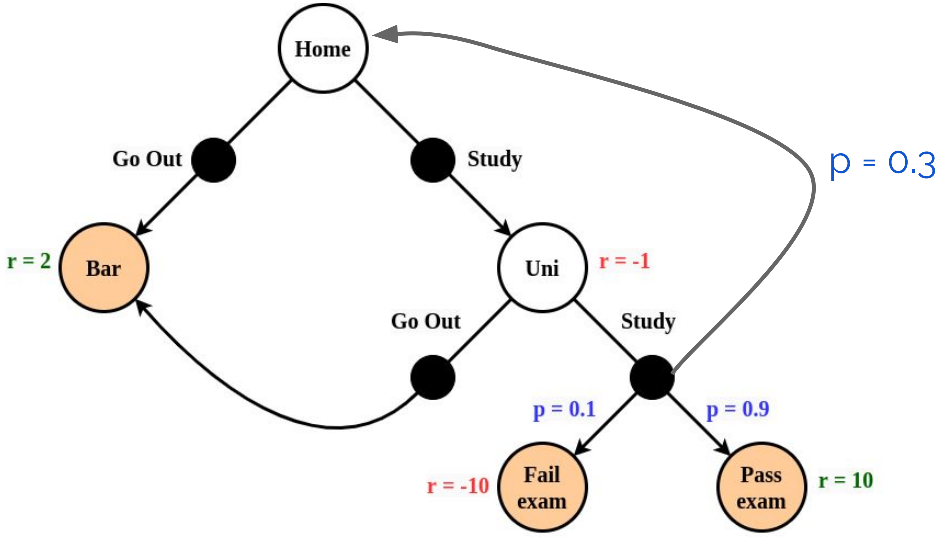
Home
Go Out
Study
r = 2
Bar
Uni
r = -1
Go Out
Study
p = 0.1
p = 0.9
r = -10
Fail exam
Pass exam
r = 10

Loops



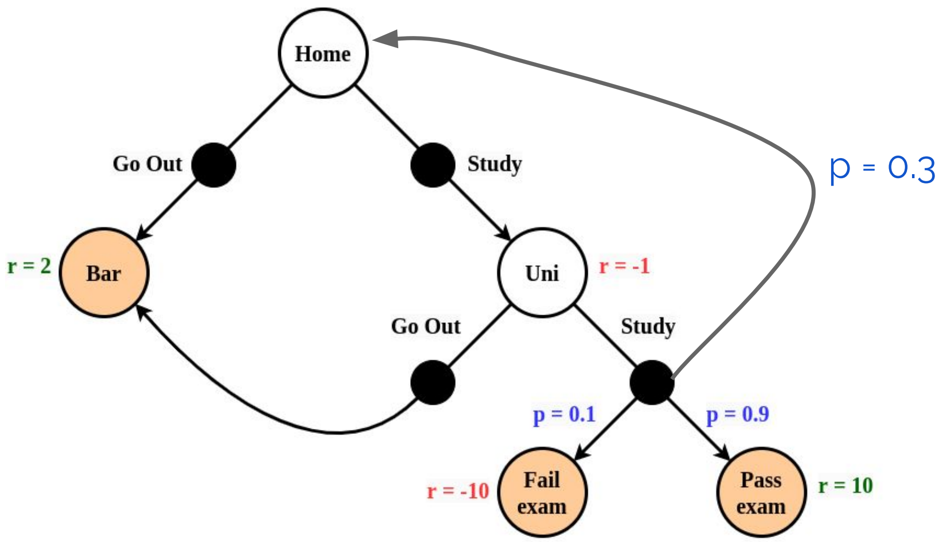
- Our toy MDP is a *directed acyclic graph*:
    - Can only move from top to bottom - useful for conceptual illustration.

Loops



Home
Go Out
Study
p = 0.3
r = 2
Bar
Uni
r = -1
Go Out
Study
p = 0.1
p = 0.9
r = -10
Fail exam
Pass exam
r = 10

Loops



- In practice, MDPs are *directed cyclic graphs*: they contain (many) *loops*
    - Same principles still apply & our later solution methods naturally handle loops

4. Reward function

Intuition:

Type:

Notation:

4. Reward function

Intuition: How good or bad is a certain transition

Type:

Notation:

4. Reward function

Intuition: How good or bad is a certain transition

Type: Function

Notation:

4. Reward function

Intuition: How good or bad is a certain transition

Type: Function

Notation: $r(s, a, s')$

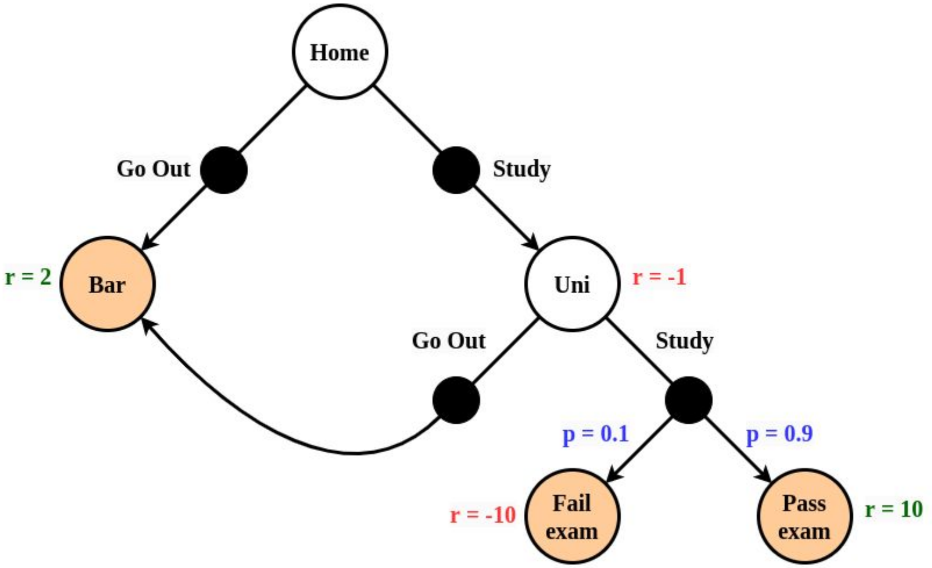
4. Reward function

Intuition: How good or bad is a certain transition

Type: Function

Notation: $r(s, a, s')$

Q: What is r(Uni,Study,Pass exam)?

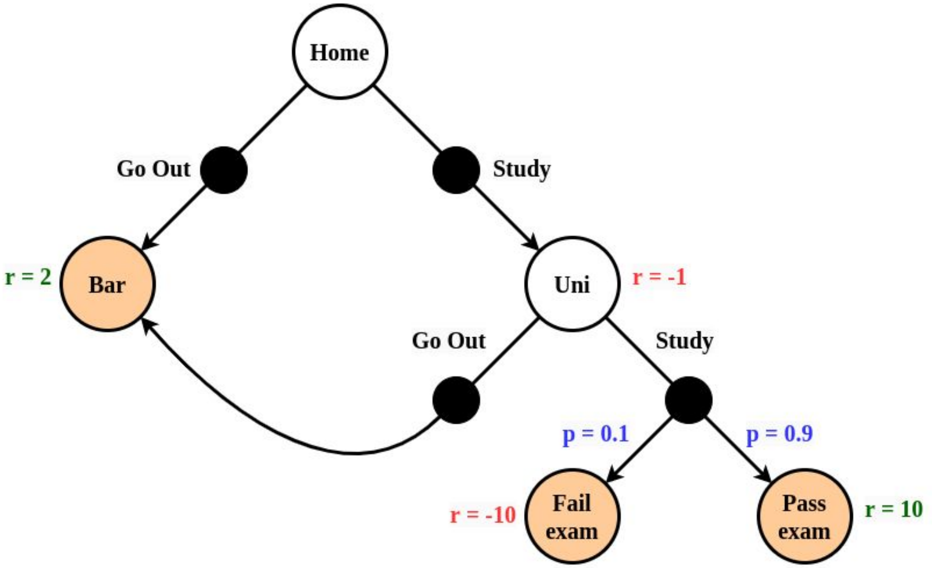


4. Reward function

Intuition: How good or bad is a certain transition

Type: Function

Notation: $r(s, a, s')$



Q: What is r(Uni,Study,Pass exam)?

A: 10.0

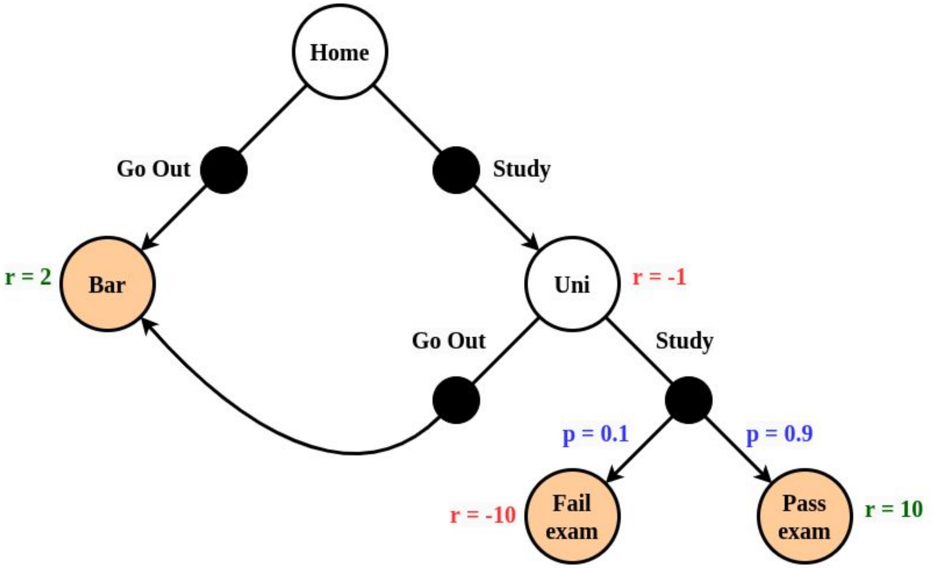
4. Reward function

Intuition: How good or bad is a certain transition

Type: Function

Notation: $r(s, a, s')$

Q: What is r(Uni,Study,Home)?



4. Reward function

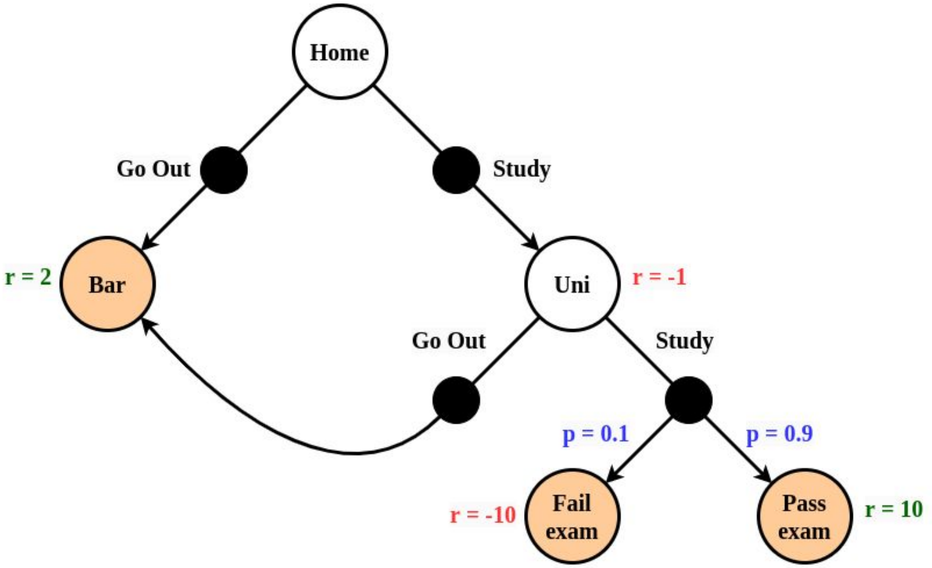
Intuition: How good or bad is a certain transition

Type: Function

Notation: $r(s, a, s')$

Q: What is r(Uni,Study,Home)?

A: undefined (transition impossible)



4. Reward function

For atomic state and action spaces, the reward function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

4. Reward function

For atomic state and action spaces, the reward function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

However, since some transitions will be impossible, we may also store:

$$r(s, a) \quad \text{or} \quad r(s')$$

4. Reward function

For atomic state and action spaces, the reward function can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}| \times |\mathcal{S}|$$

However, since some transitions will be impossible, we may also store:

$r(s, a)$ or $r(s')$

s'	$r(s')$
Home	-
Bar	2.0
Uni	-1.0
Fail exam	-10.0
Pass exam	10.0

Reward versus cost

Reward versus cost

Differences in terminology per field

- Path planning uses **cost** per step, reinforcement learning uses **reward** per step

Reward versus cost

Differences in terminology per field

- Path planning uses **cost** per step, reinforcement learning uses **reward** per step

- But cost is negative reward:

$$c(s,a,s') = -\,r(s,a,s')$$

Reward versus cost

Differences in terminology per field

- Path planning uses **cost** per step, reinforcement learning uses **reward** per step

- But cost is negative reward:

c(s,a,s') = - r(s,a,s')

- Therefore:

Cost minimization = **reward maximization**

(planning) (reinforcement learning)

5. Discount factor

Intuition:

Type:

Notation:

5. Discount factor

Intuition: How much do we ignore long-term rewards

Type:

Notation:

5. Discount factor

Intuition: How much do we ignore long-term rewards

Type: Scalar (constant)

Notation:

5. Discount factor

Intuition: How much do we ignore long-term rewards

Type: Scalar (constant)

Notation: $\gamma \in [0, 1]$

5. Discount factor

Intuition: How much do we ignore long-term rewards

Type: Scalar (constant)

Notation: $\gamma \in [0, 1]$

We will discuss this in a few slides

Markov Decision Process: Summary

Item	Symbol	Description
State space	$\mathcal{S}$	Which observations are possible
Action space	$\mathcal{A}$	Which actions are possible
Transition function	$p(s'\|s, a)$	What is the effect of an action in a state
Reward function	$r(s, a, s')$	How good or bad is a certain transition
Discount factor	γ	How much do we ignore long-term reward

Break

Part IV:

Policy

How do we actually *act* in the Markov Decision Process?

