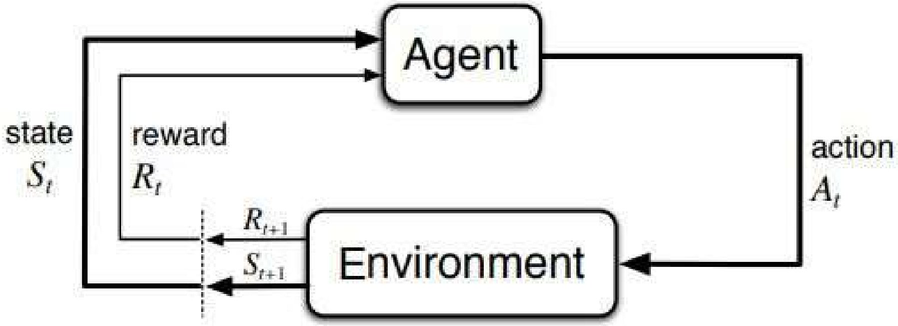
Reinforcement Learning Model

How to formulate RL problems mathematically?

- Solving reward-based problems
- Agent : learner that make intelligent decisions
- Environment: real-world environment with which the agent interacts as part of its operation.
- State: represents the current 'state of the world' at any point.
- Action: actions that the agent takes to interact with the environment
- Reward: positive or negative reinforcement that the agent receives from the environment as a result of its actions.
- Markov Decision process

How to formulate RL problems mathematically?



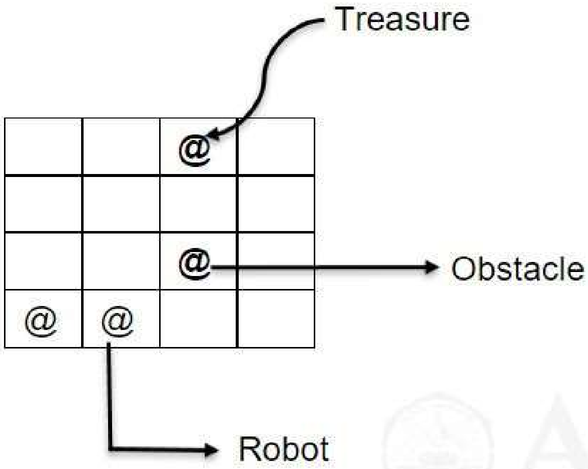
- Agent interacts with the environment .
- Environment provides rewards
- At each stage t, the agent selects an action that transforms the state of the environment from st to st +1

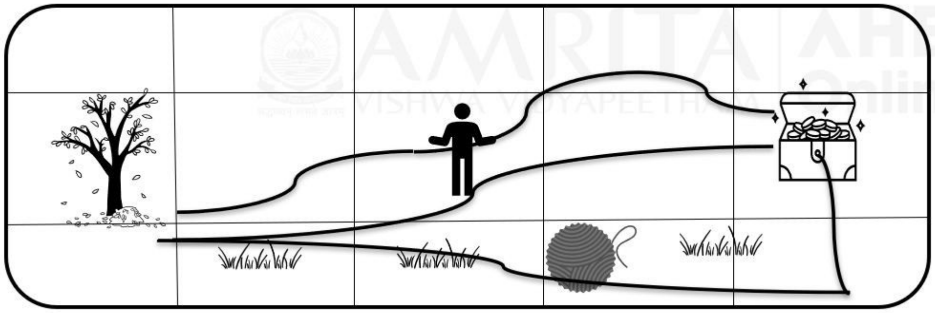
This Photo by Unknown author is licensed under CC BY-SA.

Markov Decision Process

- Markov decision processes (MDP) is the mathematical framework to describe an environment for reinforcement learning
- MDP includes a set of finite environment states, set of possible actions in each state, a reward function and a transition model of the environment
- Actions influence not just immediate rewards, but also subsequent situations
- Fully observable environment

Example 1: Robot walking in grid world





- Actions: L / R / U / D
- States: grid positions
- Reward: Treasure state +10 , obstacle state -10 and –1 for all other states
- Transition probability : uniform random policy(0.25 for all actions)

Example 2: Pick and Place Robot

- Actions: voltages applied to each motor at each joint
- States: sensor readings of joint angles and velocities.
- Reward: +10 for each object successfully picked up and placed. -10 for jerk

This Photo by Unknown author is licensed under CC BY-SA.

Markov Property

- Memoryless property of a stochastic process
- Future states depends only upon the present state given the present

"Future is independent of the past given the present"

Mathematically we can express this statement as:

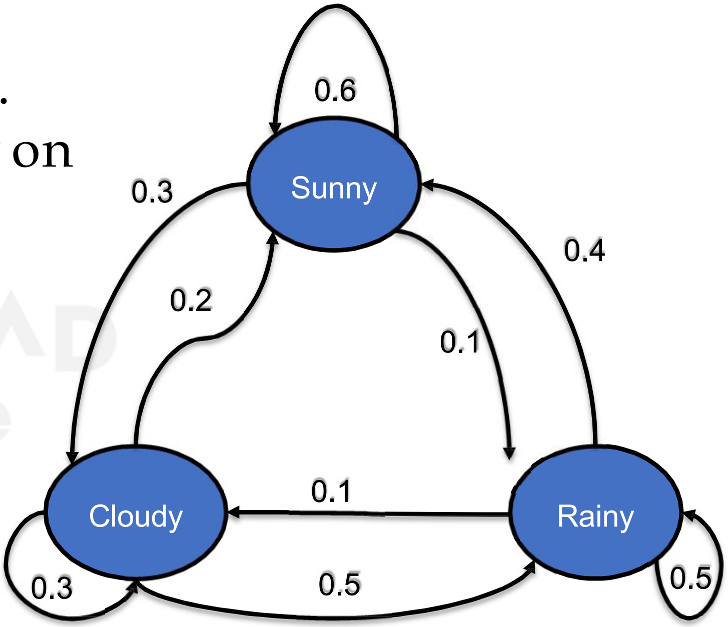
$$P\ [S_{t+1} \mid S\ ] = P\ [S_{t+1} \mid S_1, S_2 \ldots \ldots., S_t]$$

https://en.wikipedia.org/wiki/Markov_property

Markov Chain Example

- Finite number of states
- Each time period, it moves from one state to another.
- The probabilities of transition to a state depend only on the probabilities associated with our current state.

Weather forecast



Weather today

https://blogs.warwick.ac.uk/ctraynor/entry/weather_forecast_with/

State Transition Probability Matrix

For a current state s and next state s` , the state transition probability is defined by

$$P_{ss'} = Prb[S_{t+1} = s' | S_t = s]$$

State transition matrix P defines transition probabilities from all states s to all successor states s`

$$P= \begin{bmatrix} p11\,p12 & \ldots & p1n \\ \ldots & \ldots & \ldots \\ pn1\,pn2 & \ldots & pnn \end{bmatrix}$$

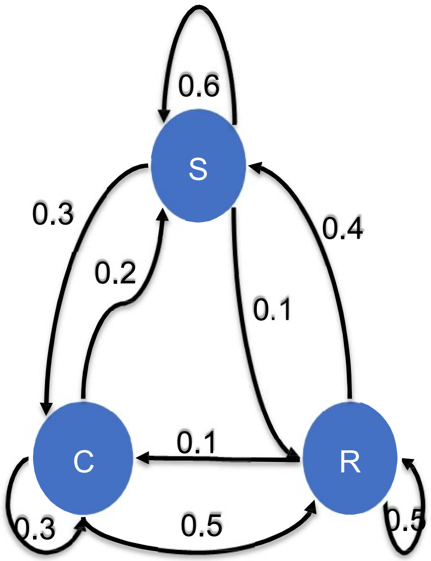
Markov Process

Markov Process is a memoryless random process, with the Markov Property.