Policy

Policy

Intuition:

Type:

Notation:

Policy

Intuition: Specify probability of each action for every possible state

Type:

Notation:

Policy

Intuition: Specify probability of each action for every possible state

Type: Conditional probability distribution

Notation:

Policy

Intuition: Specify probability of each action for every possible state

Type: Conditional probability distribution

Notation: $\pi(a|s)$

Policy

For atomic state and action spaces, the policy can be stored as an array of size

….

Policy

For atomic state and action spaces, the policy can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}|$$

Policy

For atomic state and action spaces, the policy can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}|$$

	a	
s	Go out	Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Policy

For atomic state and action spaces, the policy can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}|$$

s	**a** Go out	Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

For every state we specify the probability of each possible action

Policy

For atomic state and action spaces, the policy can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}|$$

s	**a** Go out	Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

(rows need to sum to 1.0 to make it a valid probability distribution)

For every state we specify the probability of each possible action

Policy

For atomic state and action spaces, the policy can be stored as an array of size

$$|\mathcal{S}| \times |\mathcal{A}|$$

s	a: Go out	a: Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Terminal states don't have a policy defined (no actions available)

Random policy

Random policy

Random policy: per state every action has the same probability of selection

Random policy

Random policy: per state every action has the same probability of selection

	a	
s	Go out	Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Random policy

Random policy: per state every action has the same probability of selection

	a	
s	Go out	Study
Home	0.5	0.5
Uni	0.5	0.5
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Deterministic policy

Deterministic policy

Deterministic policy: in every state we always select one particular action

Deterministic policy

Deterministic policy: in every state we always select one particular action

s	**a** Go out	Study
Home	1.0	0.0
Uni	0.0	1.0
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Deterministic policy

Deterministic policy: in every state we always select one particular action

s	**a** Go out	Study
Home	1.0	0.0
Uni	0.0	1.0
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Deterministic policy

Deterministic policy: in every state we always select one particular action

s	a: Go out	a: Study
Home	1.0	0.0
Uni	0.0	1.0
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Shorthand notation: $\pi(s)$

Deterministic policy

Deterministic policy: in every state we always select one particular action

s	Go out	Study
Home	1.0	0.0
Uni	0.0	1.0
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Shorthand notation: $\pi(s)$

Example: $\pi(\text{Home}) = \text{Go Out}$

Deterministic policy

Deterministic policy: in every state we always select one particular action

s	**a** Go out	Study
Home	1.0	0.0
Uni	0.0	1.0
Bar	-	-
Pass exam	-	-
Fail exam	-	-

Shorthand notation: $\pi(s)$

Example: $\pi(\text{Home}) = \text{Go Out}$ is short for $\pi(\text{Go Out}|\text{Home}) = 1.0$

Trace

Trace

Trace

When we act in the MDP we obtain a *trace*: a sequence of state-action-reward pairs

Trace

When we act in the MDP we obtain a *trace*: a sequence of state-action-reward pairs

$$\tau = \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, ...\}$$

Trace

When we act in the MDP we obtain a *trace*: a sequence of state-action-reward pairs

$$\tau = \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots\}$$

We use subscript *t* to indicate the timestep

Trace

When we act in the MDP we obtain a *trace*: a sequence of state-action-reward pairs

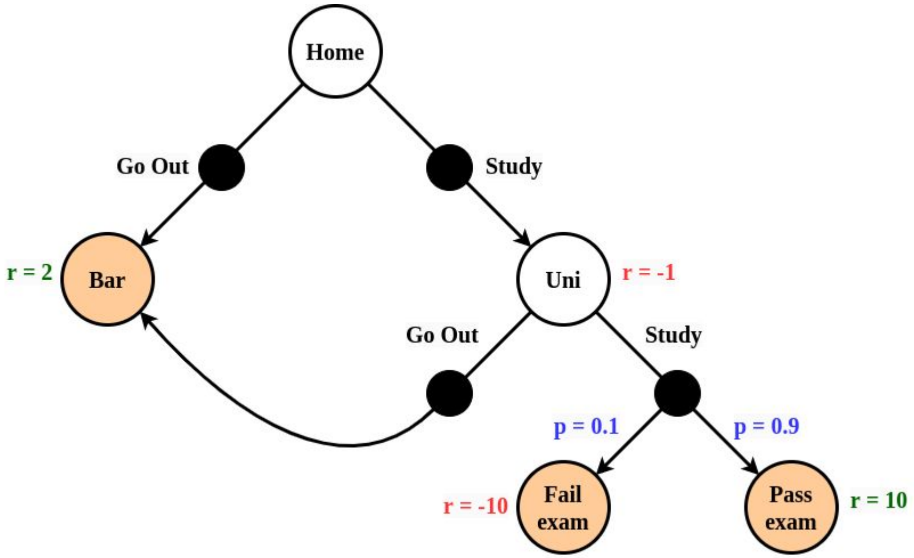
$$\tau = \{s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \ldots\}$$

We use subscript *t* to indicate the timestep

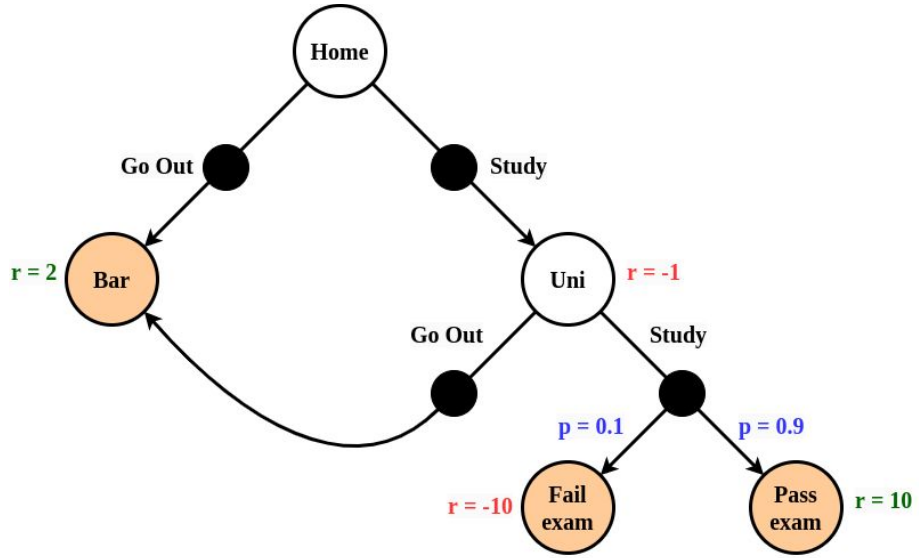
We use greek letter τ to refer to the entire trace

Trace: Illustration

Trace: Illustration

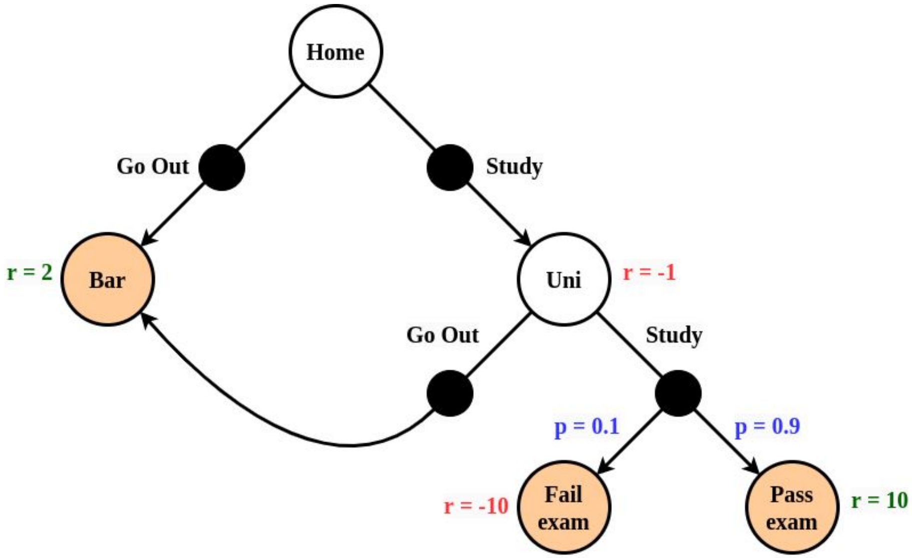


Trace: Illustration



Question: How many unique traces are possible from Home?

Trace: Illustration

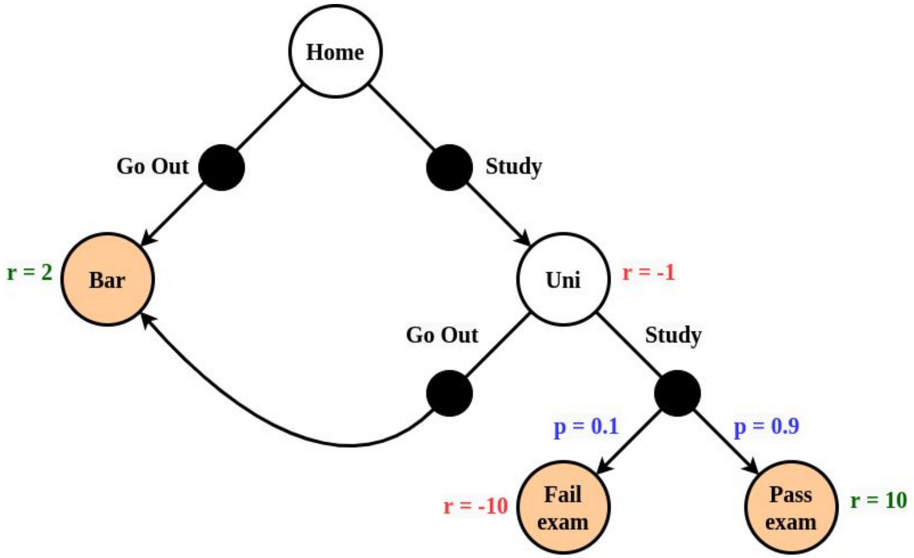


Question: How many unique traces are possible from Home?

Answer:

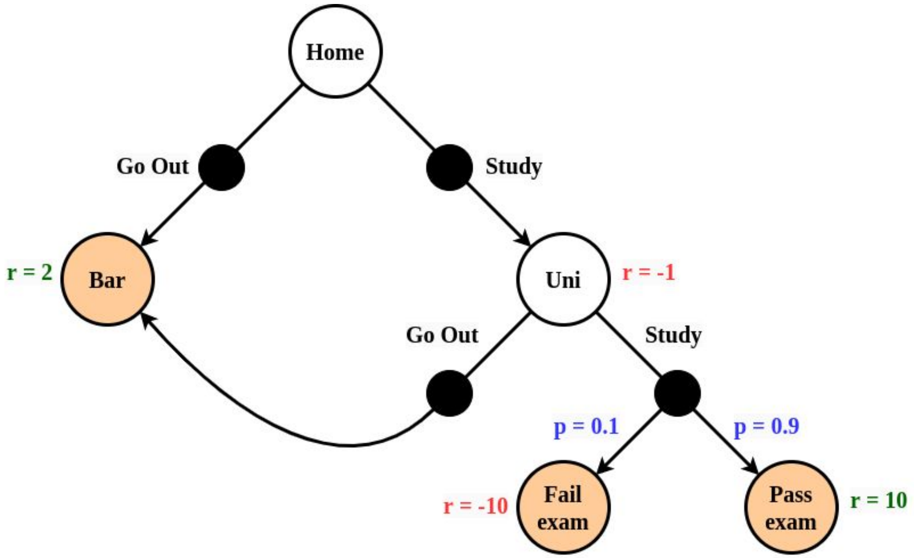
Trace (τ)
Home - Go Out - Bar
Home - Study - Uni - Go Out - Bar
Home - Study - Uni - Study - Fail exam
Home - Study - Uni - Study - Pass exam

Trace: Illustration



Question: What is the probability of each of these traces?

Trace: Illustration



Question: What is the probability of each of these traces?