A Markov Process or Markov Chain is a tuple < S, P >

- S is a (finite) set of states
- P is a state transition probability matrix,

$$P_{ss'} = Prb\,[S_{t+1} = s^{'} \mid S_t = s]$$



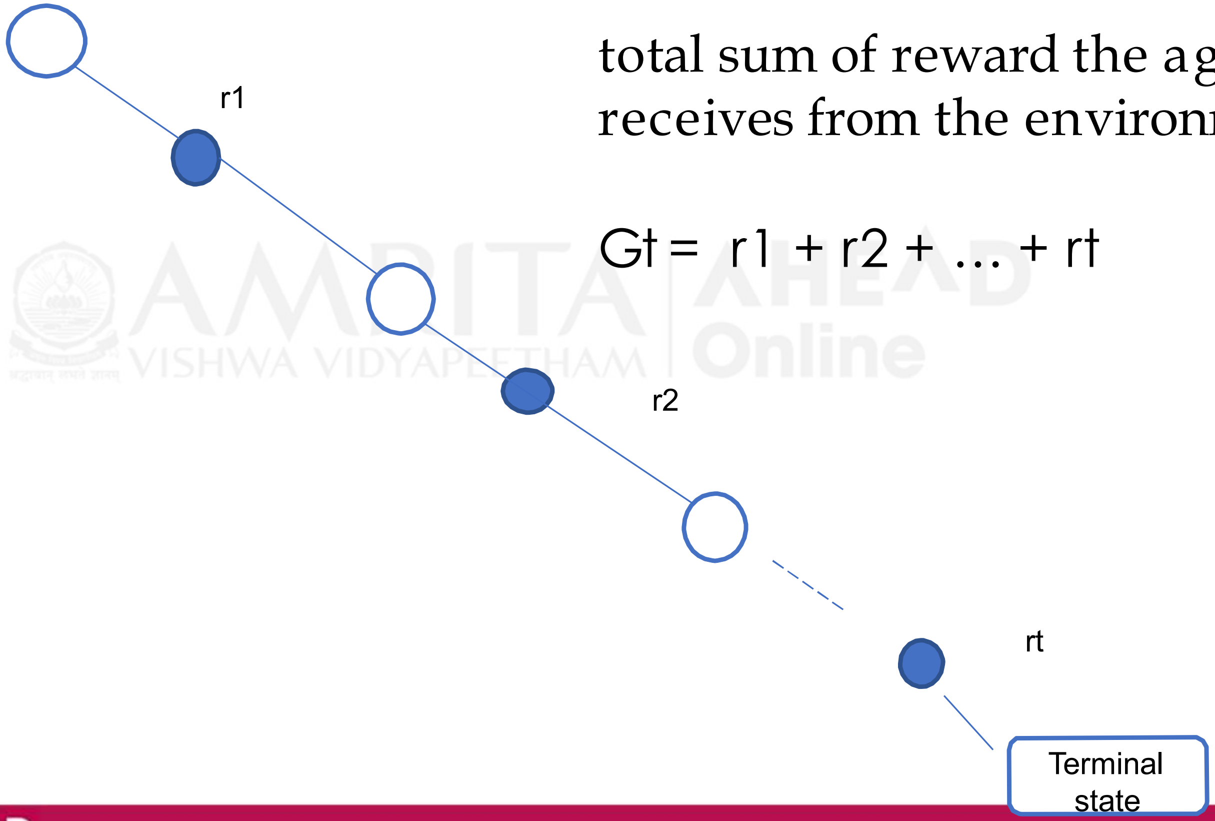
1.Weather Forecasting: Markov models are used to predict weather patterns based on historical data. The state of the system is defined by the weather condition (such as sunny, cloudy, or rainy), and the transition probabilities are based on the probability of a particular weather condition occurring given the previous weather condition.
2.Stock Market Analysis: Markov models are used to predict stock prices based on historical data. The state of the system is defined by the current price of a stock, and the transition probabilities are based on the probability of the stock price increasing or decreasing given its current price.
3.Speech Recognition: Hidden Markov models (HMMs) are used to recognize speech patterns. The state of the system is defined by the phoneme being spoken, and the transition probabilities are based on the probability of transitioning from one phoneme to another.
4.Queueing Systems: Markov models are used to model the behavior of queueing systems, such as customer service centers or manufacturing processes. The state of the system is defined by the number of customers in the queue, and the transition probabilities are based on the probability of customers arriving or leaving the system.
5.Genetics: Markov models are used to analyze DNA sequences and predict gene locations. The state of the system is defined by the nucleotide sequence, and the transition probabilities are based on the probability of a particular nucleotide occurring given the previous nucleotide.

Markov Reward Process

Reward

- Feedback on actions
- Positive or negative based on the actions of the agent
- Example: Robot walk
- +ve reward for forward motion, -ve reward for falling over
- Example: Game playing
- +ve reward for winning, -ve reward losing
- Immediate or delayed
- Reward Hypothesis: All goals can be described by the maximization of expected cumulative reward

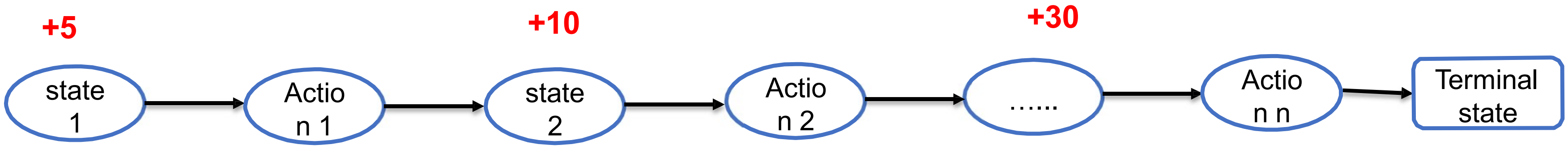
Return



total sum of reward the agent receives from the environment

$G_t = r_1 + r_2 + \ldots + r_t$

Episodic Tasks

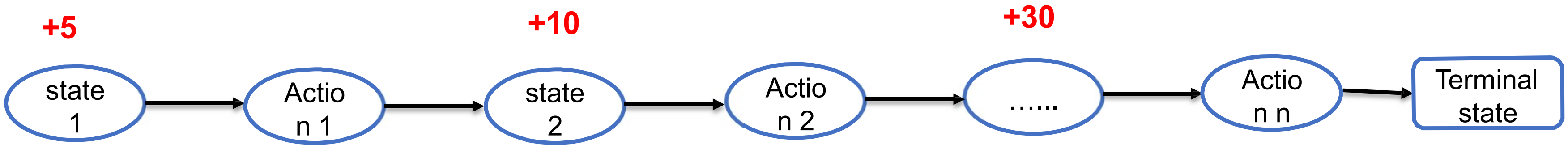


Episodic Task have a terminal state

S1-s2-s3…St

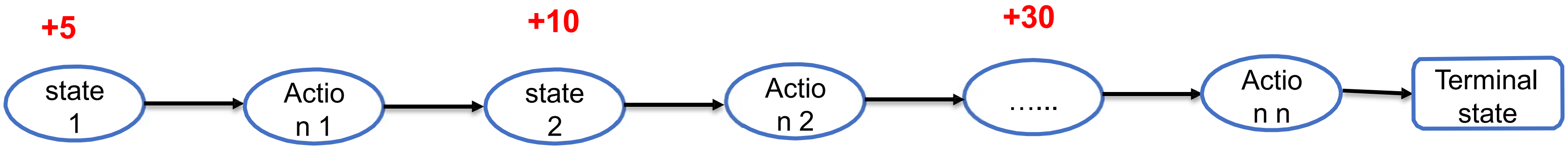
Continuous Tasks have no terminal states

Discount Factor



- How to prioritize immediate reward and future rewards
- Denoted by γ (gamma)
- Value between 0 and 1
- Avoids infinite returns in cyclic Markov processes
- Mathematically convenient to discount rewards
- In many practical situations like financial applications immediate rewards is better than delayed rewards
- Animal/human behaviour shows preference for immediate reward

Return



+5 +10 +30

state 1 → Actio n 1 → state 2 → Actio n 2 → …… → Actio n n → Terminal state

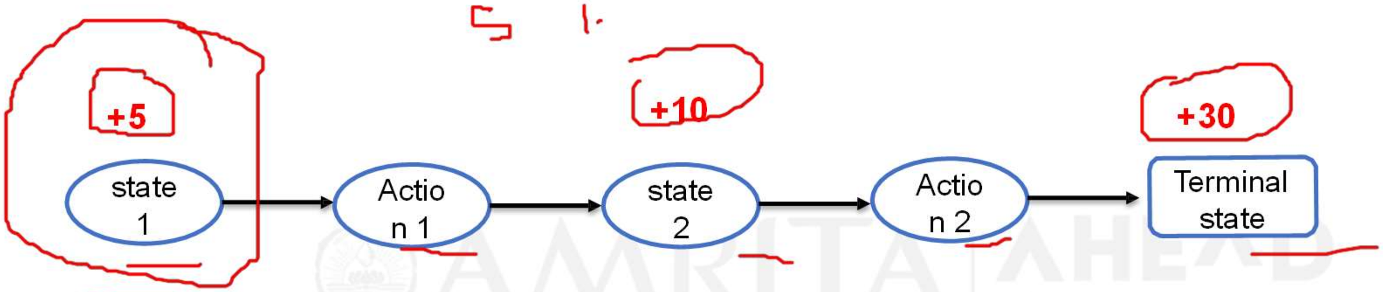
$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

- Discount factor is in the interval [0, 1]
- γ close to 0: *more importance to* ***immediate reward*** "myopic" evaluation
- γ close to 1: *more importance is given to* ***future rewards:*** "far-sighted" evaluation

Return

$$G_t \doteq \overset{5}{\underline{R_{t+1}}} + \overset{0}{\gamma R_{t+2}} + \overset{0}{\gamma^2 R_{t+3}} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

5 1.

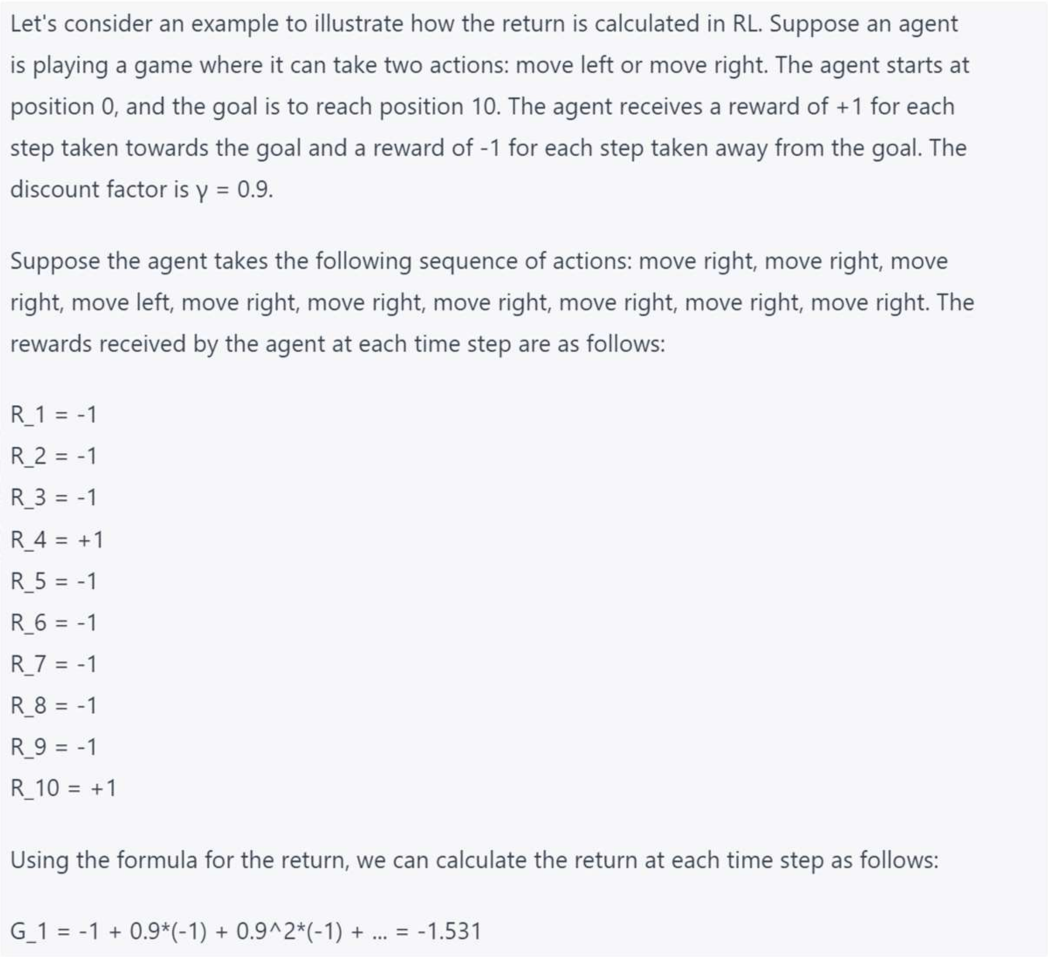


γ =0 , Gt = +5

γ =1 , Gt = 5+ 10 + 30

γ = .1

Gt = 5 + .1 x 10 + .01 x 30



Let's consider an example to illustrate how the return is calculated in RL. Suppose an agent is playing a game where it can take two actions: move left or move right. The agent starts at position 0, and the goal is to reach position 10. The agent receives a reward of +1 for each step taken towards the goal and a reward of -1 for each step taken away from the goal. The discount factor is γ = 0.9.

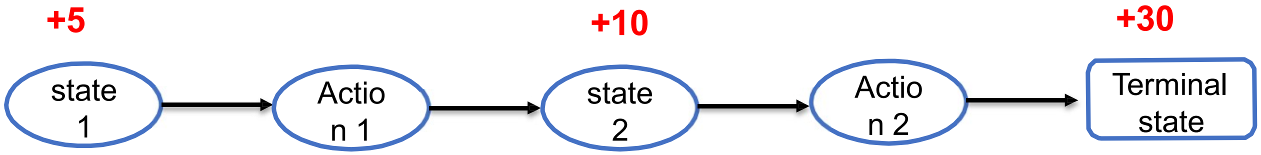
Suppose the agent takes the following sequence of actions: move right, move right, move right, move left, move right, move right, move right, move right, move right, move right. The rewards received by the agent at each time step are as follows:

R_1 = -1
R_2 = -1
R_3 = -1
R_4 = +1
R_5 = -1
R_6 = -1
R_7 = -1
R_8 = -1
R_9 = -1
R_10 = +1

Using the formula for the return, we can calculate the return at each time step as follows:

G_1 = -1 + 0.9*(-1) + 0.9^2*(-1) + ... = -1.531

Markov Reward Process



- A Markov reward process is a Markov chain with values.
- Definition

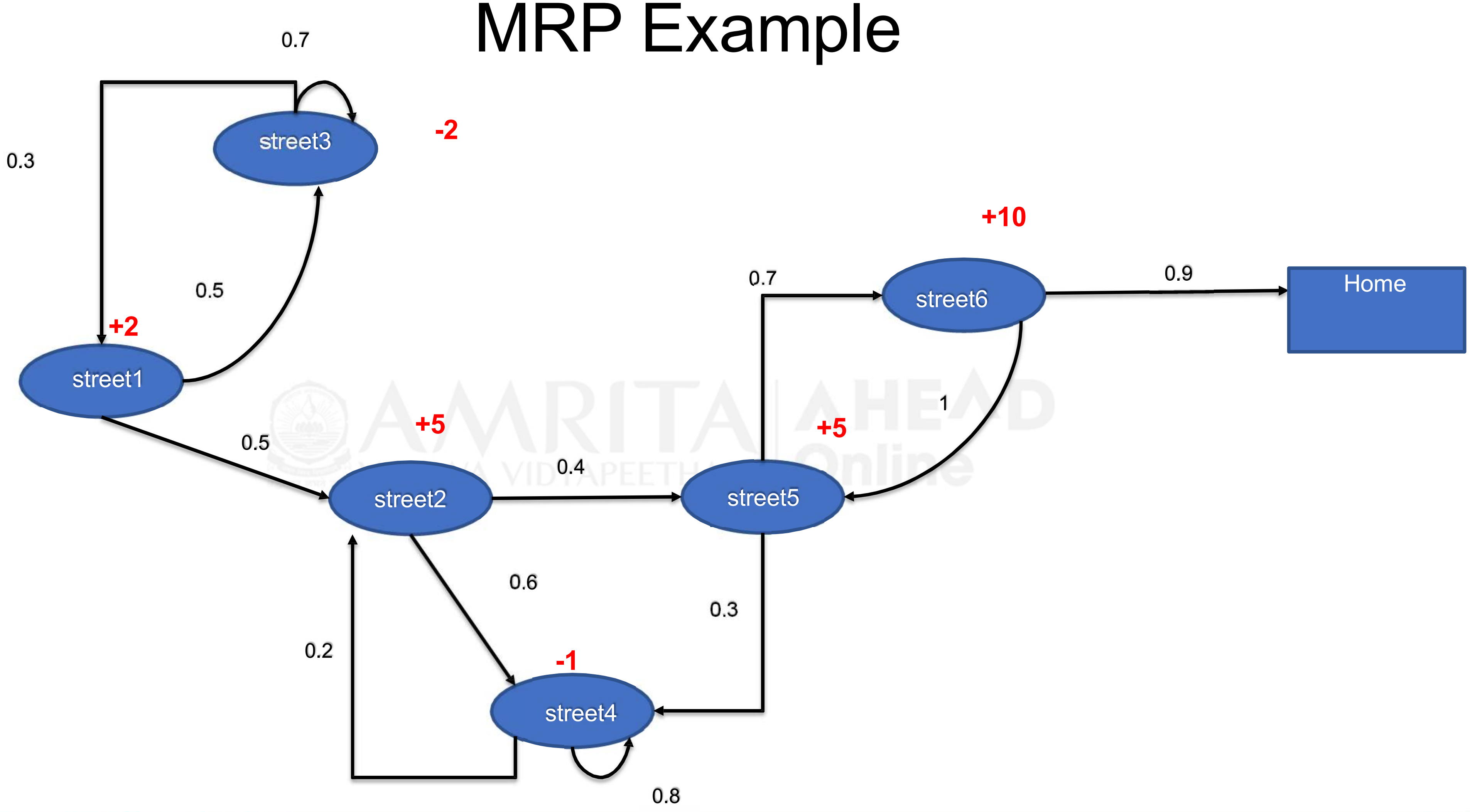
A Markov Rewards Process is a tuple <S, P, R, γ>