Answer: We don't know, since we have not specified a policy (yet)

Trace probability

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot ...$$

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot ...$$

The probability of the full trace is equal to

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot ...$$

the probability we select the first action

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot \ldots$$

times the probability we observe the first reward and next state

Trace probability

To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot \ldots$$

times the probability we select a certain next action

Trace probability

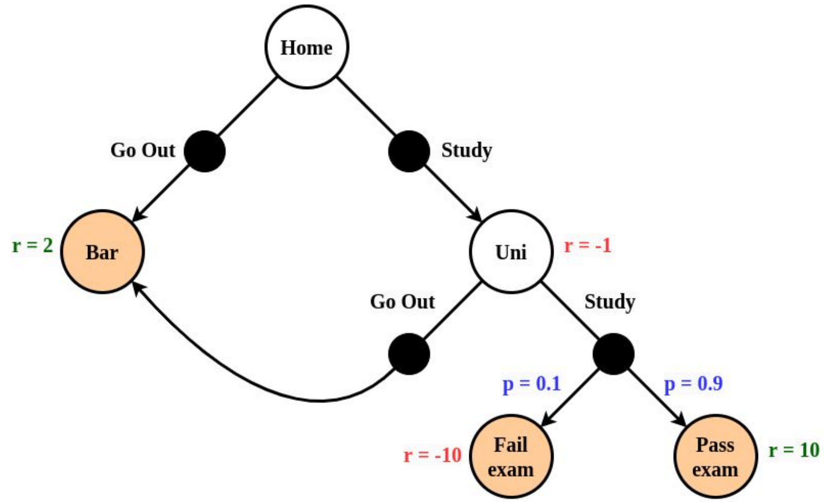
To compute the probability of a trace we multiply all individual policy and transition probabilities in the trace (= *product rule of probability*)

$$p(\tau) = \pi(a_t|s_t) \cdot p(r_t, s_{t+1}|s_t, a_t) \cdot \pi(a_{t+1}|s_{t+1}) \cdot p(r_{t+1}, s_{t+2}|s_{t+1}, a_{t+1}) \cdot ...$$

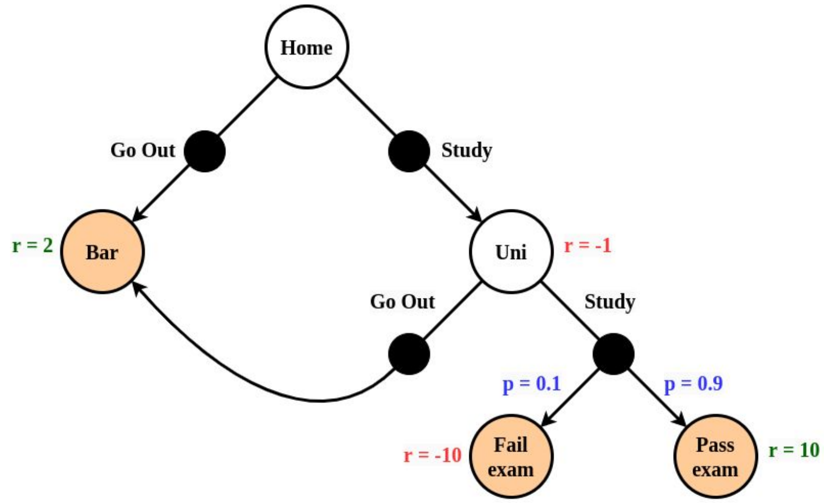
etc.

Trace probability: Illustration

Trace probability: Illustration



Trace probability: Illustration

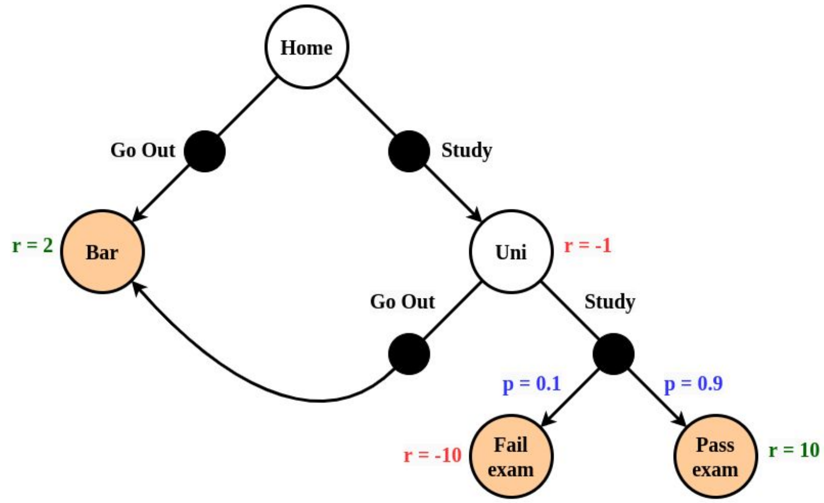


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar		
Home - Study - Uni - Go Out - Bar		
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

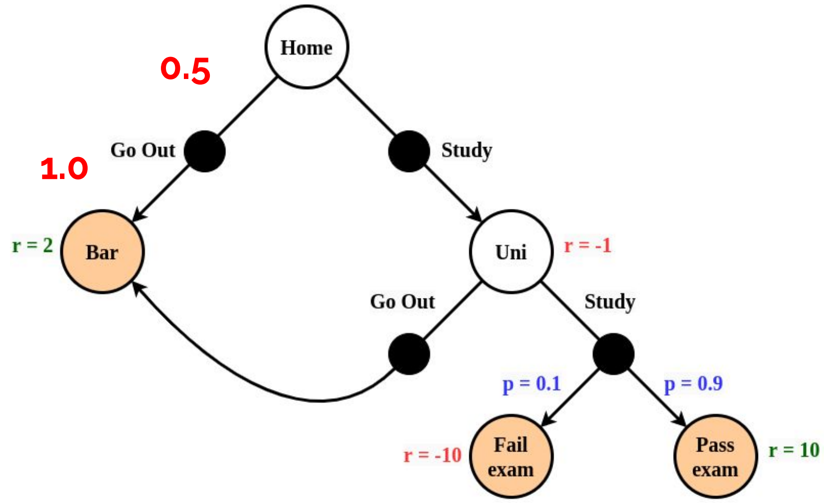


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar		
Home - Study - Uni - Go Out - Bar		
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

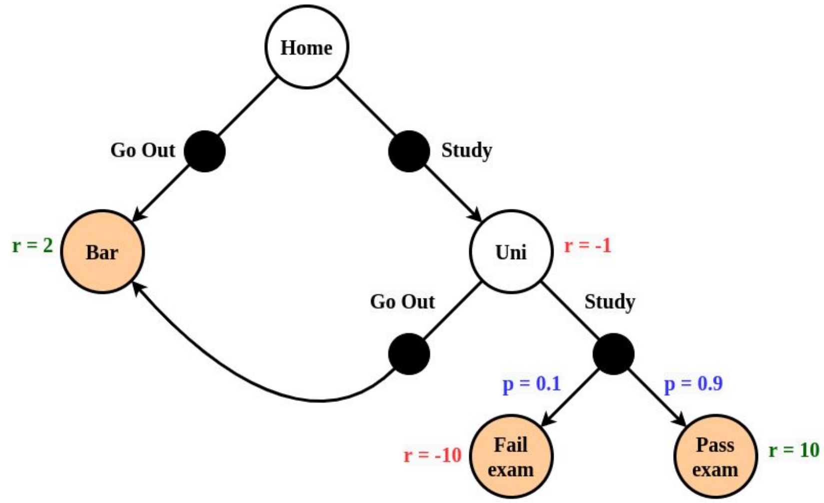


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar		
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

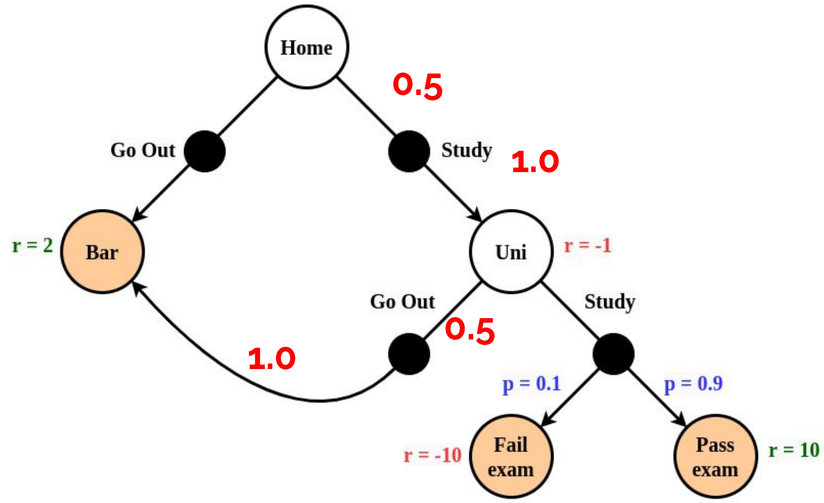


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar		
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

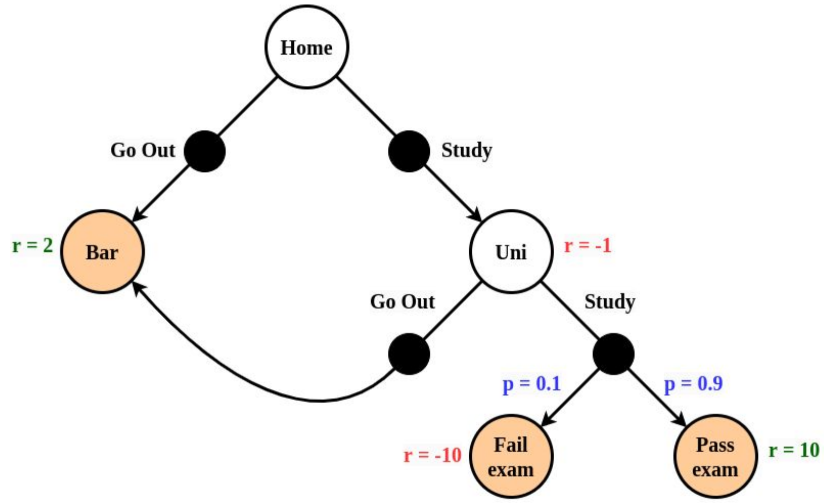


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

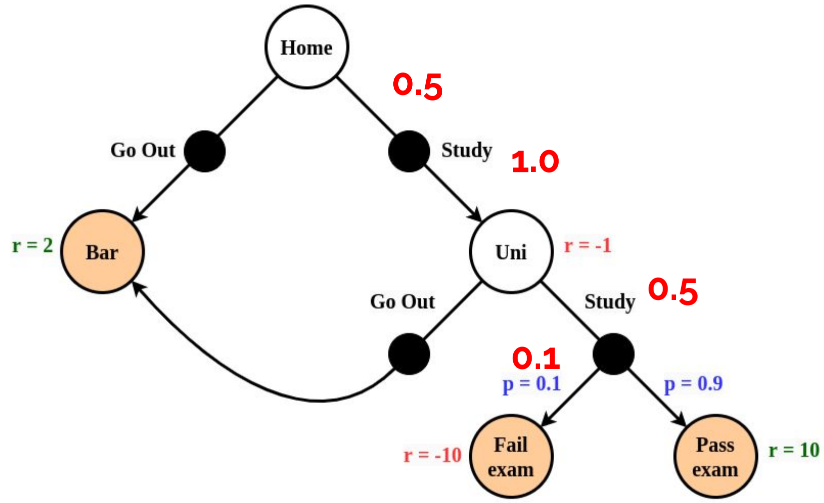


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam		
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

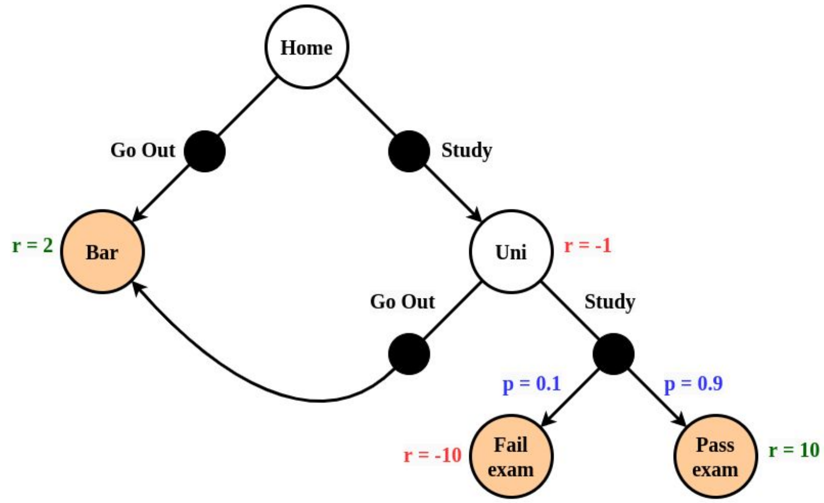


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam	0.025	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.1$
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

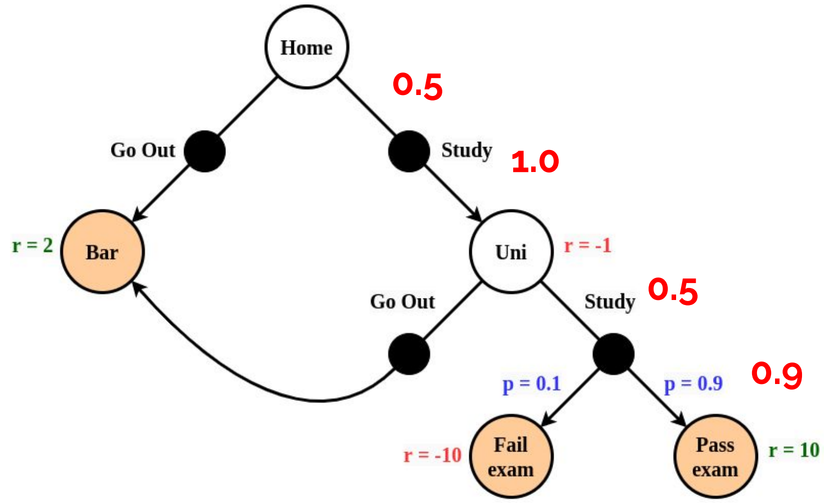


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam	0.025	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.1$
Home - Study - Uni - Study - Pass exam		

Trace probability: Illustration

Assume a *random* policy.