- S is a finite set of states
- P is a state transition probability matrix,

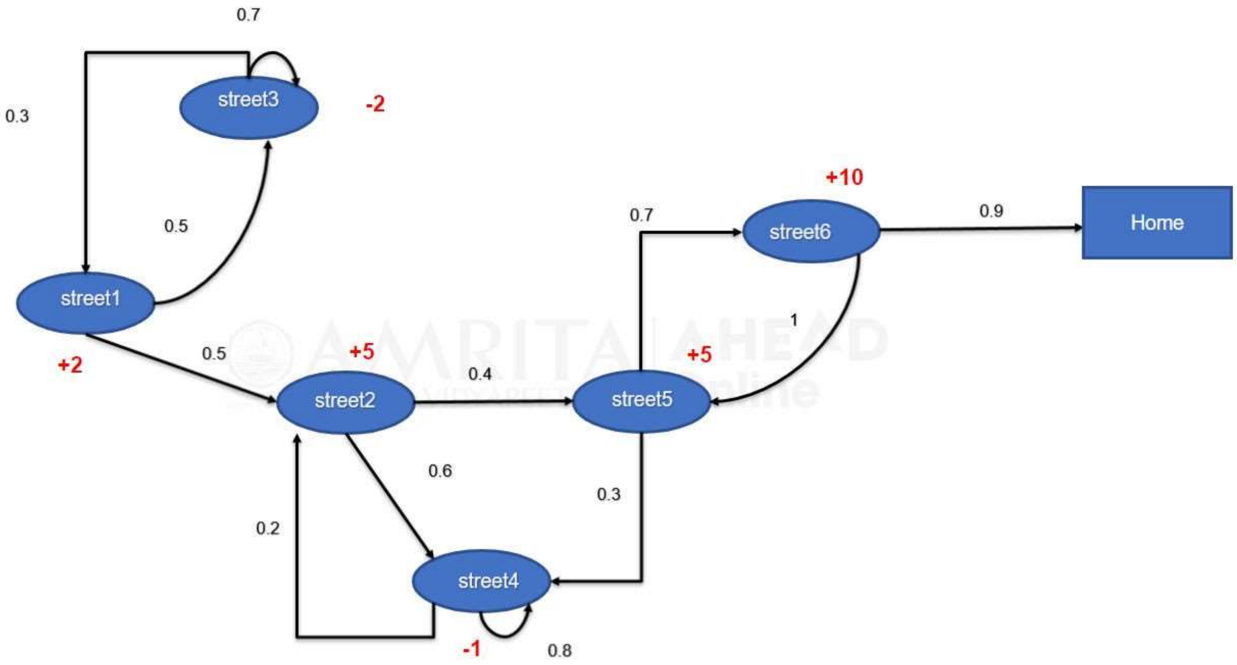
$$P_{ss'} = Prb[S_{t+1} = s^{'} | S_t = s]$$

- R is a reward function , $R_s = \mathbb{E}[R_{t+1} \quad | \quad S_t = S]$
- γ Is a discount factor, $\in [0,1]$

MRP Example


0.7
-2
street3
0.3
+10
0.5
0.7
0.9
Home
street6
+2
street1
1
+5
+5
0.5
0.4
street2
street5
0.6
0.3
0.2
-1
street4
0.8

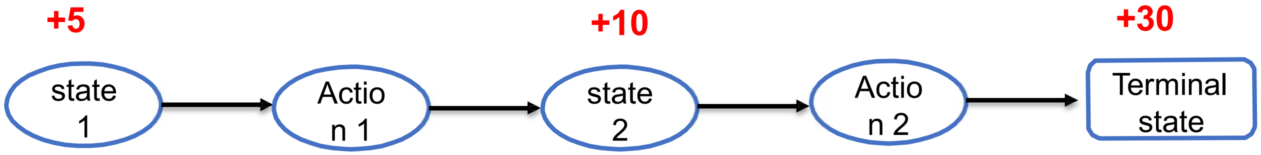
MRP Example



	S1	S2	S3	S4	S5	S6	home
S1		0.5	0.5				
S2				0.6	0.4		
S3	0.3		0.7				
S4		0.2		0.8			
S5				0.3	0.6		
S6					0.1		0.9
home							1

Bellman Equation

Markov Reward Process



- A Markov reward process is a Markov chain with values.
- Definition

A Markov Rewards Process is a tuple < S, P, R, γ >

- S is a finite set of states
- P is a state transition probability matrix,

$$P_{ss'} = \mathbb{p}[S_{t+1} = s' | S_t = s]$$

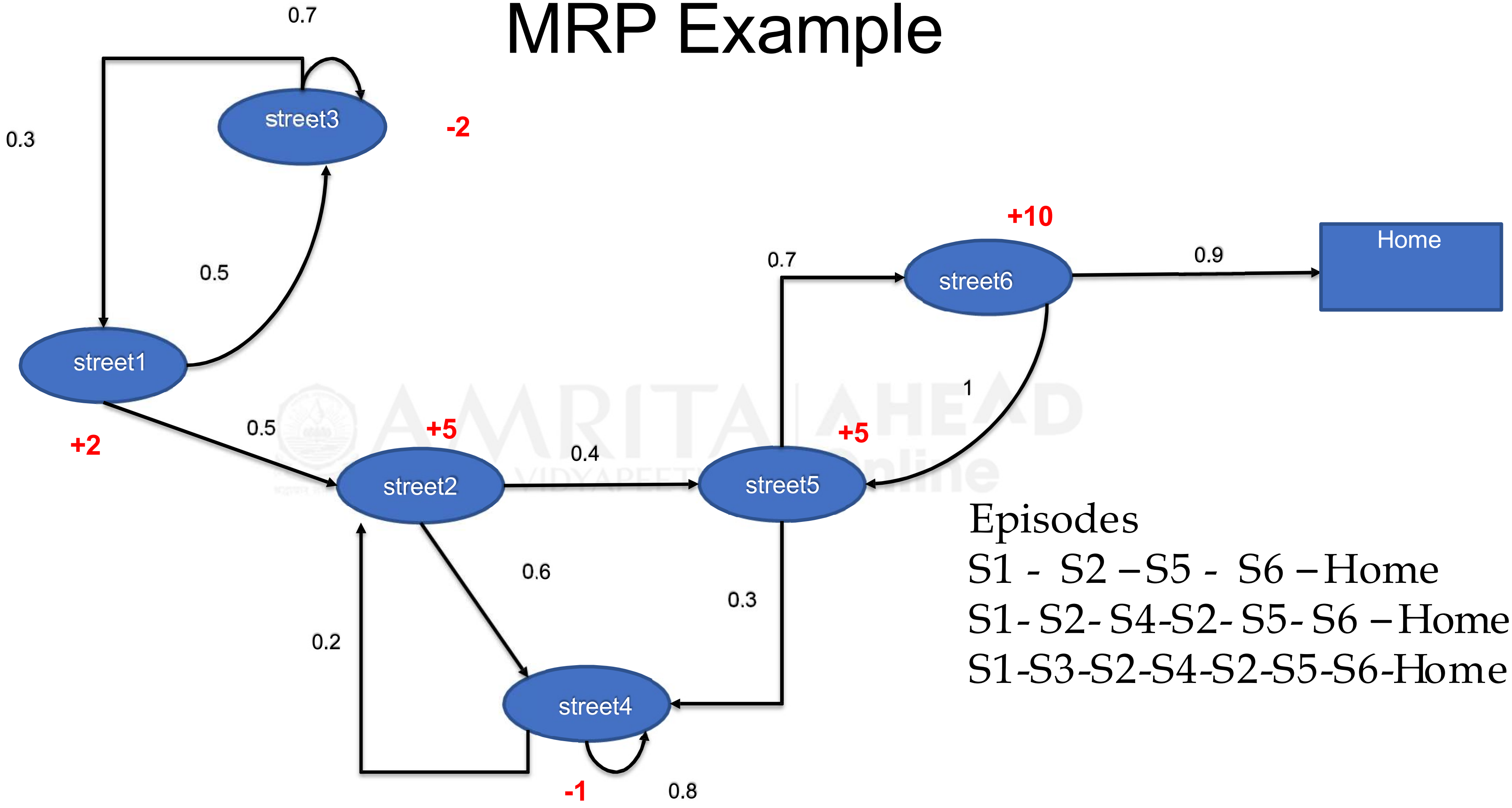
- R is a reward function , $R_s = \mathbb{E}[R_{t+1} \mid S_t = S]$
- γ Is a discount factor, $\in [0,1]$