Give the probability of

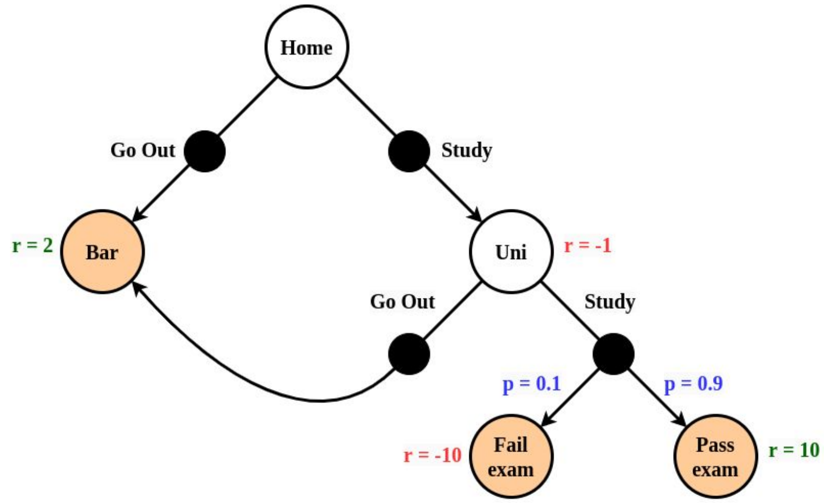


Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam	0.025	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.1$
Home - Study - Uni - Study - Pass exam	0.225	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.9$

Trace probability: Illustration

Assume a *random* policy.

Give the probability of



Distribution adds up to 1.0 again

Trace (τ)	$p(\tau)$	Computation
Home - Go Out - Bar	0.5	$0.5 \cdot 1.0$
Home - Study - Uni - Go Out - Bar	0.25	$0.5 \cdot 1.0 \cdot 0.5 \cdot 1.0$
Home - Study - Uni - Study - Fail exam	0.025	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.1$
Home - Study - Uni - Study - Pass exam	0.225	$0.5 \cdot 1.0 \cdot 0.5 \cdot 0.9$

Part IV:

Return

What makes a trace actually *good*?

Return

Return

Each trace also achieves a certain sum of rewards,

which we call the *cumulative reward* or *return*

Return

Each trace also achieves a certain sum of rewards,

which we call the *cumulative reward* or *return*

$$R(\tau) = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots$$

Return

Each trace also achieves a certain sum of rewards,

which we call the *cumulative reward* or *return*

$$R(\tau) = r_t + \quad r_{t+1} + \quad r_{t+2} + \quad r_{t+3} + \dots$$

The return R of trace τ

Return

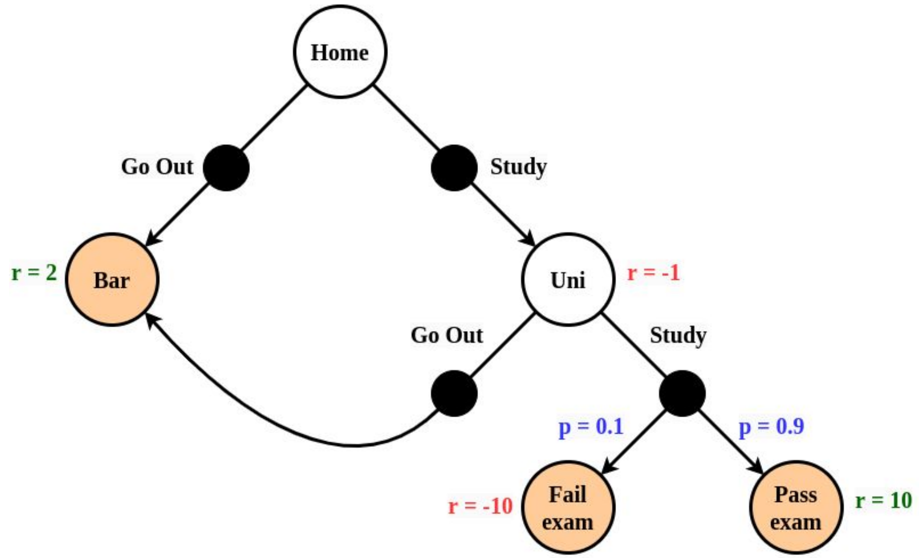
Each trace also achieves a certain sum of rewards,

which we call the *cumulative reward* or *return*

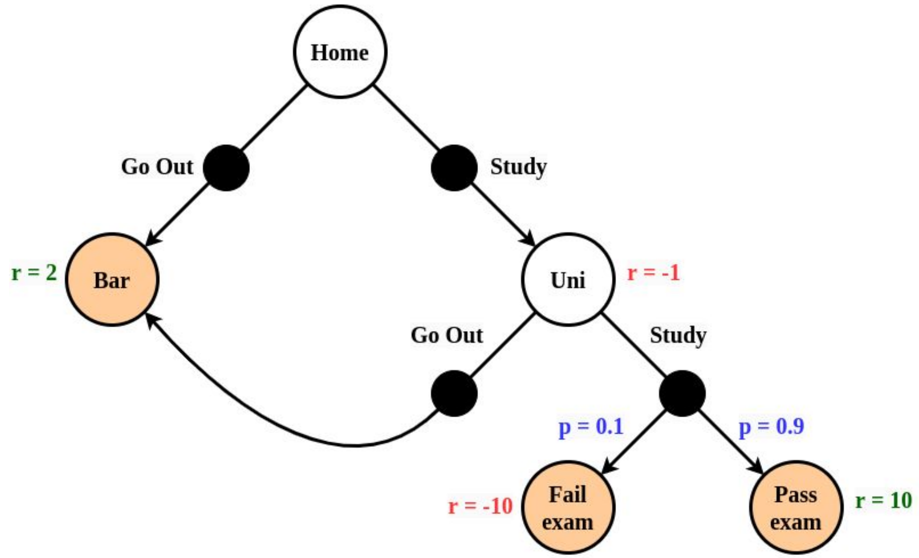
$$R(\tau) = r_t + \quad r_{t+1} + \quad r_{t+2} + \quad r_{t+3} + \ldots$$

The return R of trace τ is the sum of first reward, second reward, third reward, etc.

Return: Illustration



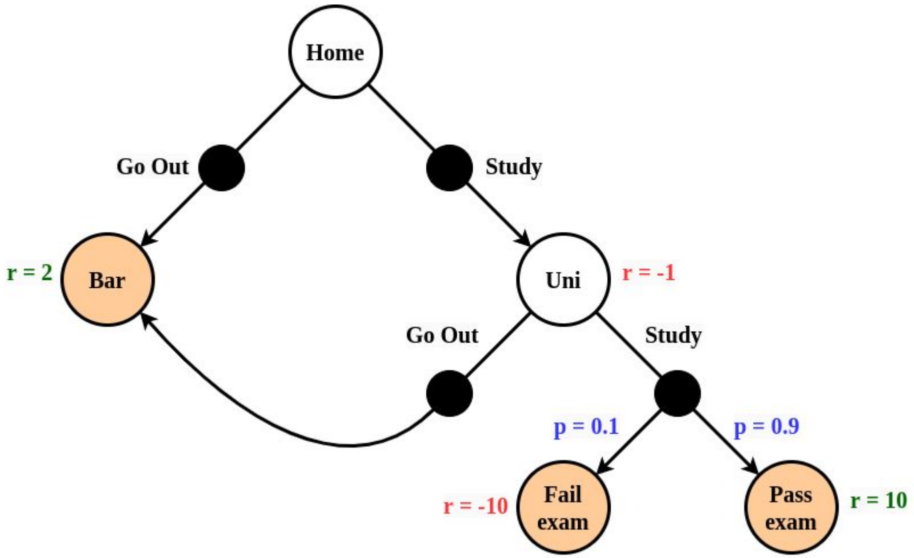
Return: Illustration



Question: What is the return of the trace **[Home, Study, Uni, Study, Fail Exam]**?

Answer:

Return: Illustration



Question: What is the return of the trace **[Home, Study, Uni, Study, Fail Exam]**?

Answer: (-1) + (-10) = -11

Discounted return

Discounted return

We may downweight long-term rewards, which we call the *discounted* return

Discounted return

We may downweight long-term rewards, which we call the *discounted* return

$$R(\tau) = r_t + \quad r_{t+1} + \quad r_{t+2} + \quad r_{t+3} + ...$$

Discounted return

We may downweight long-term rewards, which we call the *discounted* return

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + ...$$

Discounted return

We may downweight long-term rewards, which we call the *discounted* return

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + \ldots$$

Exponentially downweight future rewards

$$\gamma \in [0, 1]$$

Discount factor (γ)

Parameter we set ourselves (part of MDP definition)

Discount factor (γ)

Parameter we set ourselves (part of MDP definition)

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + ...$$

Question: What happens if we use γ=0.0?

Discount factor (γ)

Parameter we set ourselves (part of MDP definition)

$$R(\tau) = r_t$$

Question: What happens if we use γ=0.0?

Answer: Myopic/greedy agent - only cares about immediate reward

Discount factor (γ)

Parameter we set ourselves (part of MDP definition)

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + ...$$

Question: What happens if we use γ=1.0?

Discount factor (γ)

Parameter we set ourselves (part of MDP definition)

$$R(\tau) = r_t + \quad r_{t+1} + \quad r_{t+2} + \quad r_{t+3} + \ldots$$

Question: What happens if we use γ=1.0?

Answer: Long-term agent - rewards from all timesteps contribute equally

Spectrum of discount value choices

Spectrum of discount value choices

Spectrum of discount value choices

We ideally want **γ**=1.0, i.e., full sequential decision-making

Spectrum of discount value choices

In practice we may set it slightly lower, e.g., γ=0.99, for

Spectrum of discount value choices

In practice we may set it slightly lower, e.g., **γ**=0.99, for

- Numerical stability (ensure the sum of rewards stays bounded/finite)

Spectrum of discount value choices

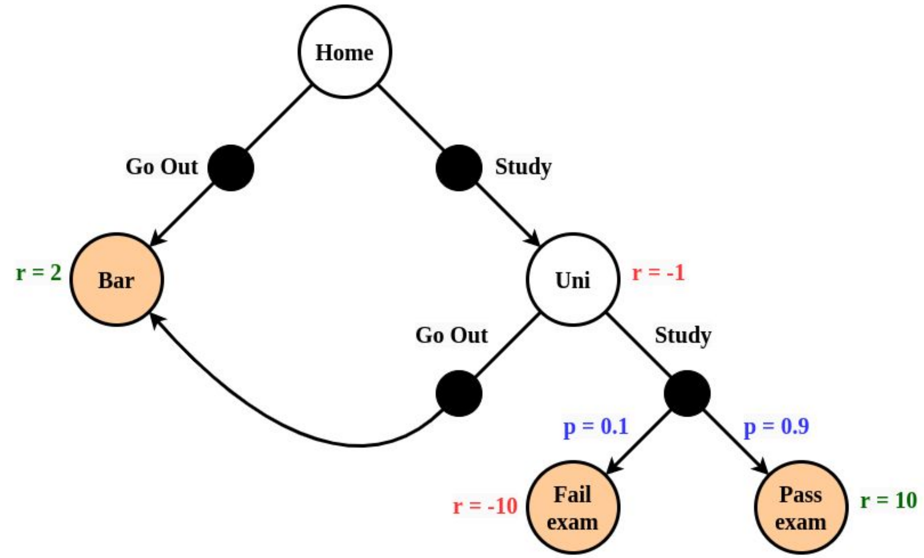
In practice we may set it slightly lower, e.g., γ=0.99, for

- Numerical stability (ensure the sum of rewards stays bounded/finite)
- Implicitly enforcing an agent to take as little steps a possible (since every extra step discounts the next obtained rewards - useful when there are transitions with r=0)

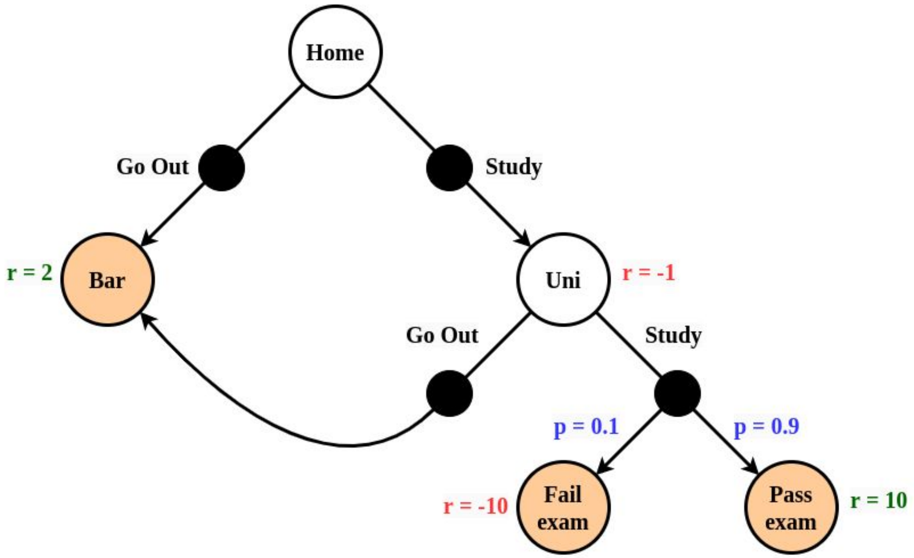
Spectrum of discount value choices

For this course we will mostly fix it at **γ**=1.0

Discounted return: Illustration



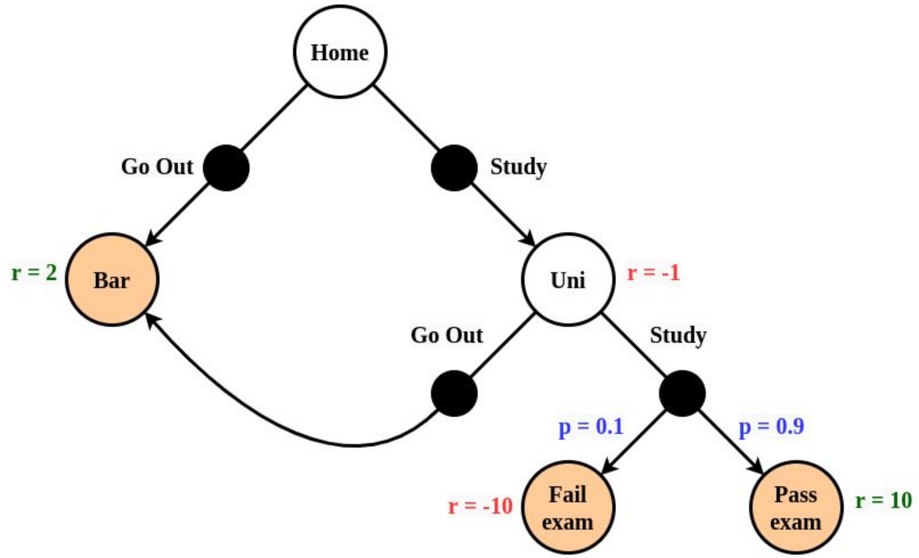
Discounted return: Illustration



Use γ=0.9

Question: What is the discounted return of the trace **[Home, Study, Uni, Go Out, Bar]**?

Discounted return: Illustration



Use γ=0.9

Question: What is the discounted return of the trace **[Home, Study, Uni, Go Out, Bar]**?

Answer: $-1 + 0.9 \cdot 2.0 = 0.8$

(Note the difference - the second reward now has smaller weight than the first)

Trace Horizon

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + ...$$

Trace Horizon

$$
\begin{aligned}
R(\tau) &= r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + \ldots \\
&= \sum_{i=0}^{\infty} (\gamma)^i \cdot r_{t+i}
\end{aligned}
$$

(sum notation)

Trace Horizon

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + \ldots$$
$$= \sum_{i=0}^{\infty} (\gamma)^i \cdot r_{t+i}$$

Infinite-horizon return:

'we keep summing rewards unless we reach a terminal state'

Trace Horizon

$$R(\tau) = r_t + (\gamma) \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + (\gamma)^3 \cdot r_{t+3} + \ldots$$

$$= \sum_{i=0}^{\infty} (\gamma)^i \cdot r_{t+i}$$

Infinite-horizon return:

'we keep summing rewards unless we reach a terminal state'

(there are also *finite-horizon* MDPs, but we don't cover them)

Part VI:

Value

Value

Cumulative reward (= return)

$$r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \ldots$$

Value

~~Cumulative reward (= return)~~

Expected cumulative reward

$$\mathbb{E}_{\tau \sim p^{\pi}(\tau)}\left[r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \dots \qquad \right]$$

Value

~~Cumulative reward (= return)~~

Expected cumulative reward

$$\mathbb{E}_{\tau \sim p^{\pi}(\tau)} \left[r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \ldots \right]$$

What sum of rewards do we expect *on average* for a particular policy

Value

~~Cumulative reward (= return)~~

Expected cumulative reward

$$v^{\pi}(s) = \mathbb{E}_{\tau \sim p^{\pi}(\tau)}\left[r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \ldots | s_t = s\right]$$

Value

~~Cumulative reward (= return)~~

Expected cumulative reward

$$v^\pi(s) = \mathbb{E}_{\tau \sim p^\pi(\tau)}\left[r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \ldots | s_t = s\right]$$

We call this the *value* of a state s:

Value

~~Cumulative reward (= return)~~

Expected cumulative reward

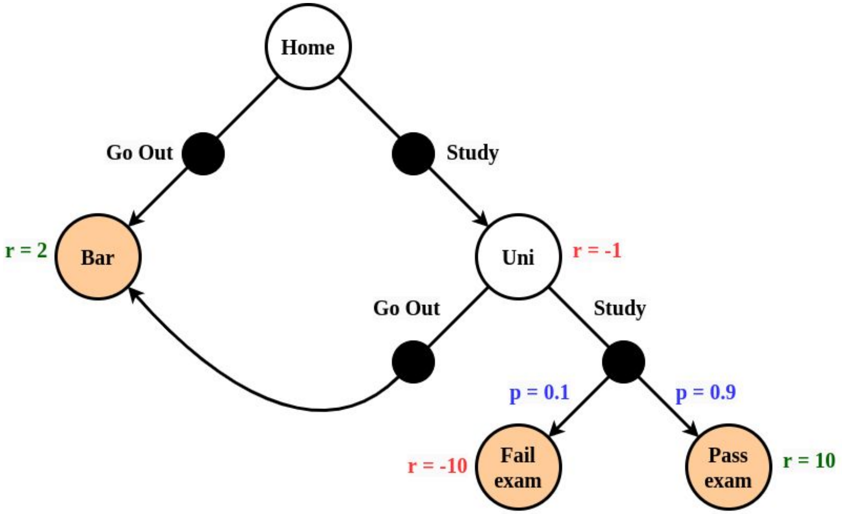
$$v^{\pi}(s) = \mathbb{E}_{\tau \sim p^{\pi}(\tau)}\left[r_t + \gamma \cdot r_{t+1} + (\gamma)^2 \cdot r_{t+2} + \ldots | s_t = s\right]$$

We call this the *value* of a state s:

Given a certain policy, how much total reward do we expect to get from s in the future

State value: Illustration

State value: Illustration

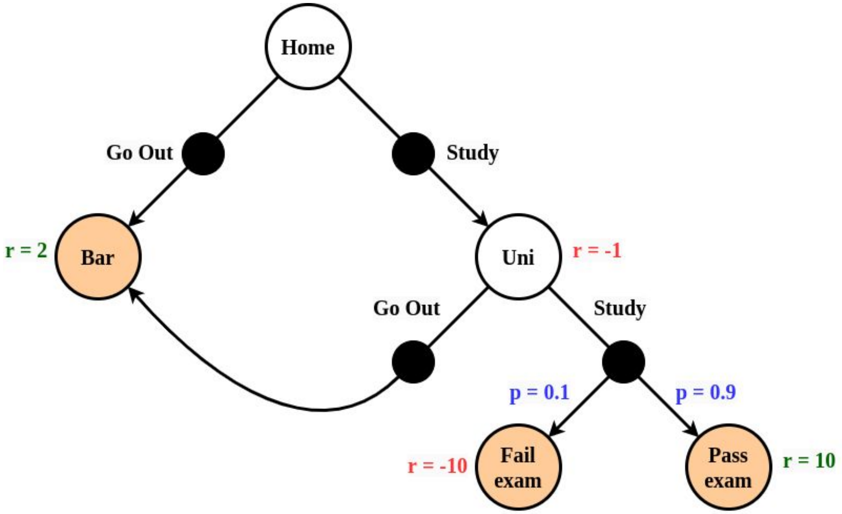


State value: Illustration

Question: Assume a *random* policy.

Can we compute v(Home)?

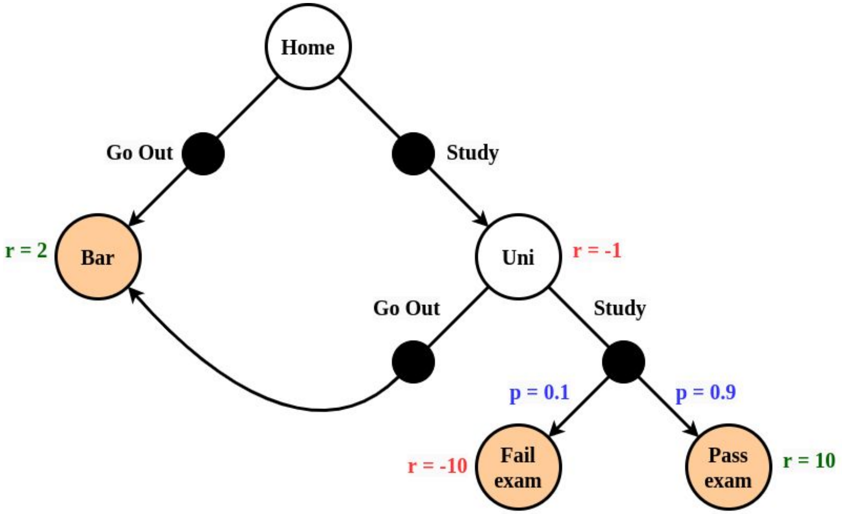
(Note: already computed this at the lecture start)



State value: Illustration

Question: Assume a *random* policy.

Can we compute v(Home)?

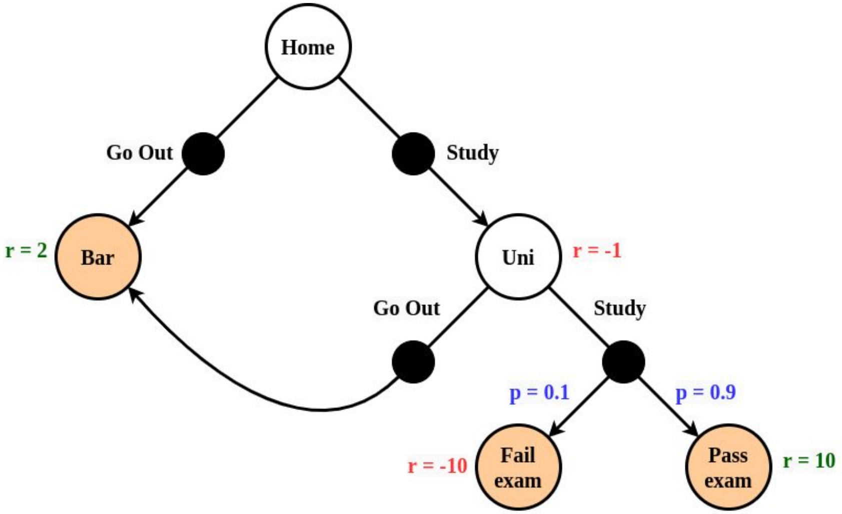


Trace (τ)
Home - Go Out - Bar
Home - Study - Uni - Go Out - Bar
Home - Study - Uni - Study - Fail exam
Home - Study - Uni - Study - Pass exam

List all possible traces from Home

State value: Illustration

Question: Assume a *random* policy.
Can we compute v(Home)?



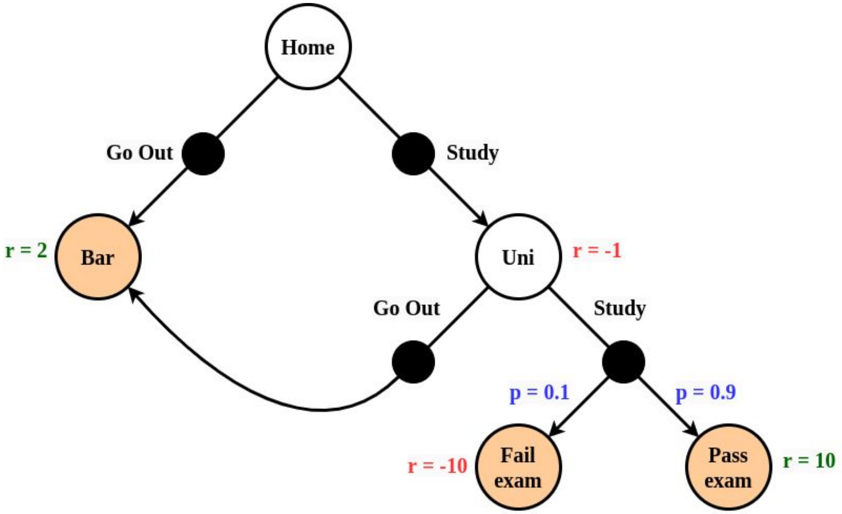
Trace (τ)	$p(\tau)$
Home - Go Out - Bar	0.5
Home - Study - Uni - Go Out - Bar	0.25
Home - Study - Uni - Study - Fail exam	0.025
Home - Study - Uni - Study - Pass exam	0.225

List all possible traces from Home with their probability

State value: Illustration

Question: Assume a *random* policy.

Can we compute v(Home)?