Value Function

- The value function v(s) gives the long-term value of state s
- The state value function v(s) of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

MRP Example



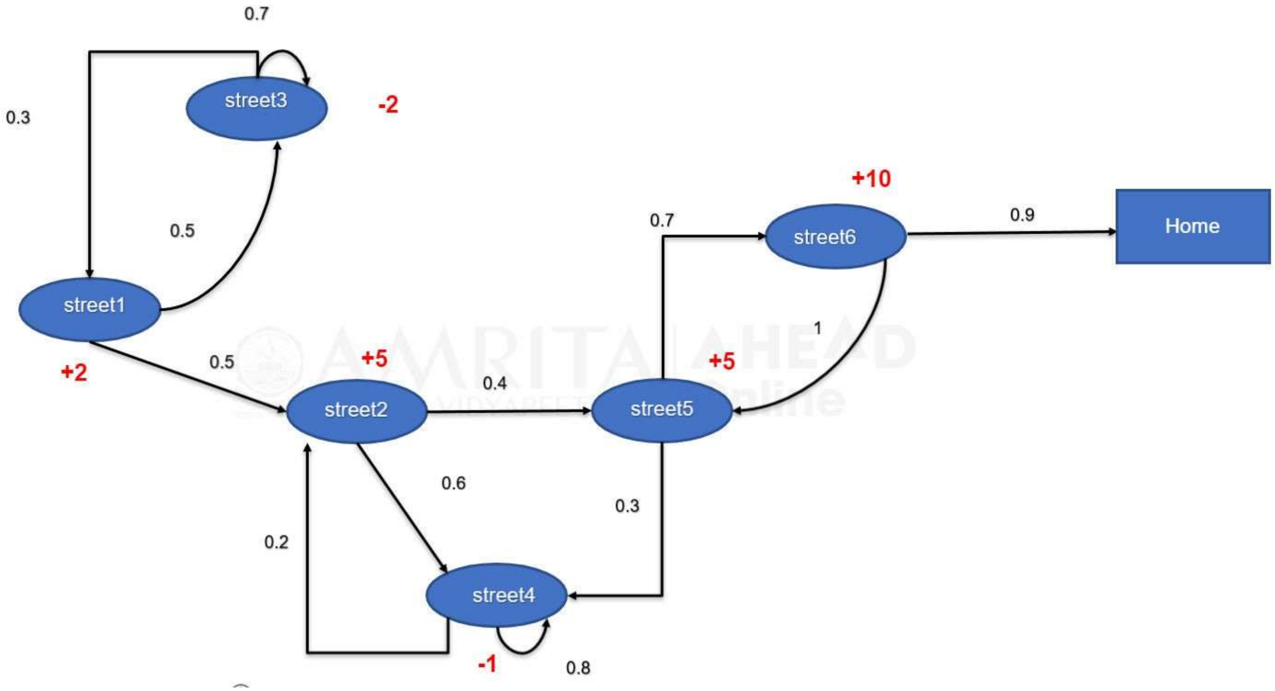
Episodes

S1 - S2 –S5 - S6 –Home

S1- S2- S4-S2- S5- S6 –Home

S1-S3-S2-S4-S2-S5-S6-Home

State Value Computation



Episodes
S1 - S2 – S5 - S6 – Home

γ = 0.25

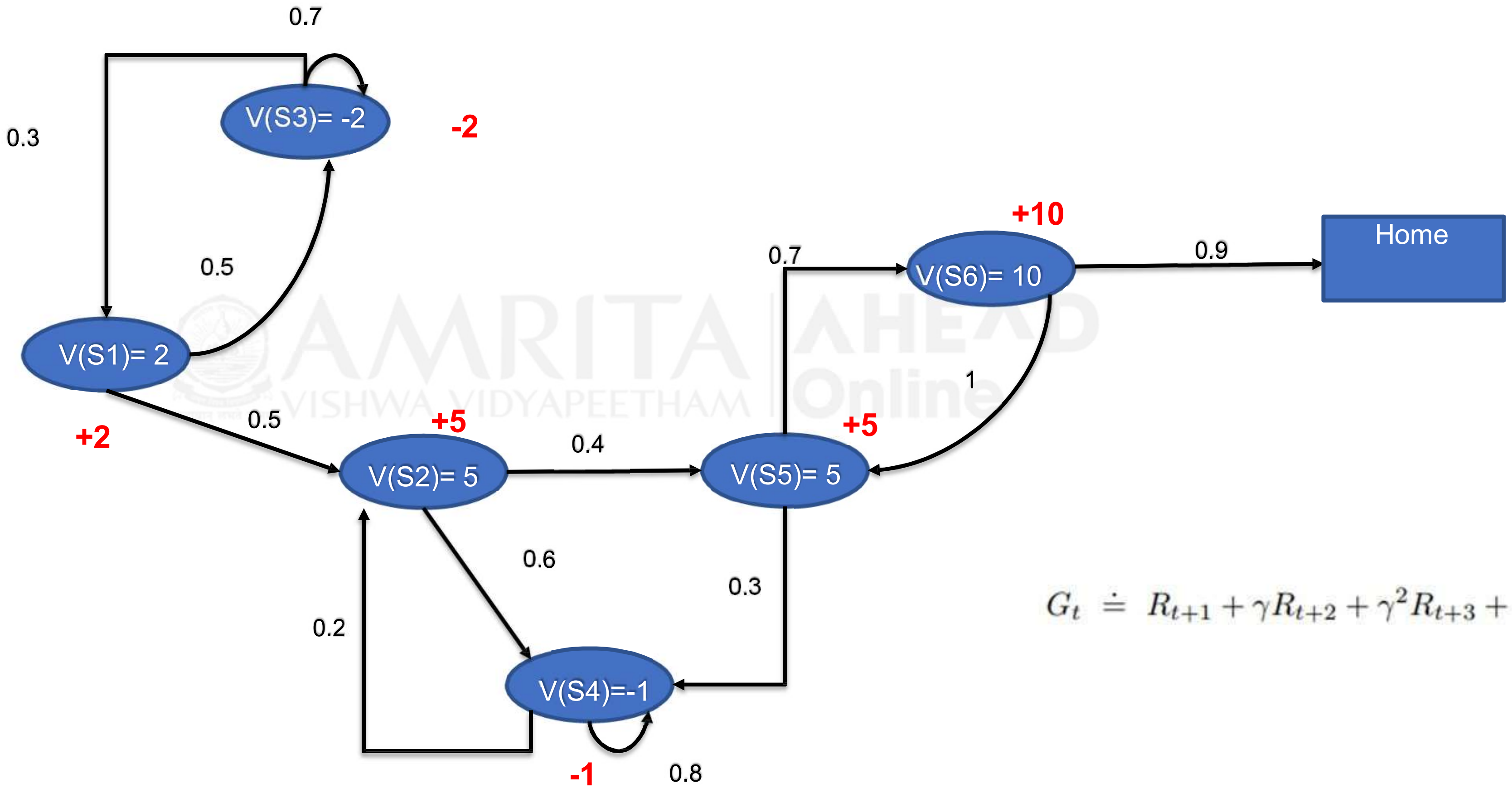
Return of S1

G1= 2 + .25 x 5 + .0625 x 5 + .0156 x 10

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

Equation: Sutton and Barto, Reinforcement Learning: An Introduction

State Value when γ = 0



$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

Bellman Equation for MRPs

The value function can be decomposed into two parts:

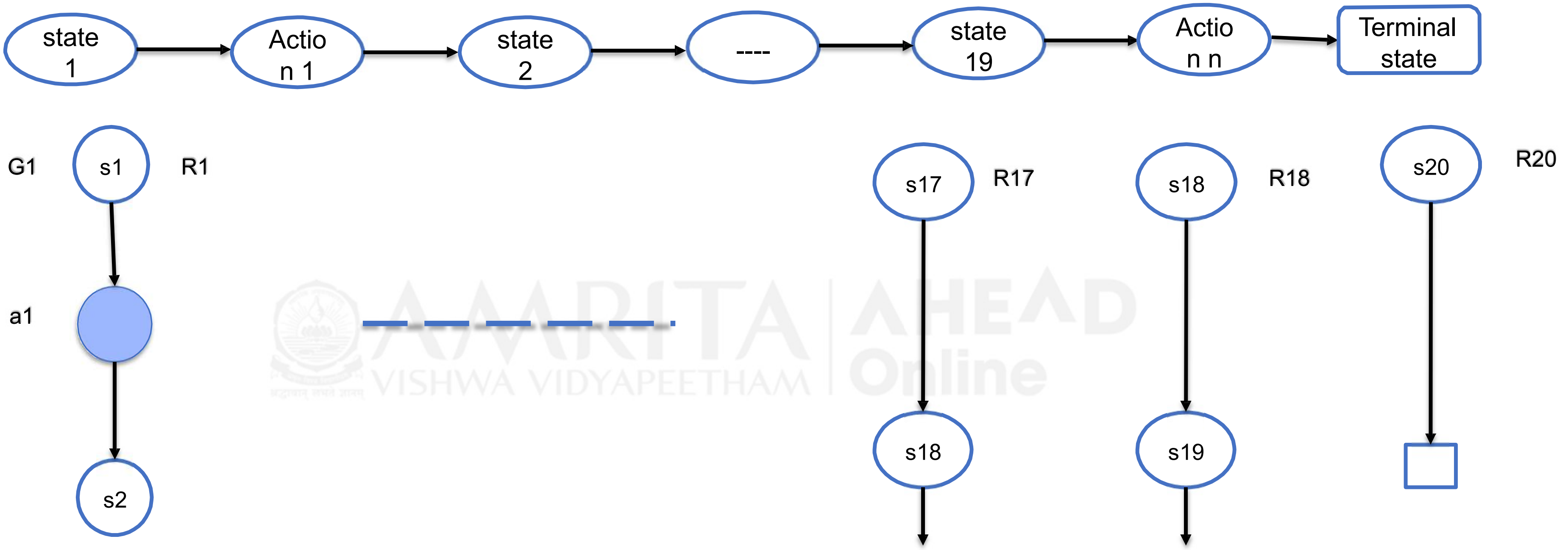
- Immediate reward R_{t+1}
- Discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
\mathrm{v(s)} &= \mathbb{E}[G_t \mid S_t] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots) \mid S_t = \quad] \\
s & \qquad\qquad\qquad\quad \mid \qquad\quad] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1} \; S_t \mid = s \quad] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1} \; S_t = s
\end{aligned}
$$

Equation: Sutton and Barto, Reinforcement Learning: An Introduction

Bellman Equation for MRPs

$G_t = R_t + G_{t+1}$



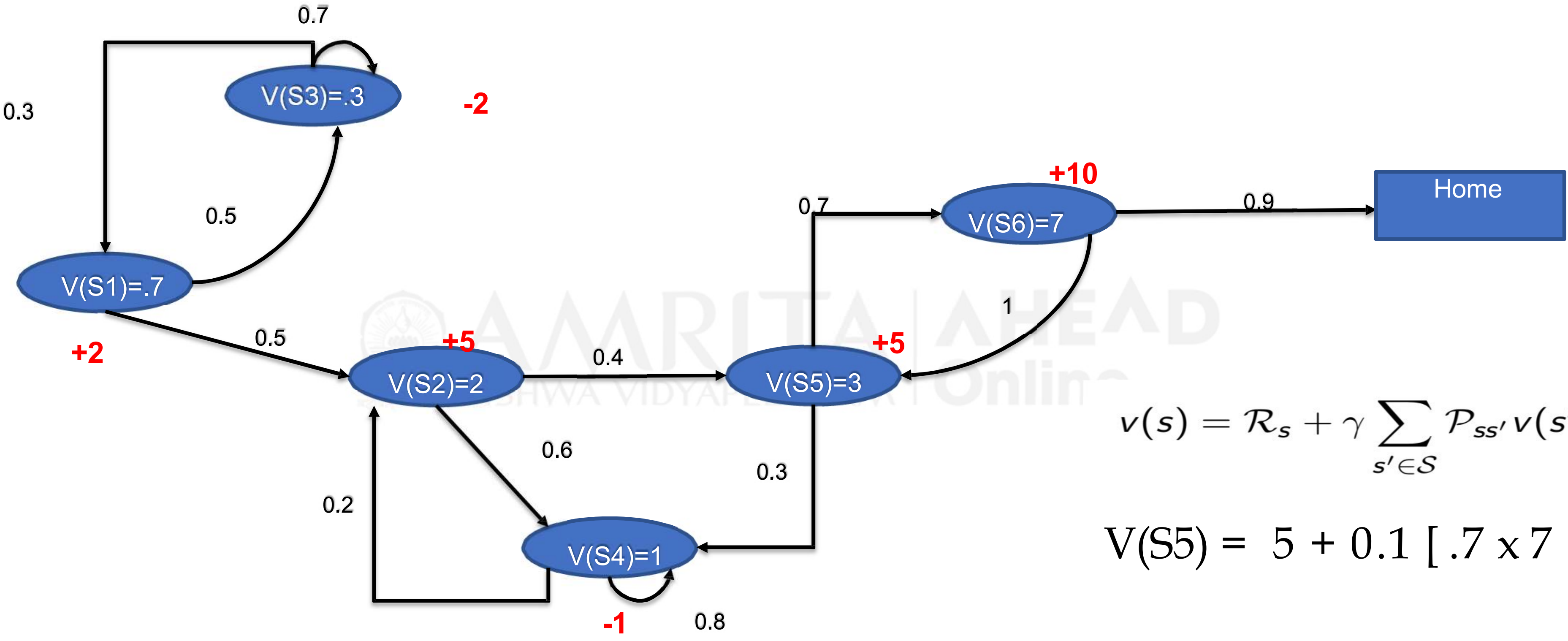
$G_{17} = R_{17} + \gamma G_{18}$

$G_{18} = R_{18} + \gamma G_{19}$

$G_{19} = R_{19} + \gamma G_{20}$

$G_1 = R_1 + \gamma G_2$

Bellman Equation for MRPs



$$v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} v(s')$$

V(S5) = 5 + 0.1 [.7 x 7 + .3 x 1]