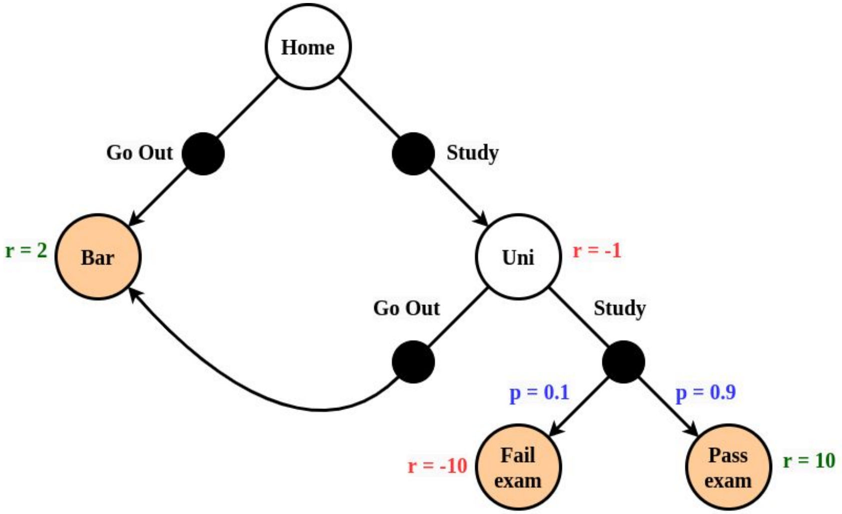
Trace (τ)	$p(\tau)$	$R(\tau)$
Home - Go Out - Bar	0.5	2.0
Home - Study - Uni - Go Out - Bar	0.25	1.0
Home - Study - Uni - Study - Fail exam	0.025	-11
Home - Study - Uni - Study - Pass exam	0.225	9

List all possible traces from Home with their probability and obtained return (sum of rewards)

State value: Illustration

Question: Assume a *random* policy.

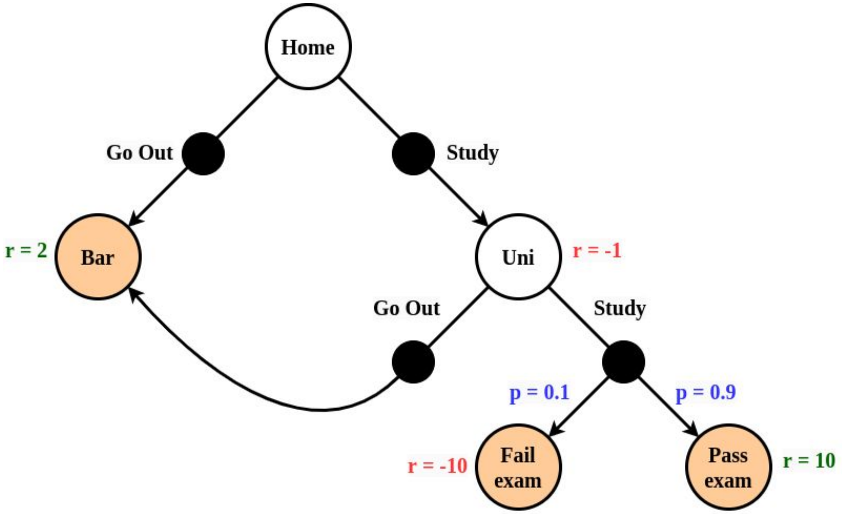
Can we compute v(Home)?



Trace (τ)	$p(\tau)$	$R(\tau)$	$p(\tau) \cdot R(\tau)$
Home - Go Out - Bar	0.5	2.0	1.0
Home - Study - Uni - Go Out - Bar	0.25	1.0	0.25
Home - Study - Uni - Study - Fail exam	0.025	-11	-0.275
Home - Study - Uni - Study - Pass exam	0.225	9	2.025

Reweight each return based on its probability

State value: Illustration



Question: Assume a *random* policy.

Can we compute v(Home)?

Answer: v(Home) = 3.0

(Matches our earlier computation!)

Trace (τ)	$p(\tau)$	$R(\tau)$	$p(\tau) \cdot R(\tau)$
Home - Go Out - Bar	0.5	2.0	1.0
Home - Study - Uni - Go Out - Bar	0.25	1.0	0.25
Home - Study - Uni - Study - Fail exam	0.025	-11	-0.275
Home - Study - Uni - Study - Pass exam	0.225	9	2.025
			3.0

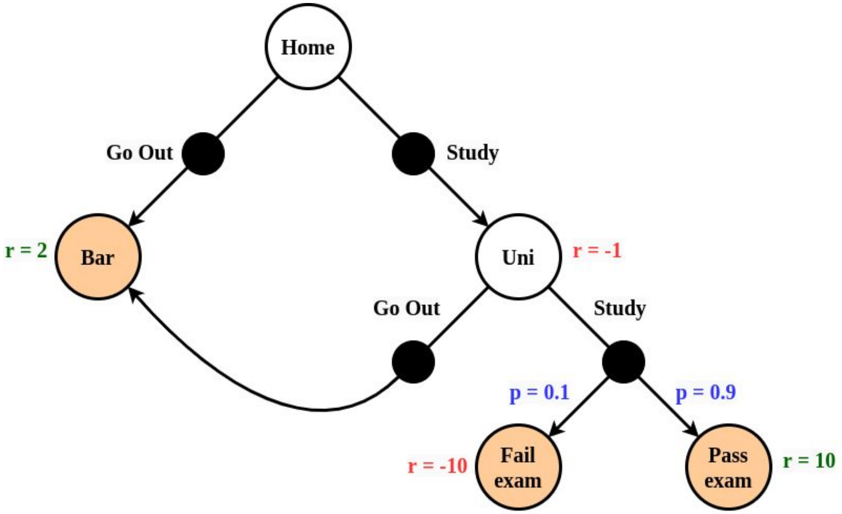
Reweight each return based on its probability and add up to get the value (expected return)

State value: Illustration

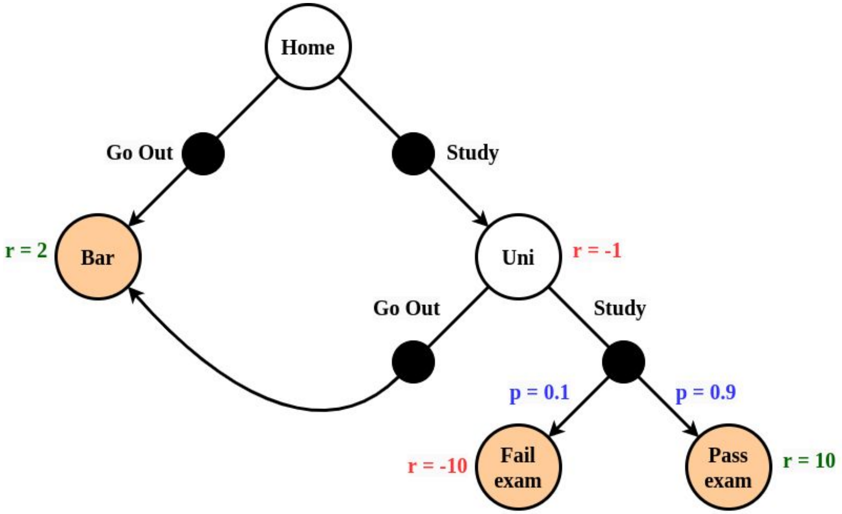
Question: Assume a *random* policy.

Can do the same for all other states.

So what is v(Bar)?



State value: Illustration



Question: Assume a *random* policy.

Can do the same for all other states.

So what is v(Bar)?

Answer: v(Bar) =0.0

The value of a terminal state is always 0.0, since we can never obtain any reward from it!

Value function

Value function

The state value V(s) is a *function*: every state has its own value given a certain policy.

Value function

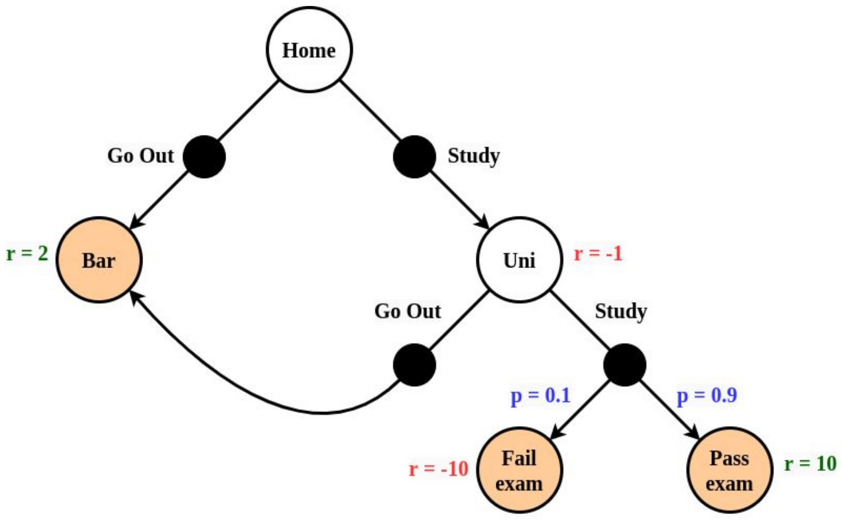
The state value V(s) is a *function*: every state has its own value given a certain policy.

Can represent this in memory as a table of size |S|

Value function

The state value V(s) is a *function*: every state has its own value given a certain policy.

Can represent this in memory as a table of size |S|

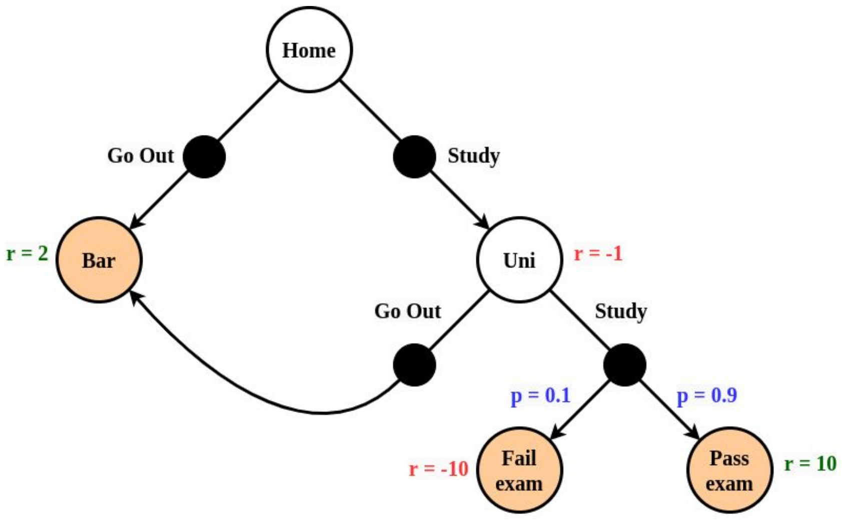


Value function

The state value V(s) is a *function*: every state has its own value given a certain policy.

Can represent this in memory as a table of size |S|

s	$v^{\text{random}}(s)$
Home	3.0
Uni	5.0
Bar	0.0
Pass exam	0.0
Fail exam	0.0

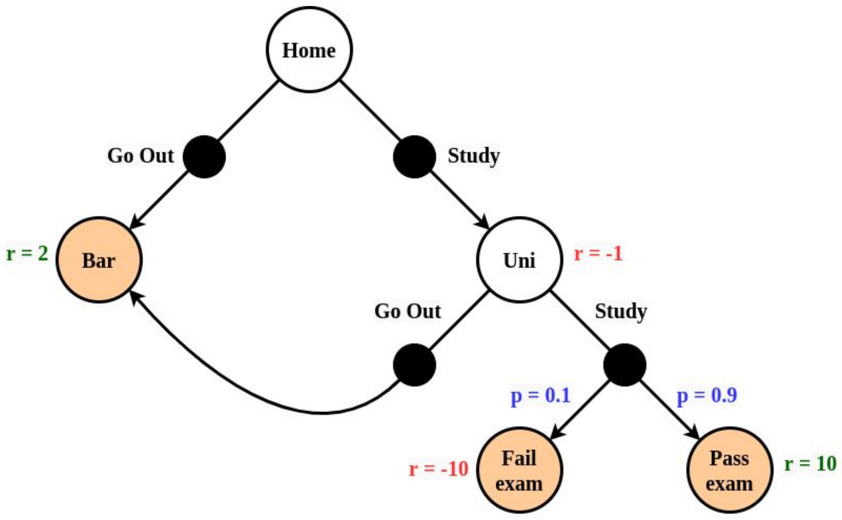


Value function

The state value V(s) is a *function*: every state has its own value given a certain policy.

Can represent this in memory as a table of size |S|

s	$v^{\text{random}}(s)$
Home	3.0
Uni	5.0
Bar	0.0
Pass exam	0.0
Fail exam	0.0



Check v(Uni) yourself by listing all possible traces from Uni, their probability and return

State-action value

State-action value

We can equally define the value of a *state-action pair*

State-action value

We can equally define the value of a *state-action pair*

$$q^\pi(s,a) = \mathbb{E}_{\tau\sim p^\pi(\tau)}\left[R(\tau)|s_t = s, a_t = a\right]$$

State-action value

We can equally define the value of a *state-action pair*

$$q^{\pi}(s,a) = \mathbb{E}_{\tau \sim p^{\pi}(\tau)}\left[R(\tau) | s_t = s, a_t = a\right]$$

Exact same principle as before, but now we also condition on the first action we take

State-action value function

State-action value function

The state value q(s,a) is a *function*: every state-action has its own value given a certain policy.

State-action value function

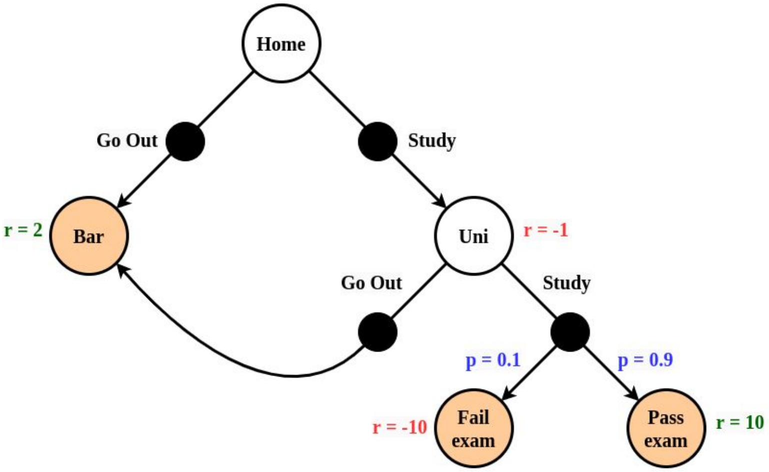
The state value $q(s,a)$ is a *function*: every state-action has its own value given a certain policy.

Can represent this in memory as a table of size $|S| \times |A|$

State-action value function

The state value q(s,a) is a *function*: every state-action has its own value given a certain policy.

Can represent this in memory as a table of size |S| x |A|

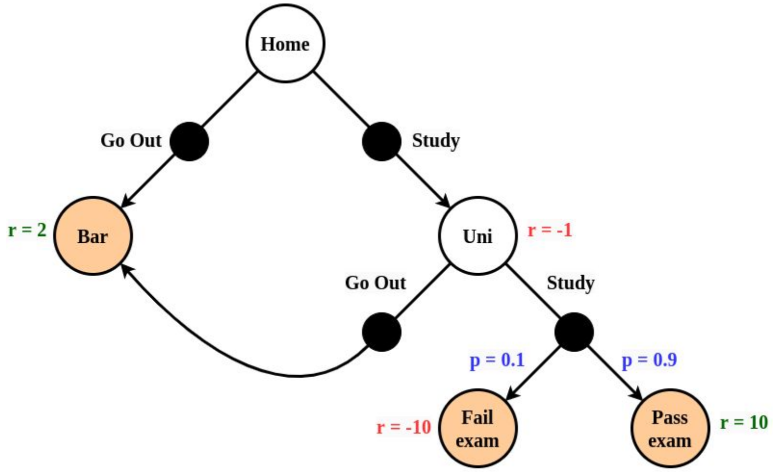


State-action value function

The state value q(s,a) is a *function*: every state-action has its own value given a certain policy.

Can represent this in memory as a table of size |S| x |A|

s	$q^{\text{random}}(s,a)$ Go out	Study
Home	2.0	4.0
Uni	2.0	8.0
Bar	0.0	0.0
Pass exam	0.0	0.0
Fail exam	0.0	0.0



Optimal value & policy

Optimal value function

Optimal value function

The best value we can achieve from every state/state-action

Optimal value function

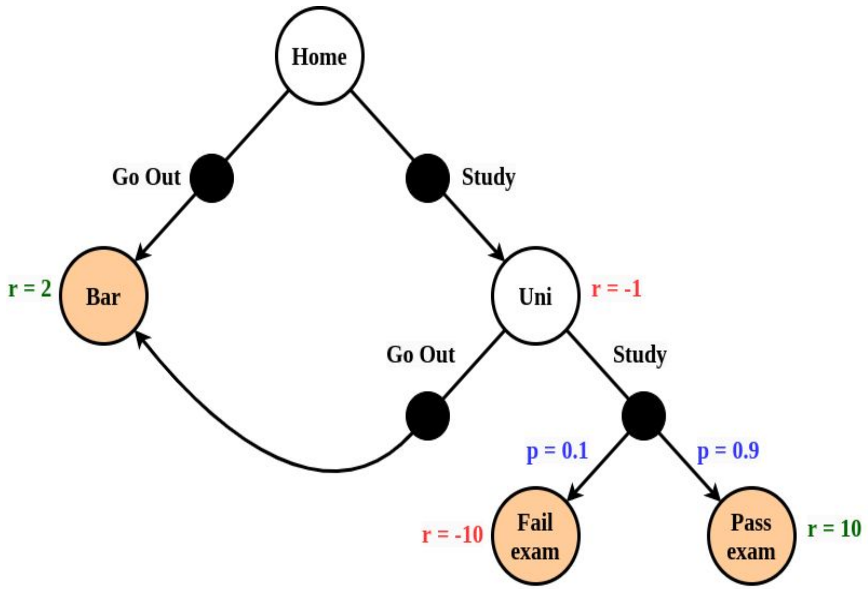
The best value we can achieve from every state/state-action

Notation: $v^\star(s)$, $q^\star(s, a)$

Optimal value function

The best value we can achieve from every state/state-action

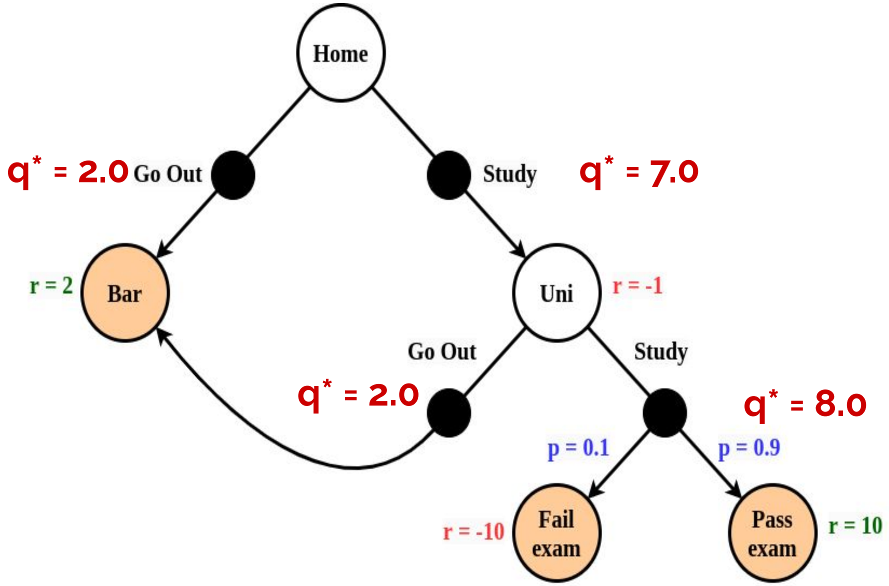
Notation: $v^{\star}(s)$, $q^{\star}(s, a)$



Optimal value function

The best value we can achieve from every state/state-action

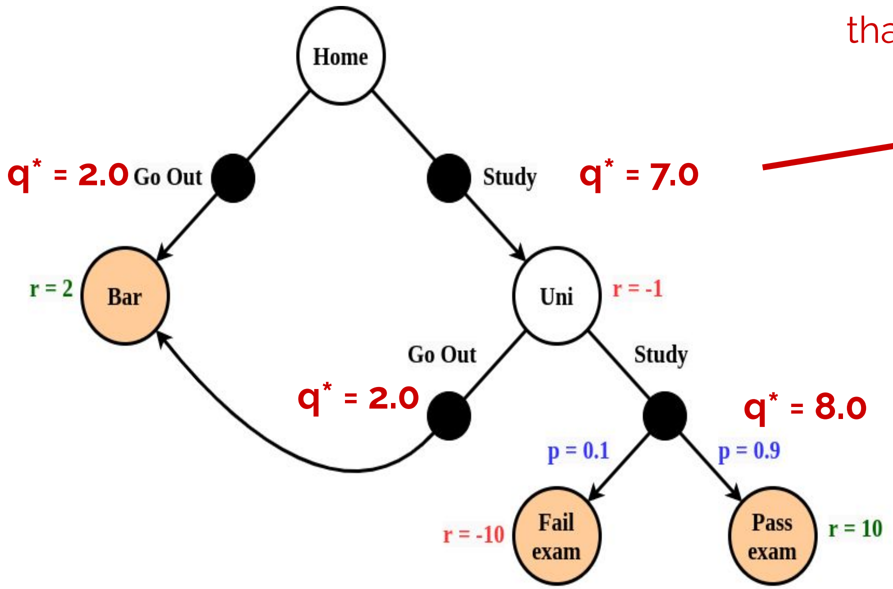
Notation: $v^\star(s)$, $q^\star(s,a)$



Optimal value function

The best value we can achieve from every state/state-action

Notation: $v^{\star}(s)$, $q^{\star}(s,a)$

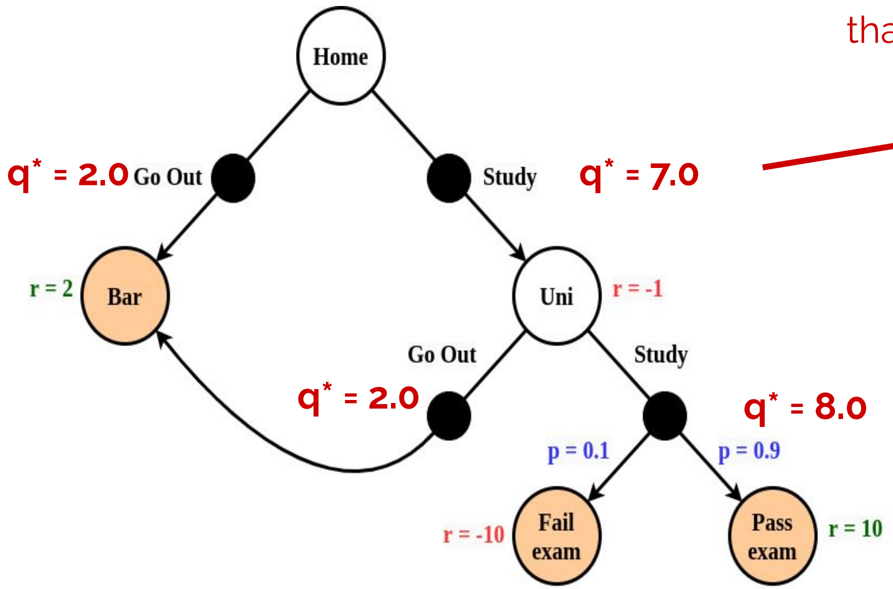


It is impossible to achieve more than 7.0 on average from (Home, Study)

Optimal value function

The best value we can achieve from every state/state-action

Notation: $v^{\star}(s)$, $q^{\star}(s, a)$



It is impossible to achieve more than 7.0 on average from (Home, Study)

Each MDP has only one optimal value function = best we can achieve with optimal behaviour

Optimal policy

Optimal policy

A policy that obtains the optimal value function.

Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

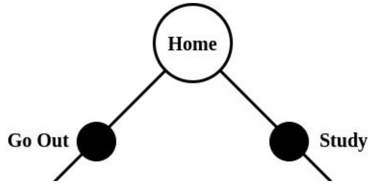
Intuition: In principle the greedy (max) policy

Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Intuition: In principle the greedy (max) policy

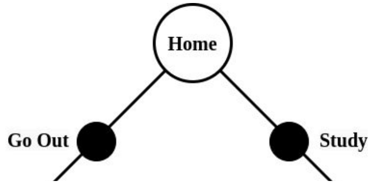


Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Intuition: In principle the greedy (max) policy

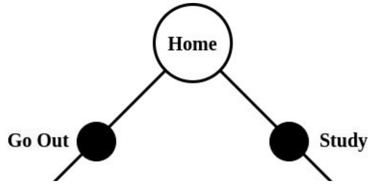
q^*=2  q^*=7

Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Intuition: In principle the greedy (max) policy



$q^*=2$ $q^*=7$

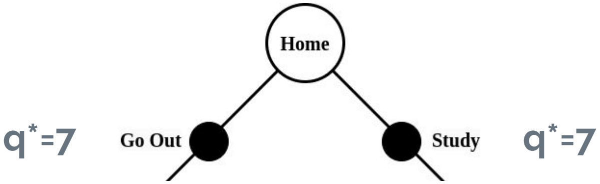
$\pi^*(\text{Home})=\text{Study}$

Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Intuition: In principle the greedy (max) policy



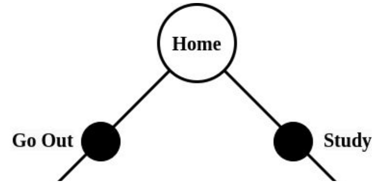
Optimal policy

A policy that obtains the optimal value function.

Notation: $\pi^\star$

Intuition: In principle the greedy (max) policy

q^*=7



q^*=7

π^*(Home)=Study

or

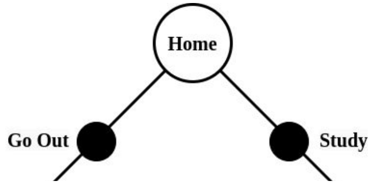
π^*(Home)=Go Out

Optimal policy

A policy that obtains the optimal value function.

Notation: π^*

Intuition: In principle the greedy (max) policy

$q^*=7$  $q^*=7$

π^*(Home)=Study

or

π^*(Home)=Go Out

When we have ties we can arbitrarily break them

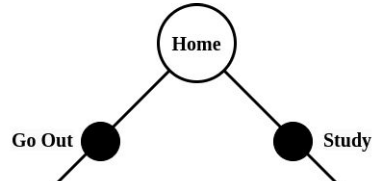
Optimal policy

A policy that obtains the optimal value function.