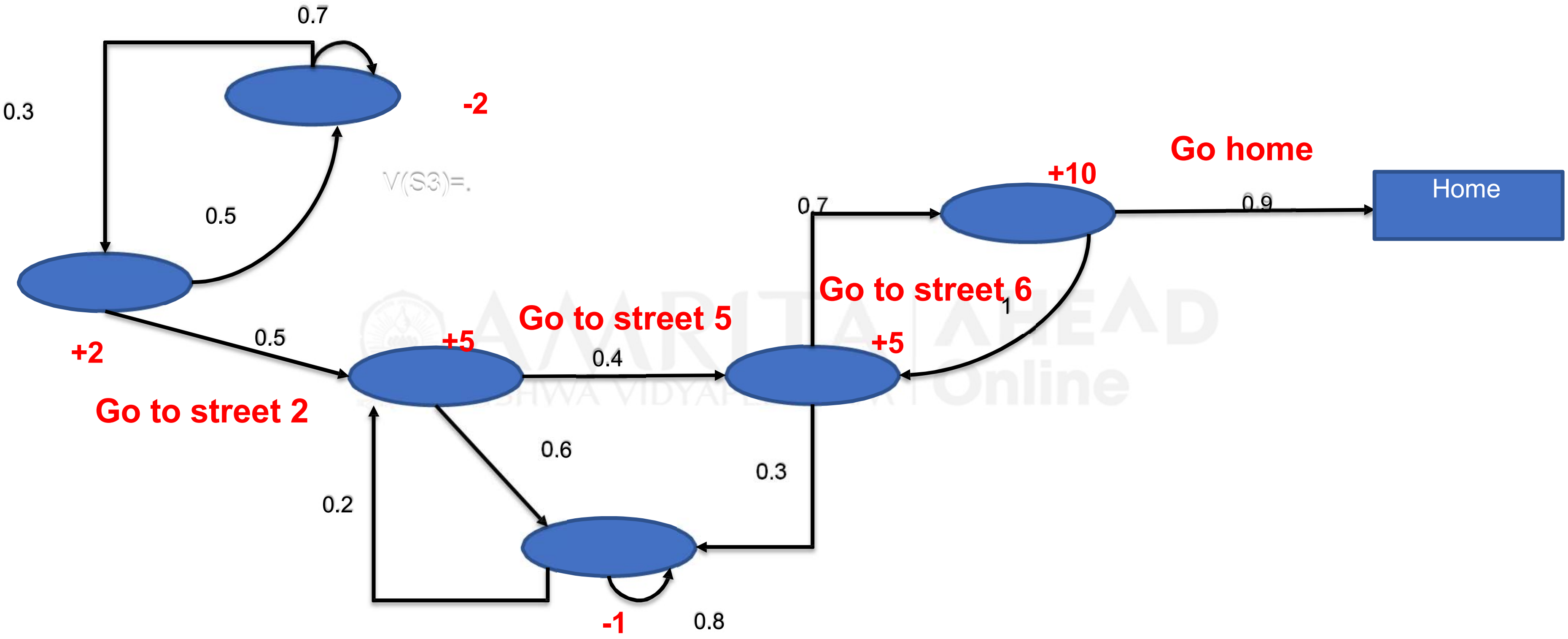
Markov Decision Process

- Markov decision process (MDP) is a Markov reward Process with decisions. It is an environment in which all states are Markov.
- Definition : A Markov Decision process is a tuple $< S, A, P, R, \gamma >$
- S is a finite set of states
- A is a finite set of actions
- P is a state transition probability matrix, $P_{SS'}^{a} = \mathbb{p}[S_{t+1} = s' | S_{t} = s, A_{t} = a]$
- R is a reward function $R_s^a = \mathbb{E}[R_{t+1} | S_t = s \ , A_t = \quad]$

a

- γ is a discount factor $\gamma \in [0,1]$

Example MDP


0.7
0.3
-2
V(S3)=.
0.5
Go home
+10
0.7
0.9
Home
Go to street 6
1
Go to street 5
0.5
+5
+2
0.4
+5
Go to street 2
0.6
0.3
0.2
-1
0.8

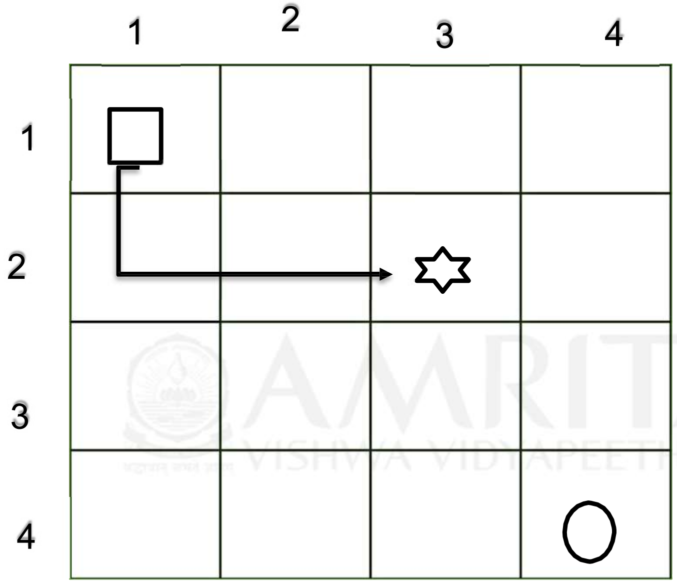
Policy

A Policy π is a distribution over actions given states,

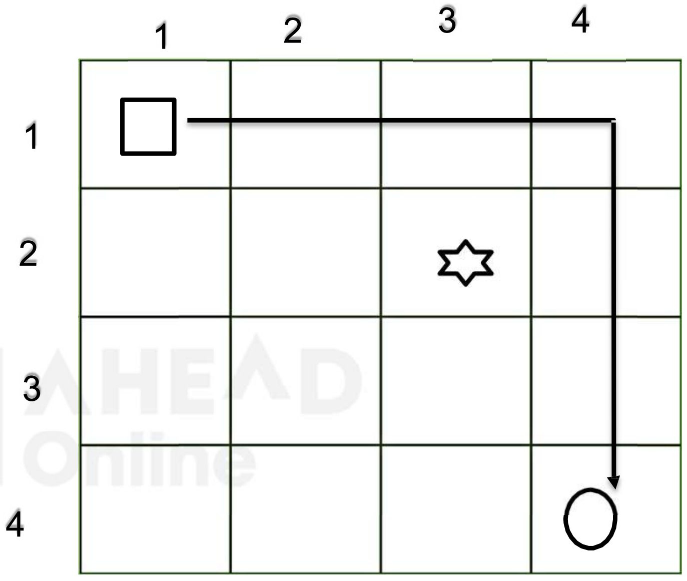
$$\pi(a|s) = \mathbb{p}[A_t = a | S_t = s]$$

- A policy fully defines the behaviour of an agent
- MDP Policies depend on the current state

Example of a Policy



π_1



π_2

Value Function

- The state-value function $V_\pi(S)$ of an MDP is the expected return starting from state s and then following policy π

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

- The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$$

Bellman Expectation Equation

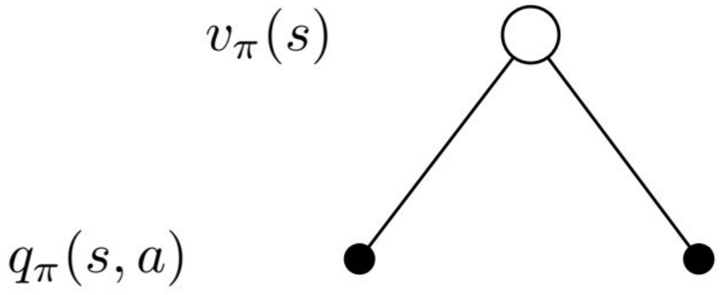
- The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$V_\pi(S) = \mathbb{E}_\pi[R_{t+1} + \qquad (S_{t+1})|S_t = \quad]$$
$$\gamma V_\pi \qquad\qquad\qquad\qquad\qquad\qquad s$$

- The action-value function can similarly be decomposed,

$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \qquad (S_{t+1}, \qquad)|S_t = s, A_t = \quad]$$
$$\gamma q_\pi \qquad\qquad\qquad\qquad A_{t+1} \qquad\qquad a$$