Notation: π^*

Intuition: In principle the greedy (max) policy

$q^*=7$



$q^*=7$

π^*(Home)=Study

or

π^*(Home)=Go Out

When we have ties we can arbitrarily break them

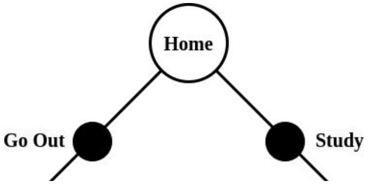
(i.e., therefore there can theoretically be more than one optimal policy)

Optimal policy

A policy that obtains the optimal value function.

Notation: π^*

Intuition: In principle the greedy (max) policy

q^*=7  q^*=7

π^*(Home)=Study

or

π^*(Home)=Go Out

When we have ties we can arbitrarily break them

(i.e., therefore there can theoretically be more than one optimal policy)

(but they all have the same optimal value function)

Summary

Summary

- Policy: We act in the MDP according to a policy $\pi(a|s)$

Summary

- Policy: We act in the MDP according to a policy
  $\pi(a|s)$

- Trace: When we act with the policy we induce a sequence of (s,a,r) pairs, i..e, a trace.
  τ

Summary

- Policy: We act in the MDP according to a policy
  $\pi(a|s)$

- Trace: When we act with the policy we induce a sequence of (s,a,r) pairs, i..e, a trace.
  τ

- Return: Each such trace has a certain (discounted) cumulative reward, i.e., a return.
  $R(\tau)$

Summary

- Policy: We act in the MDP according to a policy
  $\pi(a|s)$

- Trace: When we act with the policy we induce a sequence of (s,a,r) pairs, i..e, a trace.
  τ

- Return: Each such trace has a certain (discounted) cumulative reward, i.e., a return.
  $R(\tau)$

- Value: Given a policy, each state(-action) has an average/expected return, i.e., a value.
  $v^{\pi}(s)$, $q^{\pi}(s,a)$

Summary

- Policy: We act in the MDP according to a policy
  $\pi(a|s)$

- Trace: When we act with the policy we induce a sequence of (s,a,r) pairs, i..e, a trace.
  τ

- Return: Each such trace has a certain (discounted) cumulative reward, i.e., a return.
  $R(\tau)$

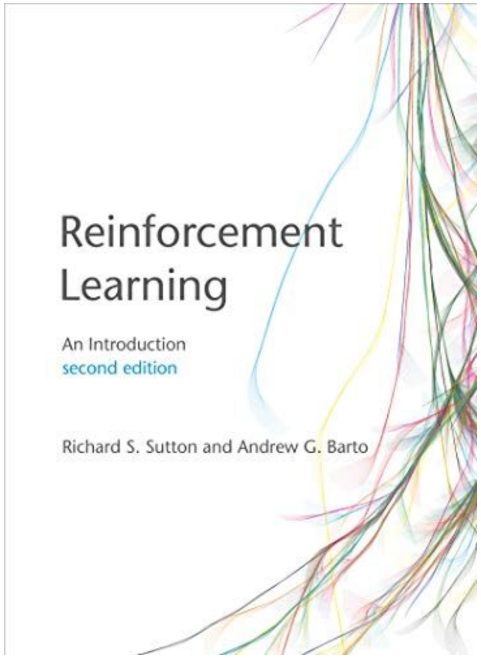
- Value: Given a policy, each state(-action) has an average/expected return, i.e., a value.
  $v^{\pi}(s)$, $q^{\pi}(s,a)$

- Optimal value/policy: There is only one optimal value function, with a greedy/max policy.
  $v^{*}(s)$, $q^{*}(s,a)$, $\pi^{*}(s)$

At Home

At Home

Read Sutton & Barto, Chapter 3



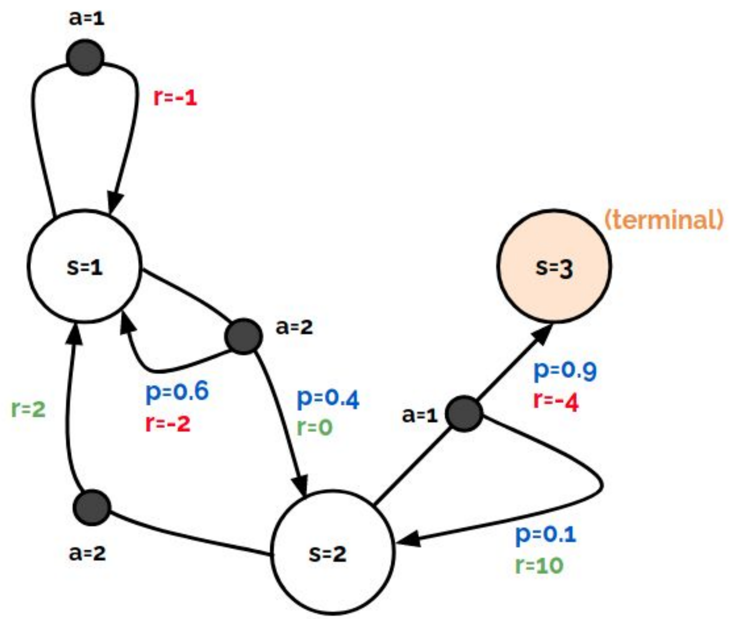
At Home (by hand)

At Home (by hand)

1. Draw your own MDP

At Home (by hand)

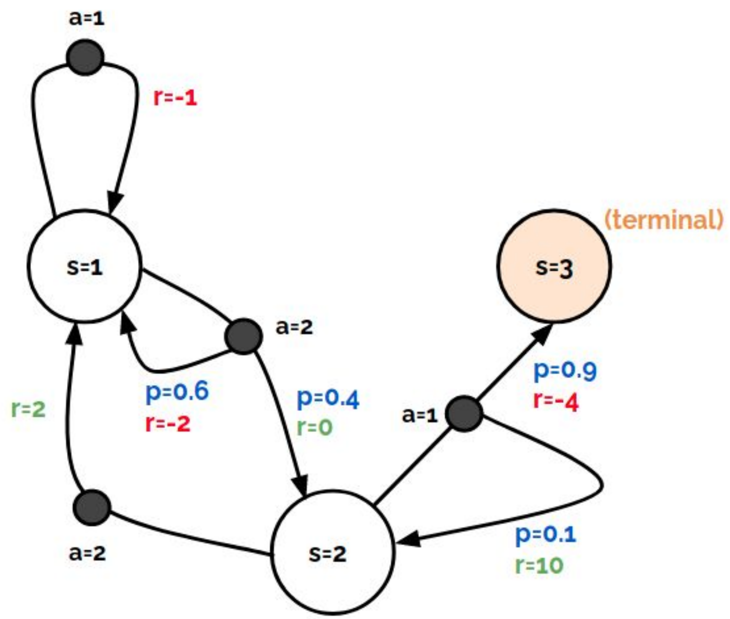
1. Draw your own MDP, e.g.:



At Home (by hand)

1. Draw your own MDP, e.g.:

2. Define a policy in this MDP

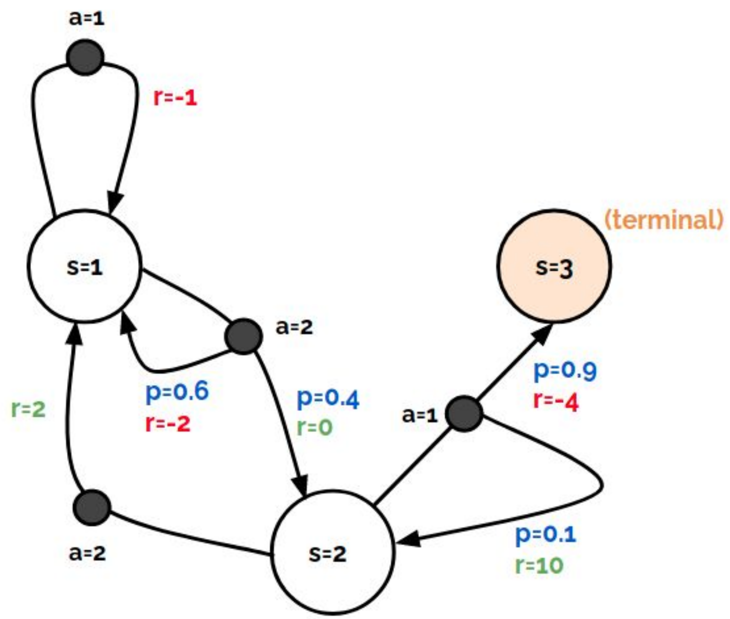


At Home (by hand)

1. Draw your own MDP, e.g.:

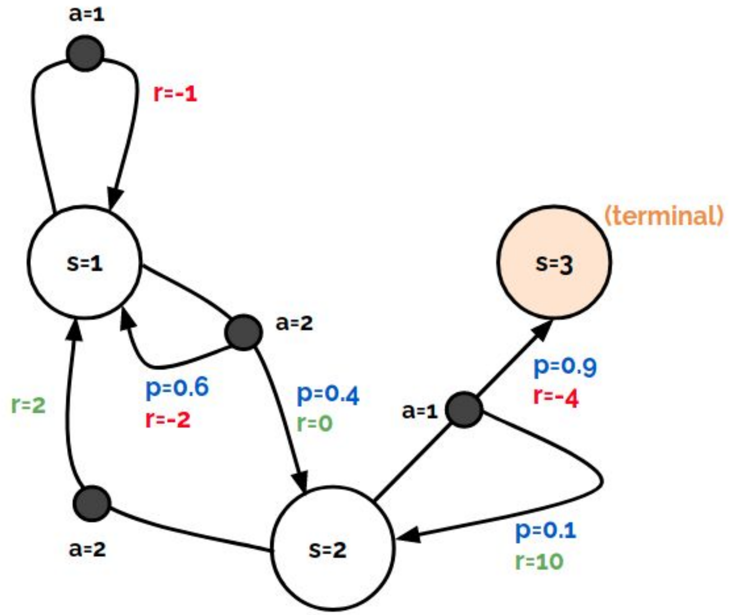
2. Define a policy in this MDP

3. Compute a trace distribution



At Home (by hand)

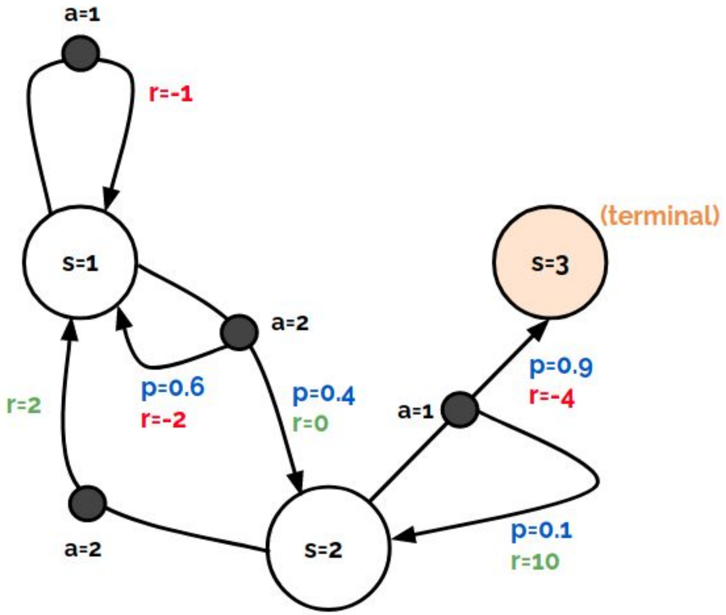
1. Draw your own MDP, e.g.:



2. Define a policy in this MDP
3. Compute a trace distribution
4. Compute the returns of these traces

At Home (by hand)

1. Draw your own MDP, e.g.:
2. Define a policy in this MDP
3. Compute a trace distribution
4. Compute the returns of these traces
5. Compute v(s) and q(s,a) for your policy



At Home (code)

At Home (code)

Go to Colab: http://tiny.cc/ntbjvz

At Home (code)

Go to Colab: http://tiny.cc/ntbjvz

1. MDP: Definition

Let's define the Markov Decision Process shown in the figure, and then sample a few episodes.

Code: Study MDP definition

```
[ ] #@title Code: Study MDP definition

    import numpy as np
    import random

    class Study():

        def __init__(self):

            # Define the state space
            self.states = ('Home','Bar','Uni','Fail exam','Pass exam')
            self.n_states = len(self.states)
            self.state_to_index = {'Home':0,
                                   'Bar':1,
                                   'Uni':2,
                                   'Fail exam': 3,
                                   'Pass exam':4
            }
```

At Home (code)

Go to Colab: http://tiny.cc/ntbjvz

1. MDP: Definition

Let's define the Markov Decision Process shown in the figure, and then sample a few episodes.

Code: Study MDP definition

```
[ ] #@title Code: Study MDP definition

    import numpy as np
    import random

    class Study():

        def __init__(self):

            # Define the state space
            self.states = ('Home','Bar','Uni','Fail exam','Pass exam')
            self.n_states = len(self.states)
            self.state_to_index = {'Home':0,
                                   'Bar':1,
                                   'Uni':2,
                                   'Fail exam': 3,
                                   'Pass exam':4
            }
```

Interactive code for lecture material: play around and make sure you understand!

Next Week

Next Week

1. *Bellman Equation*

   Recursive relation between the state(-action) values

Next Week

1. *Bellman Equation*

   Recursive relation between the state(-action) values

2. *Dynamic programming*

   Use this recursive relation to efficiently find the optimal value function & policy

Questions?