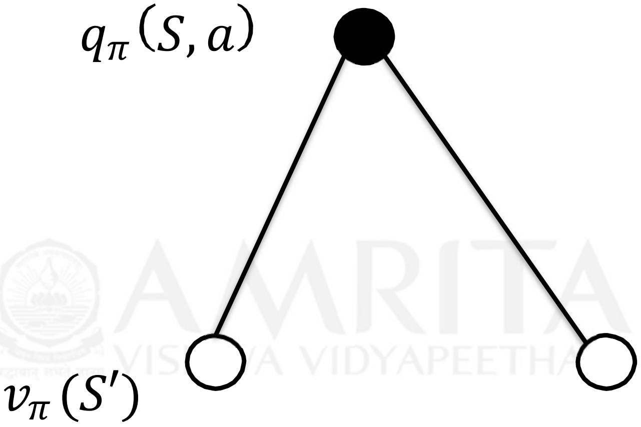
Bellman Expectation Equation for V^{π}



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

Equation: Sutton and Barto, Reinforcement Learning: An Introduction

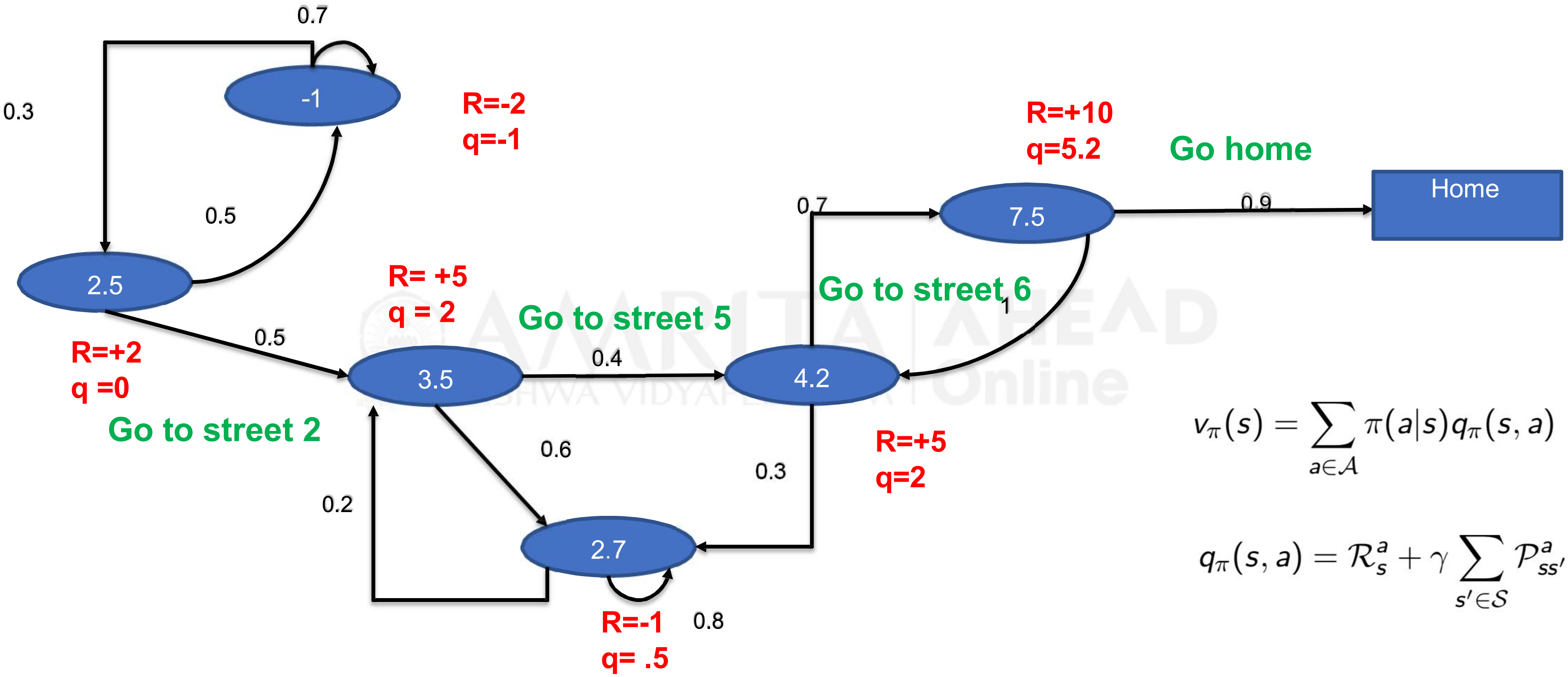
Bellman Expectation Equation for Q^π



$$q_\pi(S,a) = R^a_S + \gamma \underset{s'\,\epsilon S}{\quad} P^a_{ss'} V_\pi(S')$$

Equation: Sutton and Barto, Reinforcement Learning: An Introduction

State/Action Value Function of MDP



$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Optimal Value Function

The optimal state-value function $V_*(s)$ is the maximum value function over all policies

$$V_*(s) = \max_{\pi} V_{\pi}(s)$$

The optimal action value function
$q_*(s, a)$ is the maximum action $-$ value function over all policies

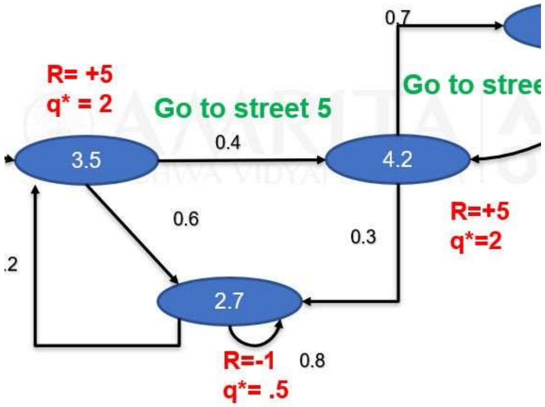
$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a)$$

Finding an Optimal Policy

An optimal policy can be found by maximising over $q_*(s,a)$,

$$\pi_*(a|s) = \begin{matrix} 1 \;\; if\ a = \underbrace{argmax}_{a\epsilon} \quad q_*(s,a) \\ 0 \qquad otherwise \end{matrix}$$

- If we know $q_*(s,a)$, we can find the optimal policy

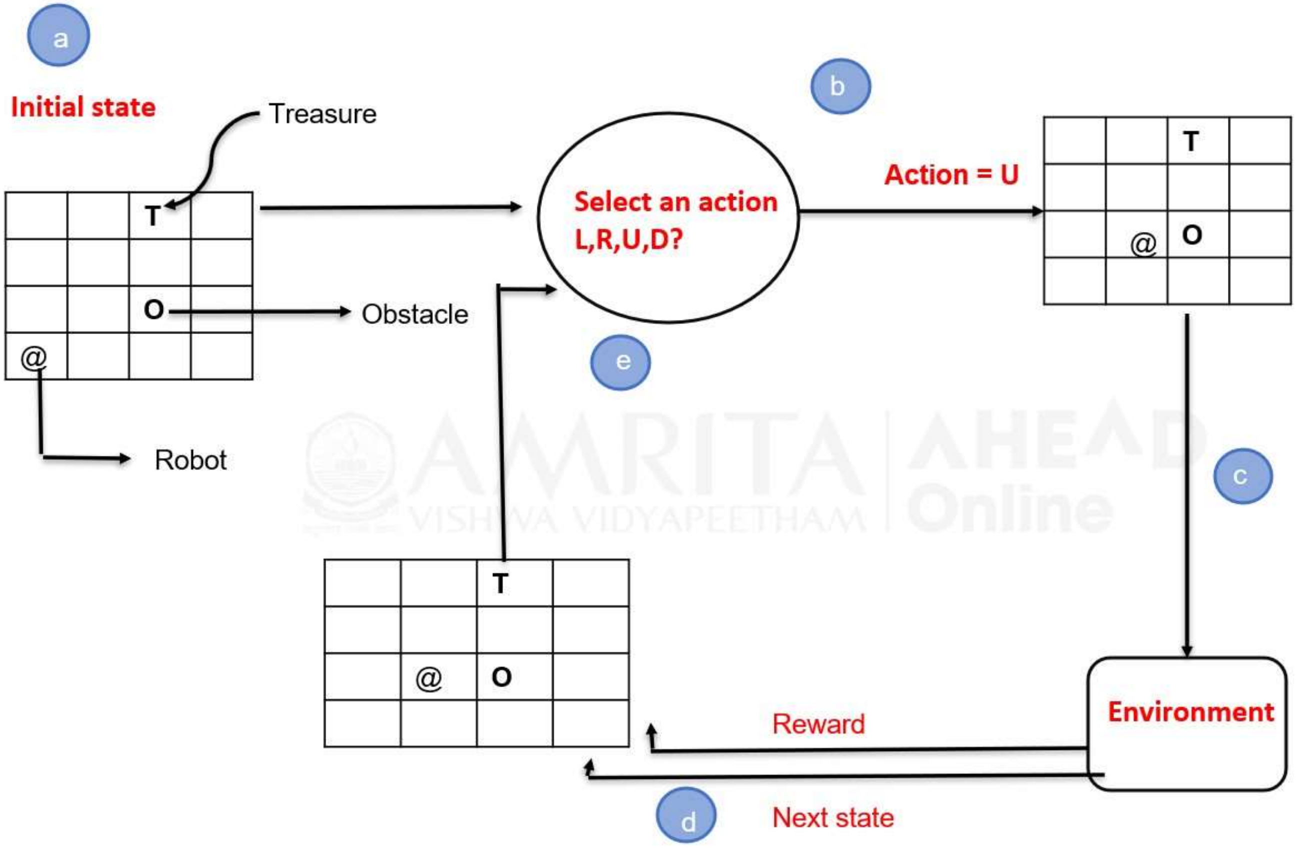


MDP Example Problem

Consider an 4x4 grid for robot navigation. The agent has 4 possible actions in each state (grid square): From the current grid, possible actions include move to a grid left, right, up, down. If the direction of movement is blocked, the agent remains in the same grid square. The initial state of the agent is IS (4,4). Two terminal states: T1, and T2 and has rewards 10 and -10 respectively. All other states have a reward of -1.

S= {(1,1), (1,2), (1,3), (1,4), (2,0),... , (4,4)}
A= {L, R, U, D}
R={ +10 for T1, -10 for T2, -1 for all other states}
Learning rate 0.7
Discount factor 0.6

How MDP works?



References

- [Chapter 3] Sutton and Barto, Reinforcement Learning: An Introduction, The MIT Press Cambridge, Massachusetts London, England, 2015
- Csaba Szepesvari, Algorithms for Reinforcement Learning, Morgan &Claypool, United States, 2010
- https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver