A Brief Introduction to Reinforcement Learning and Markov Decision Processes

Eirik Fagerhaug

Norwegian University of Science and Technology

Slides Adapted from/based on:

- Slides from Erlend Coates
- Slides from Hanna Hajishirzi
- Slides from Deepmind (Hado von Hasselt)
- Grokking DRL (Miguel Morales)
- AI: A Modern Approach (Russel and Norvig)

Reading Material

Russel and Norvig, Chapter 16.1 and Chapter 23.1

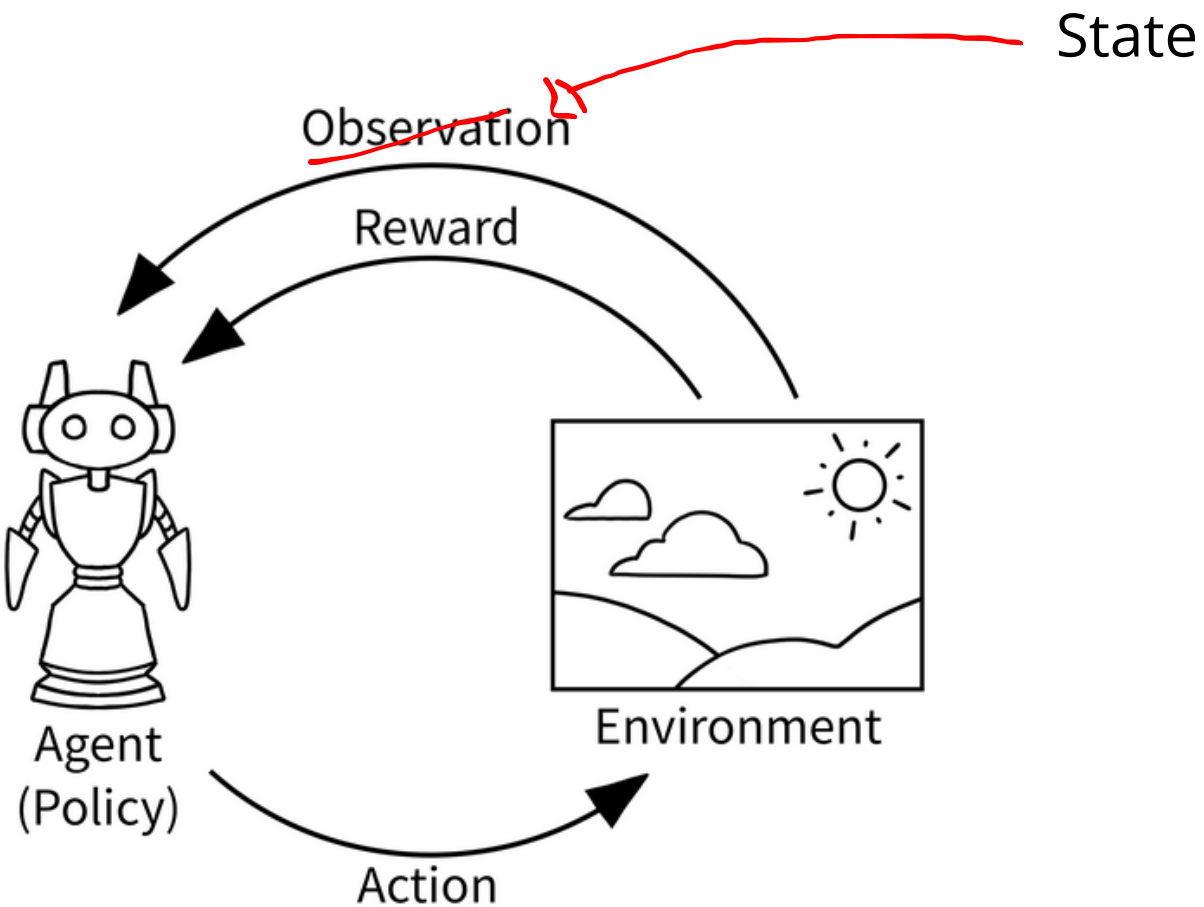
→ • Introduction to Reinforcement Learning

• Markov Decision Processes (MDPs)

Reinforcement Learning

- People and animals learn by interacting with our environment
- This differs from certain other types of learning
  - It is active rather than passive
  - Interactions are often sequential
- We are goal-directed
- We can learn without examples of optimal behavior
- Instead, we learn by receiving some reward signal

The interaction loop



Goal: optimize the sum of rewards through repeated interaction

The reward hypothesis

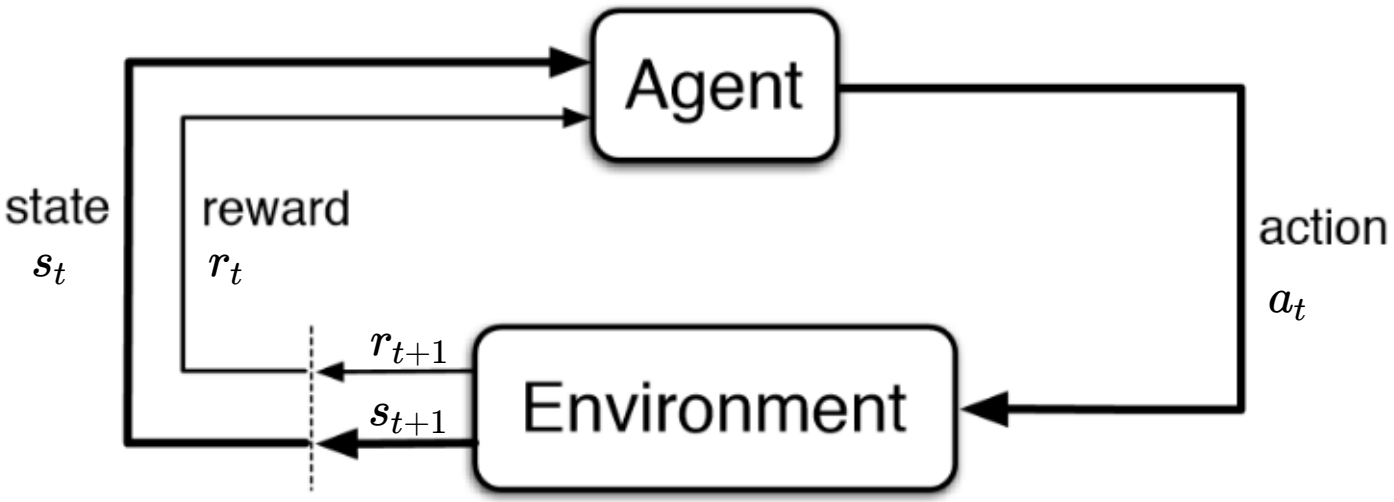
Reinforcement learning is based on the reward hypothesis:

"Any goal can be formalized as the outcome of maximizing a cumulative reward."

What is Reinforcement Learning?

- The science and framework of learning to make decisions from interaction
- This requires us to think about:
  - time
  - (long-term) consequences of actions
  - actively gathering experience
  - predicting the future
  - dealing with uncertainty
- Huge potential scope

Agent-Environment interaction



- At each step *t* the agent:
  - Receives state s_t (and reward r_t)
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_{t+1} (and reward r_{t+1})

States, Actions and Rewards

The state is a unique and self-contained configuration of the environment.

An action is what the agent does to affect the environment.

The reward is a measure of how well the agent is doing.

States, Actions and Rewards: Examples

Example: Playing Chess

- State: The organization of pieces on the board
- Action: The agents' (your) move
- Reward: +/- for winning or losing, e.g., +1 for winning, +1/2 for a draw, 0 for losing

Example: Managing an investment portfolio

- State: The stock market and your own portfolio
- Action: Sell / buy
- Reward: +/- for increase/decrease in portfolio total worth

Example: Making a robot walk

- State: Orientation (and position) of robot and its limbs
- Action: Moving the joints/limbs
- Reward: +/- for forward-motion / falling over

RL Core concepts

The reinforcement learning formalism includes:

- Environment (dynamics of the problem)
- Reward signal (specifies the goal)
- An Agent, containing:
  - Policy
  - Utility function (also called value function)
  - A model

Elements of an RL agent

An RL agent may include one or more of the following components:

- Policy: Agents' plan/behavior.
- Utility function: A function to evaluate how good/bad a state is.
- Model: A model predicts what the environment will do next.

RL methods can be classified as either utility-based or policy-based and as either model-based or model-free.

Example Video

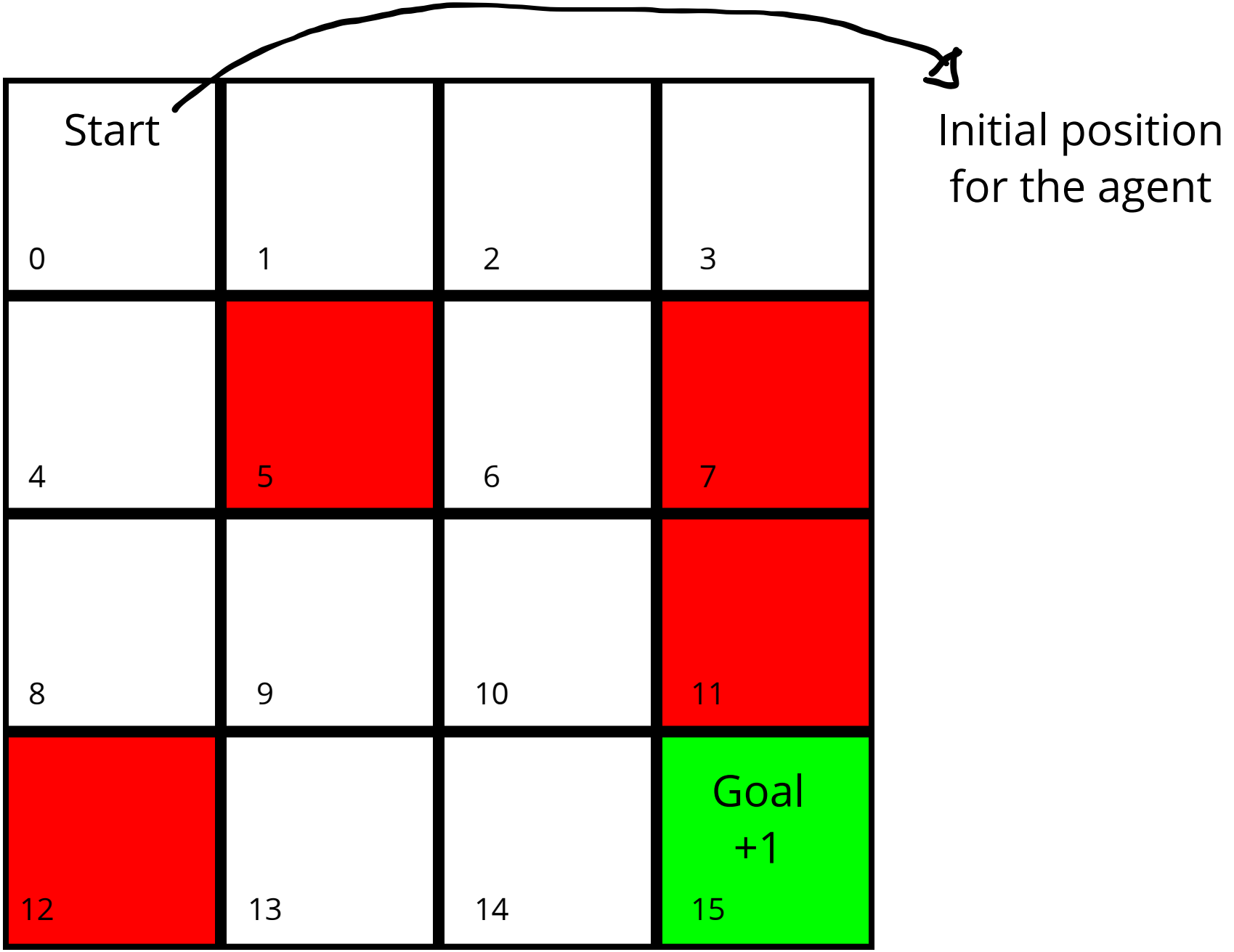
- Introduction to Reinforcement Learning

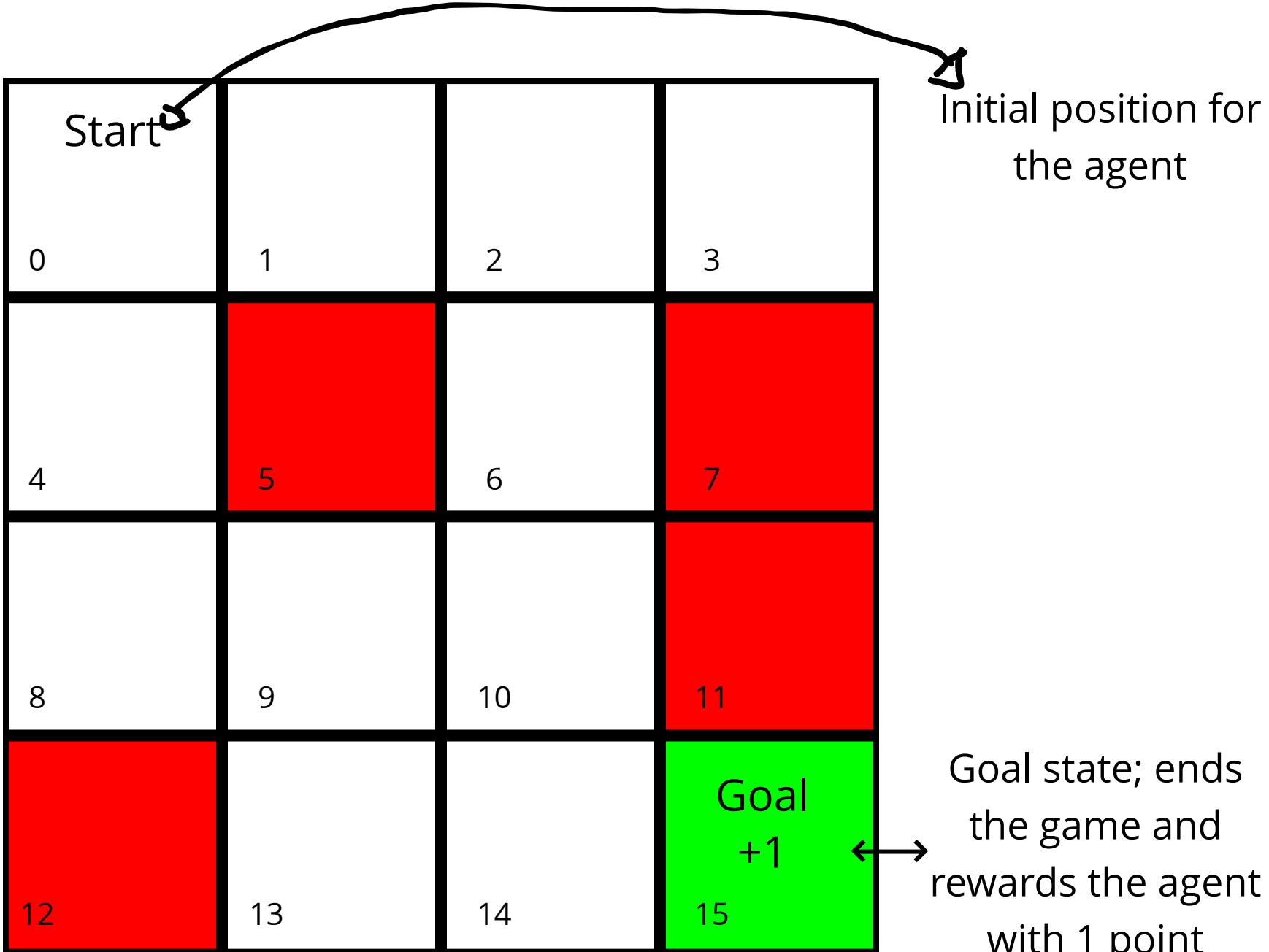
→ • Markov Decision Processes (MDPs)

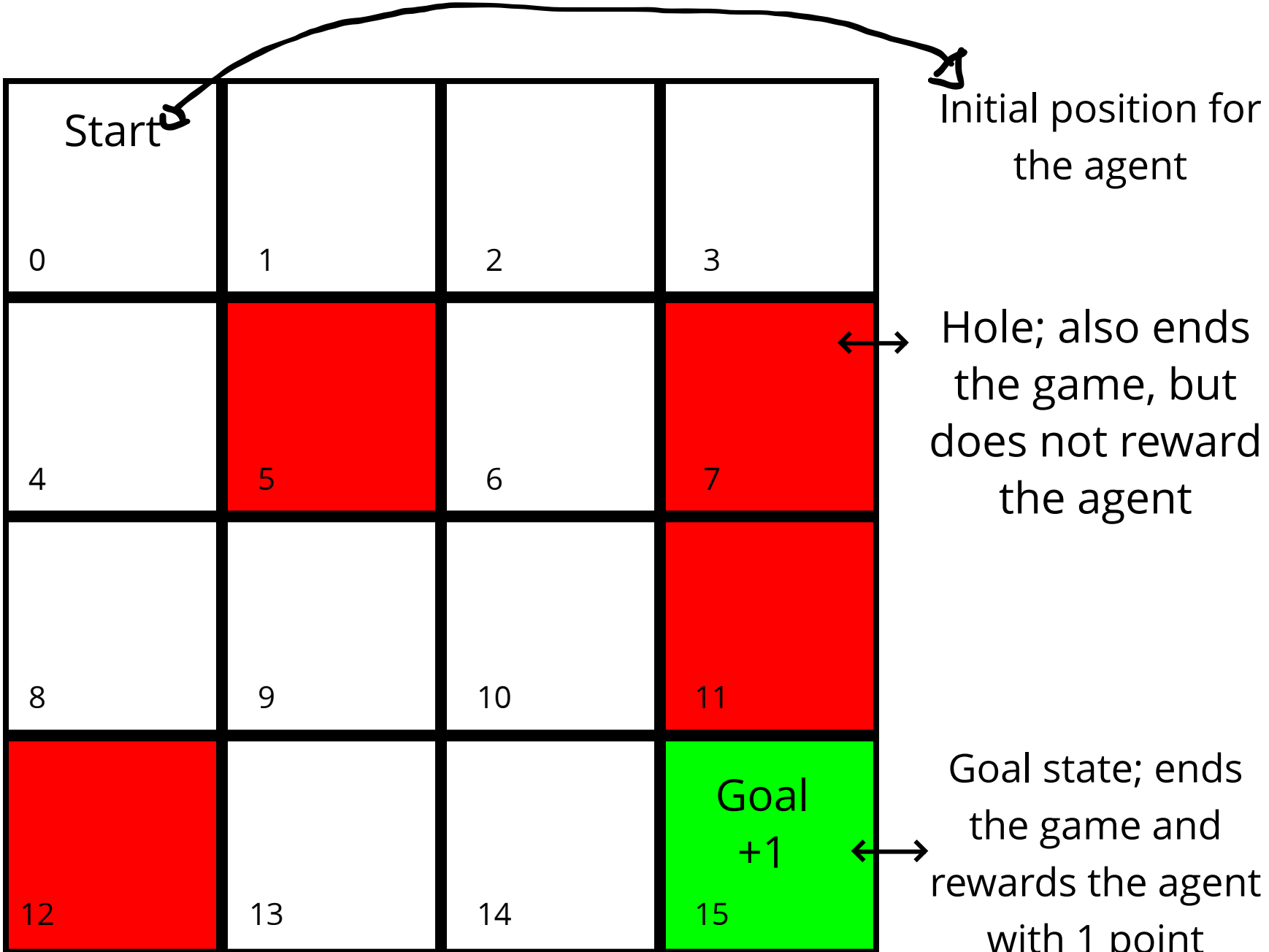
FrozenLake Environment

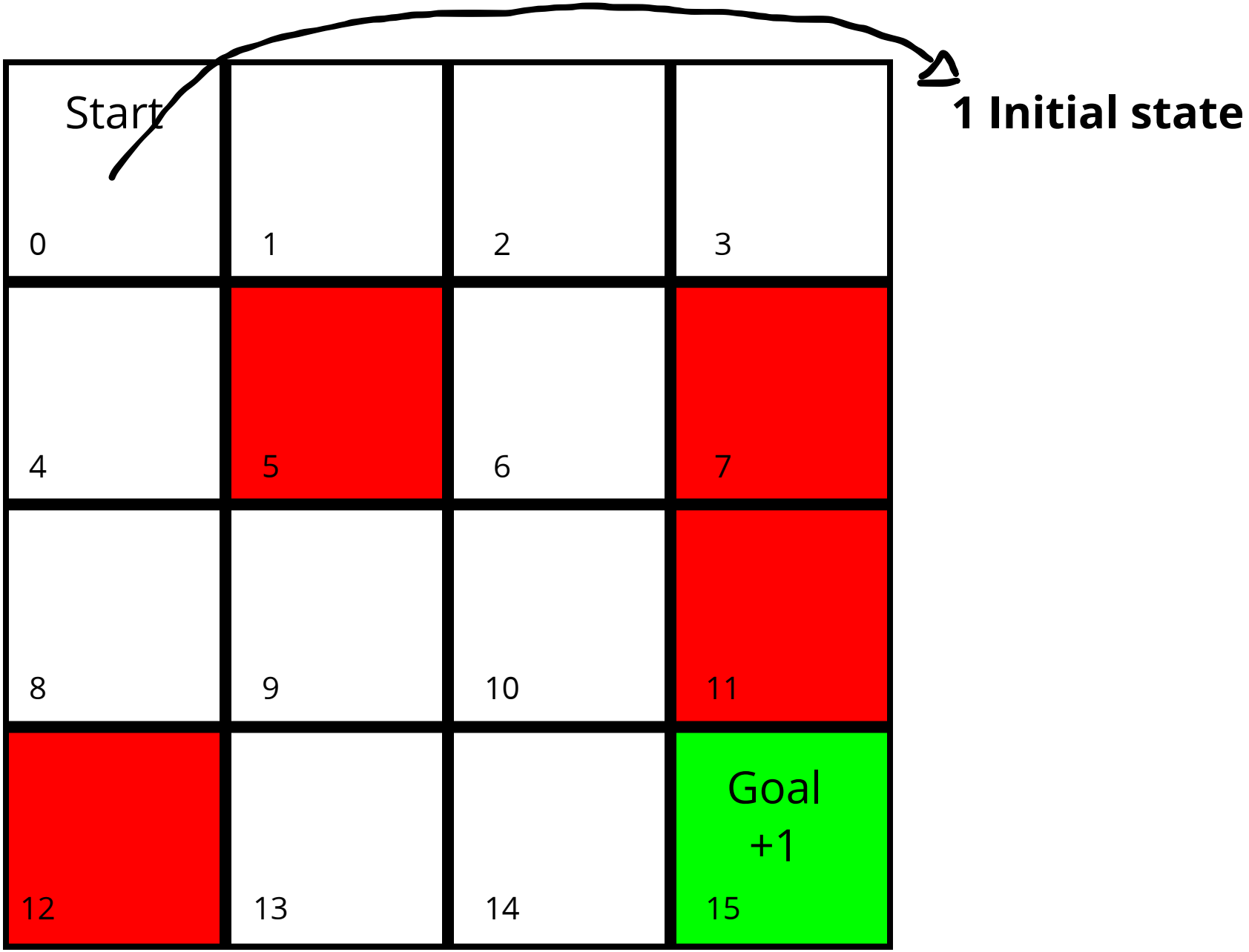
Winter is here. You and your friends were tossing around a frisbee at the park when you made a wild throw that left the frisbee out in the middle of the lake. The water is mostly frozen, but there are a few holes where the ice has melted. If you step into one of those holes, you'll fall into the freezing water. At this time, there's an international frisbee shortage, so it's absolutely imperative that you navigate across the lake and retrieve the disc. However, the ice is slippery, so you won't always move in the direction you intend.

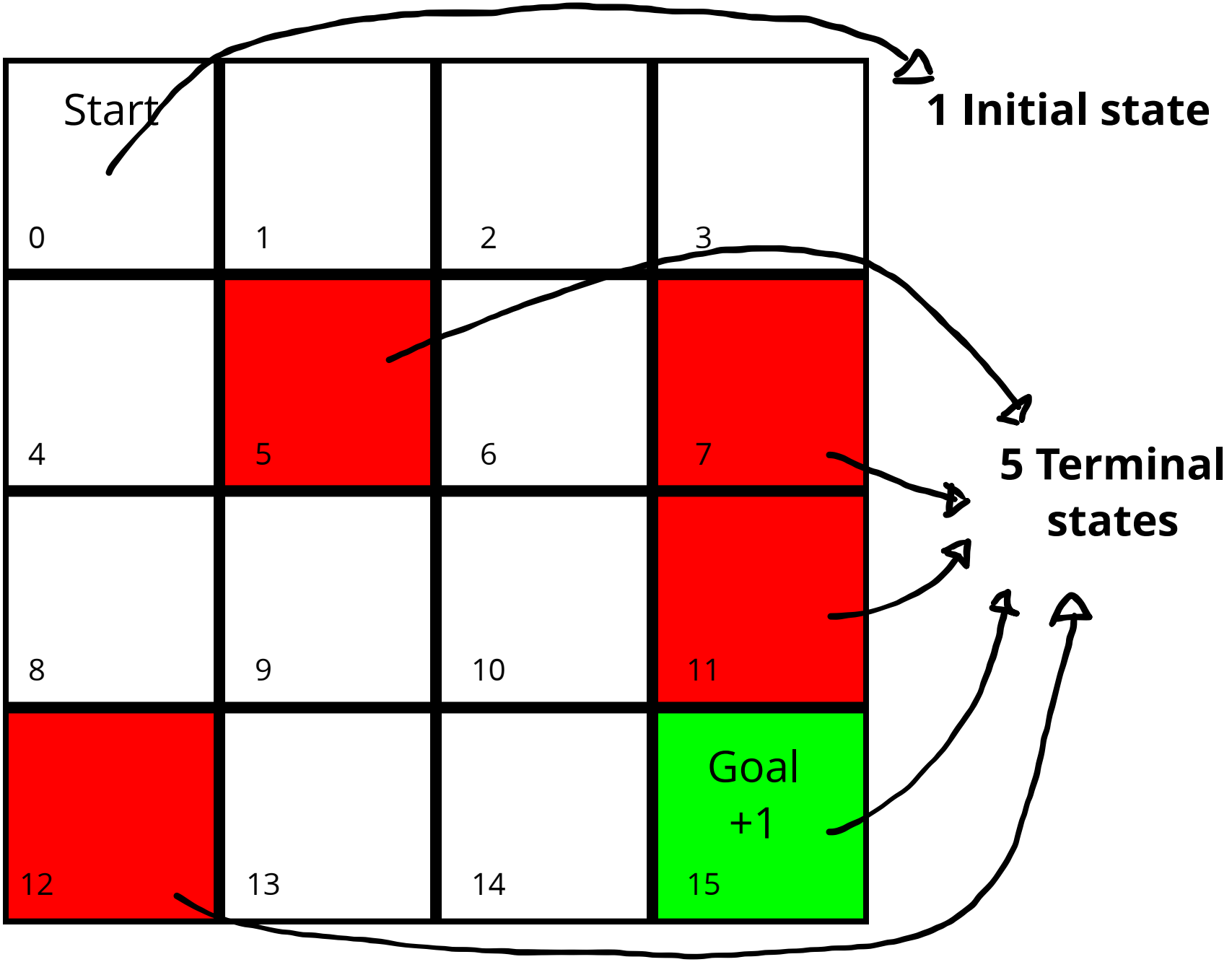
Start 0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	Goal +1 15

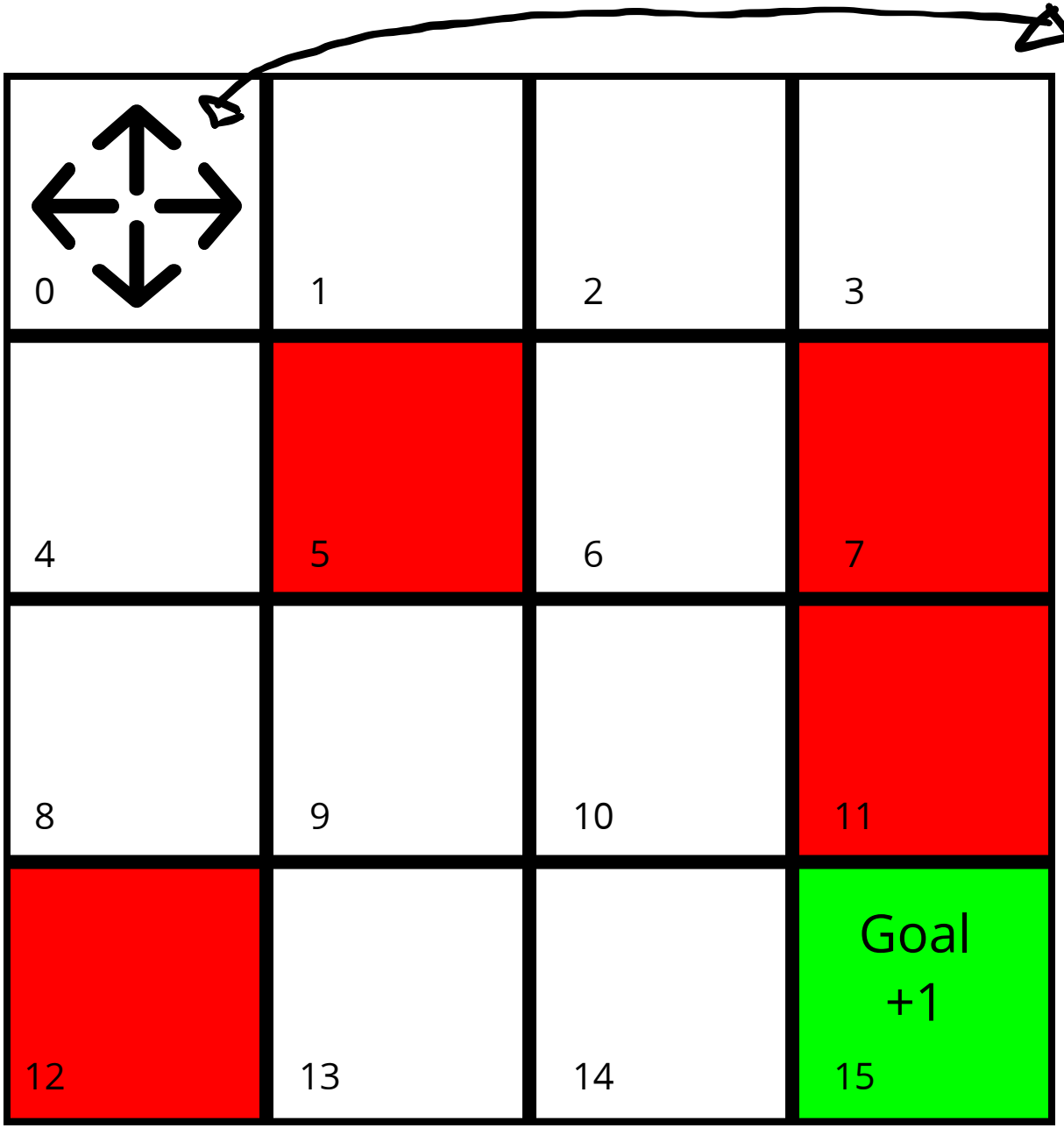

Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
Initial position
for the agent


Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
Initial position for the agent
Goal state; ends the game and rewards the agent with 1 point

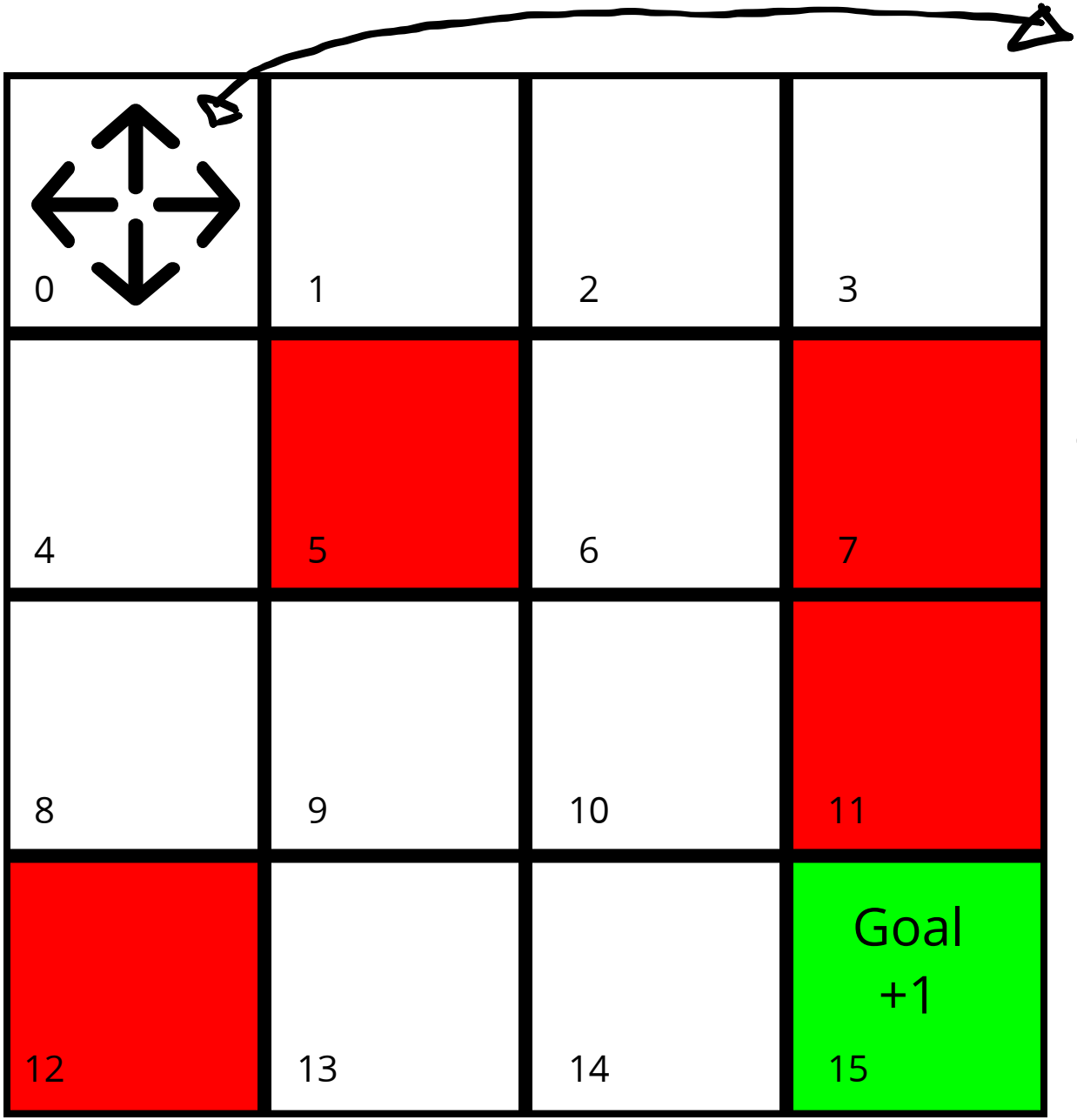

Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
Initial position for the agent
Hole; also ends the game, but does not reward the agent
Goal state; ends the game and rewards the agent with 1 point


Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
1 Initial state


Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
1 Initial state
5 Terminal states

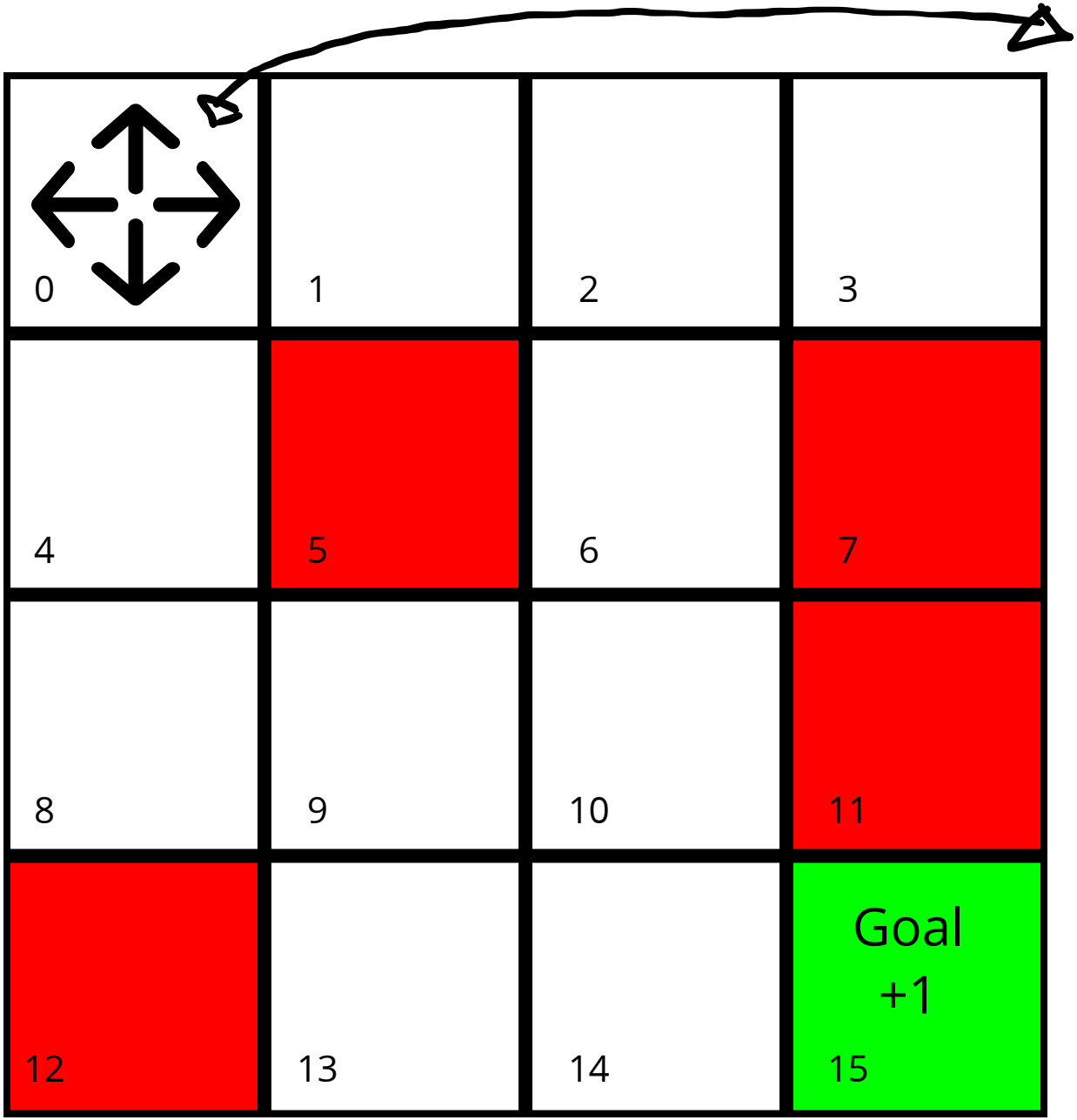


Action space; 4 Possible actions (Left, Down, Right or Up)



Action space; 4 Possible actions (Left, Down, Right or Up)

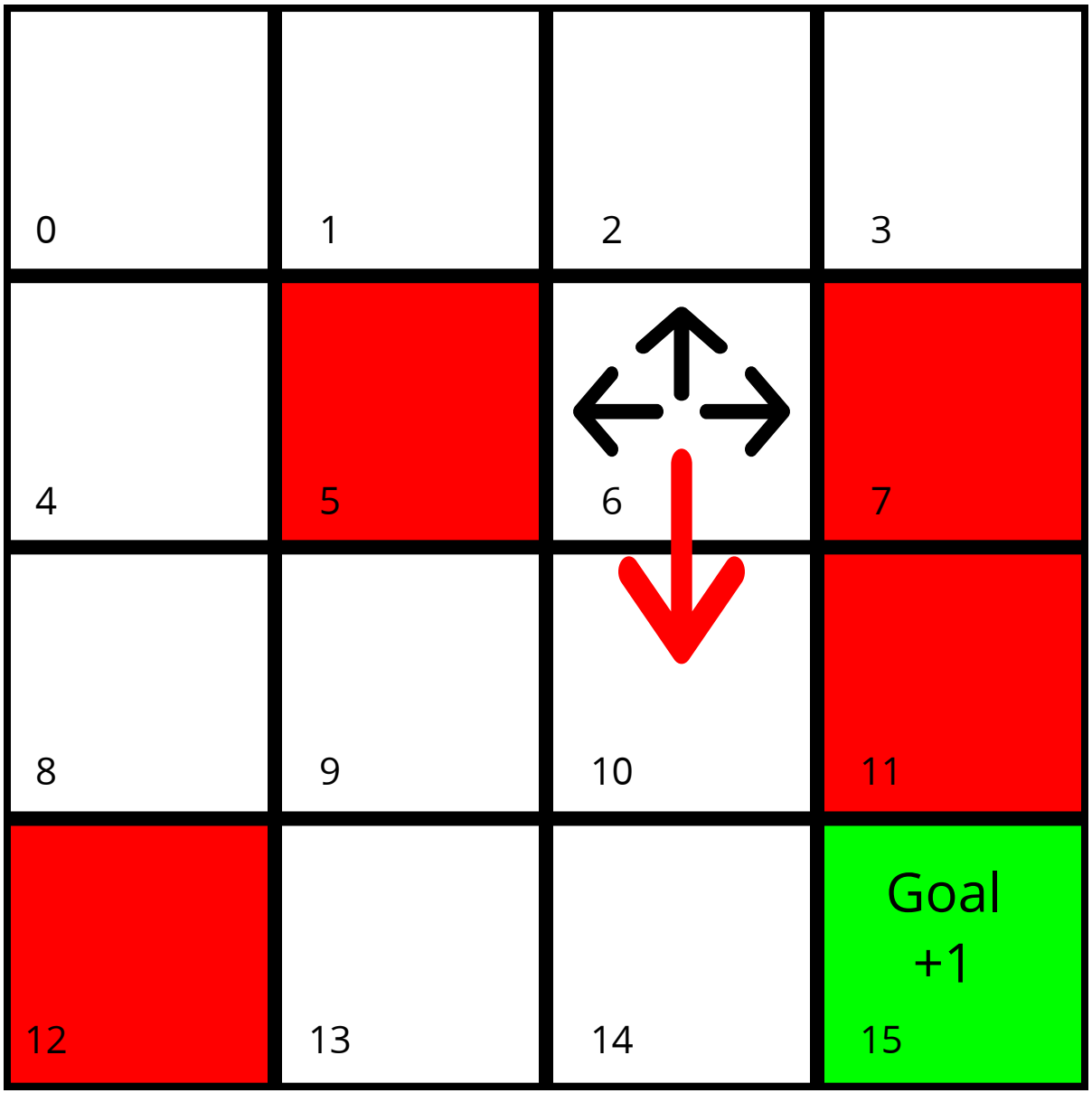
Stochastic environment; 66% chance of slipping and ending up ∓90° of intended action.

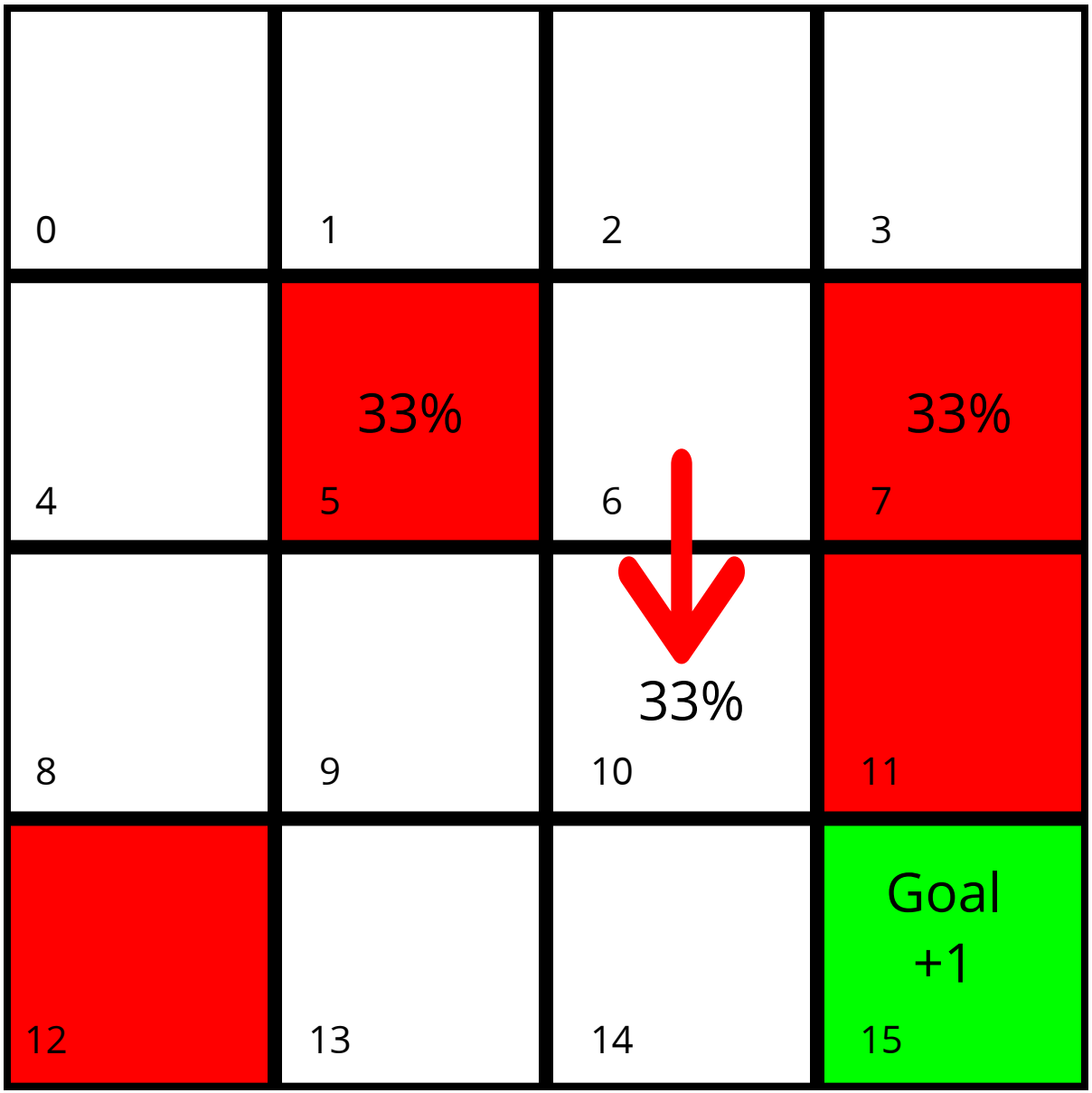


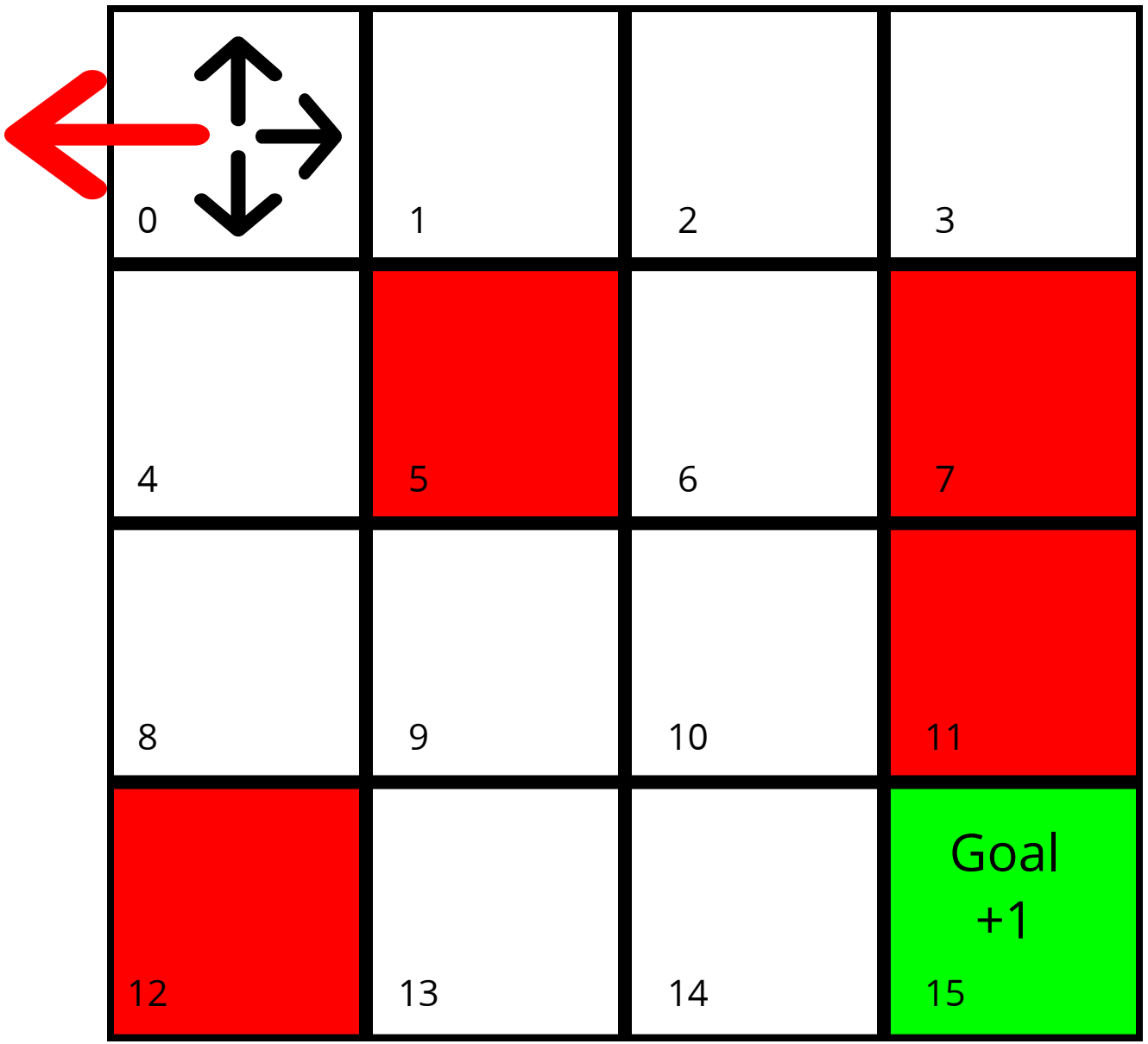
Action space; 4 Possible actions (Left, Down, Right or Up)

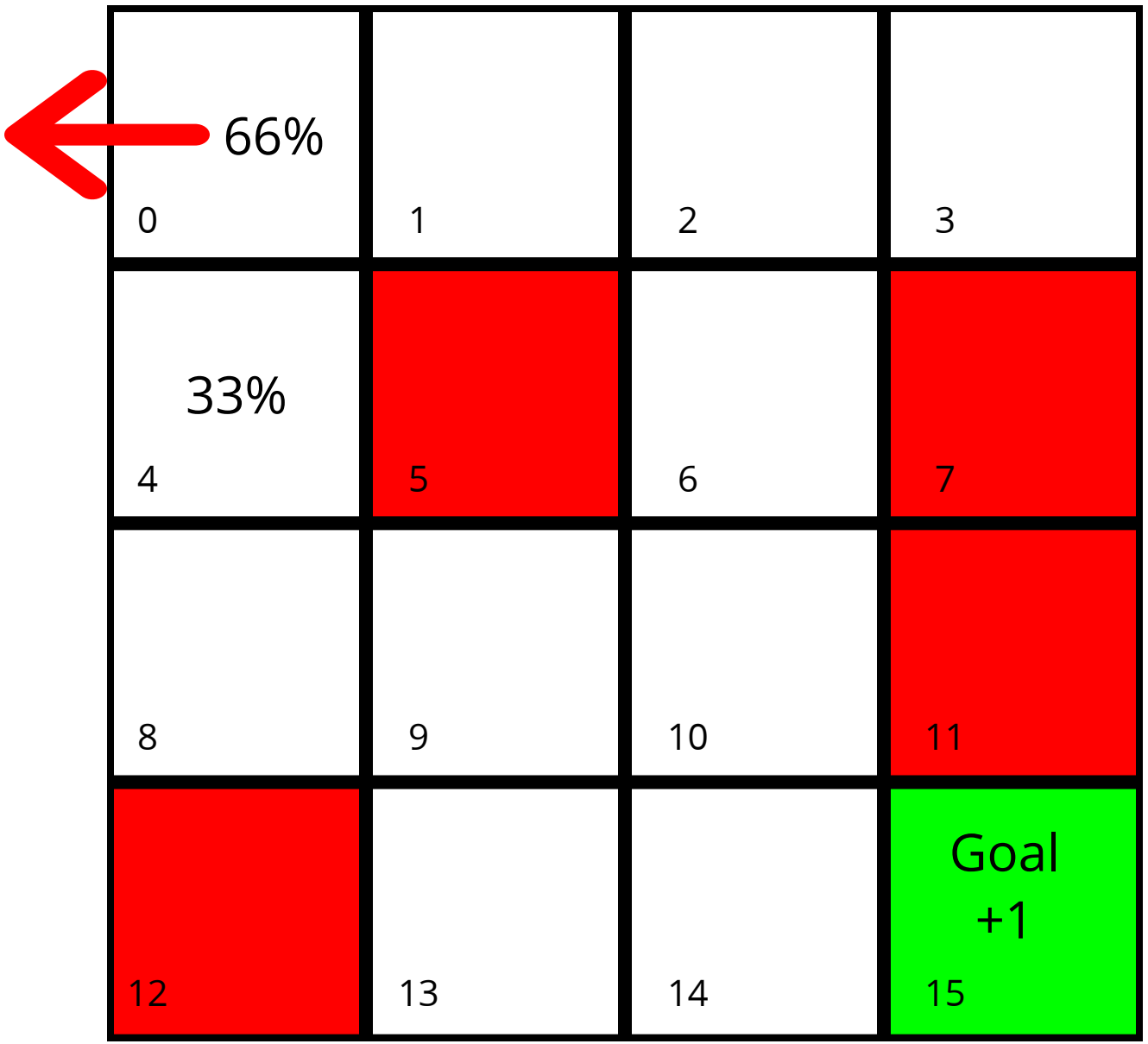
Stochastic environment; 66% chance of slipping and ending up ∓90° of intended action.

Walls; Walking into a wall will put you back to your original position


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15


0
1
2
3
4
33%
5
6
33%
7
33%
8
9
10
11
12
13
14
Goal
+1
15

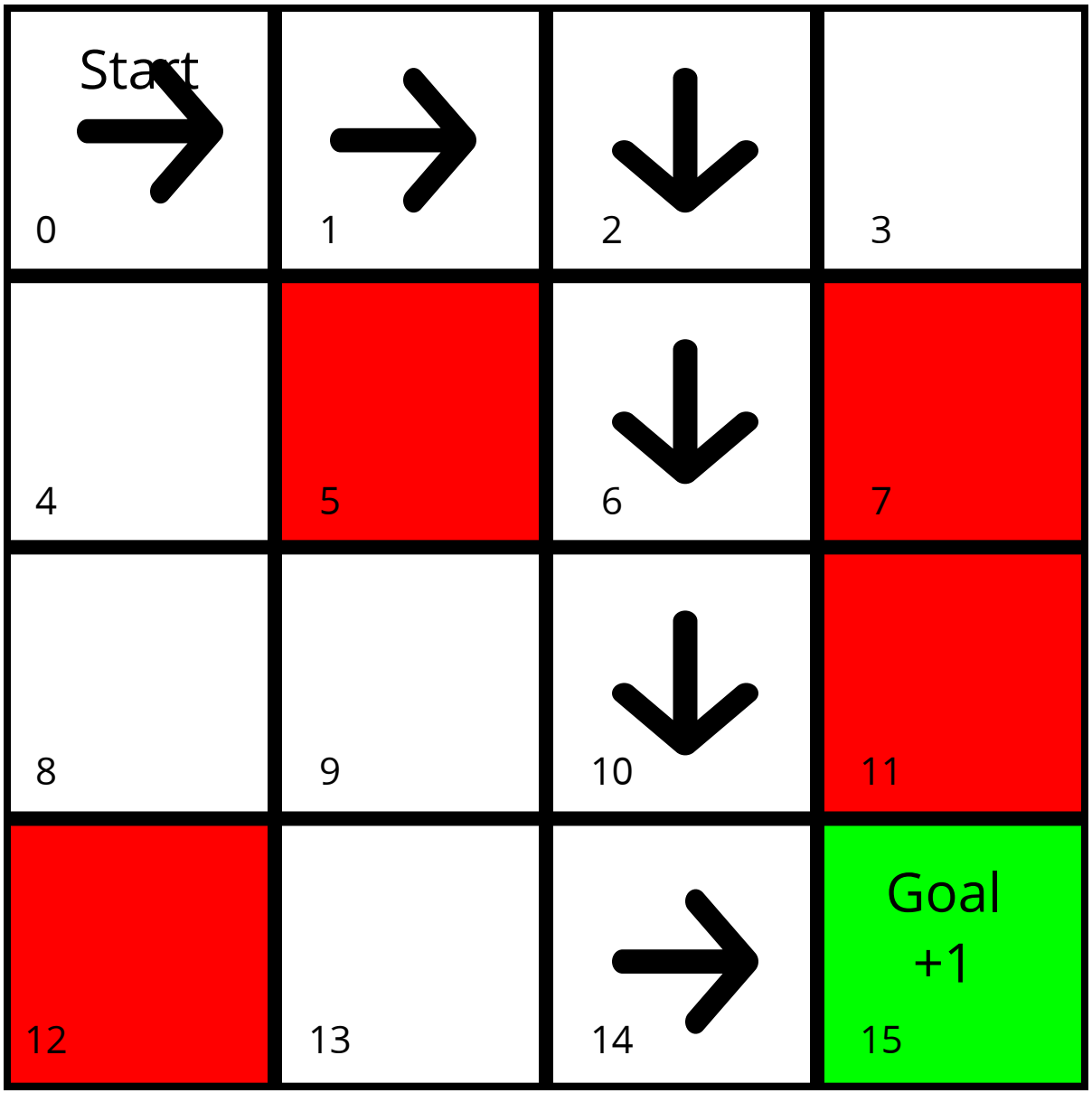

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15


66%
0
1
2
3
33%
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15

Properties of FrozenLake:

- Sequential decision problem
- Stochastic Environment
- Fully observable
- with a Markovian Transition model
- with additive rewards

These properties allow it to be modelled using the mathematical model known as Markov Decision Process (MDP)


Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15

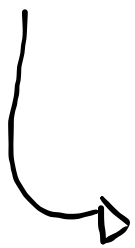
A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$

A *state* is a unique and self-contained configuration of the environment

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$



an action is what the agent does to affect the environment.

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$

Probability that action a taken from state s leads to s'

We expect the sum of the probabilities across all possible next states to sum to 1

$$\sum_{s' \in S} p(s'|s,a) = 1$$

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$

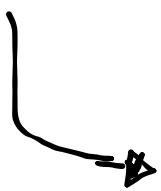
The transition function should be *Markovian*.

Action outcomes only depend on the current state.

$$p(s_{t+1}|s_t, a_t) = p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \ldots)$$

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$
- A reward function $R(s,a,s')$



the reward is a measure of how well the agent is doing.

A Markov Decision Process (MDP) is defined by:

- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$
- A reward function $R(s,a,s')$
- A start state S_0

A Markov Decision Process (MDP) is defined by:

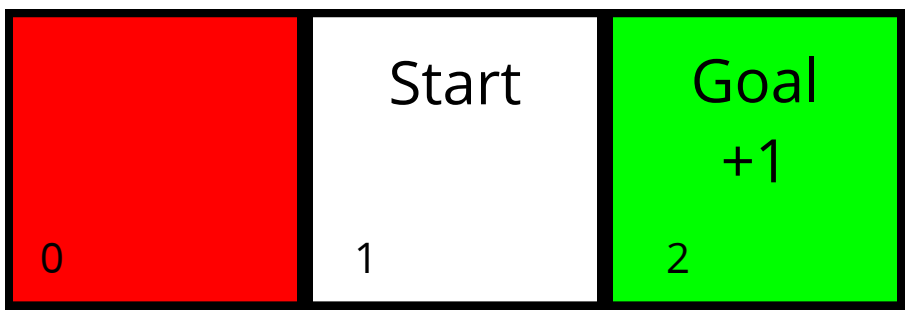
- A set of states $s \in S$
- A set of actions $a \in A$
- A transition model $P(s'|s,a)$
- A reward function $R(s,a,s')$
- A start state S_0

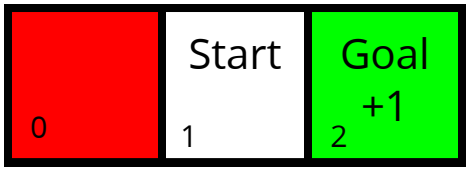
Sometimes

- A discount factor γ
- The horizon H

MDP's can be considered non-deterministic search problems

Graphical representations of MDPs





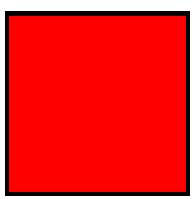
Actions:

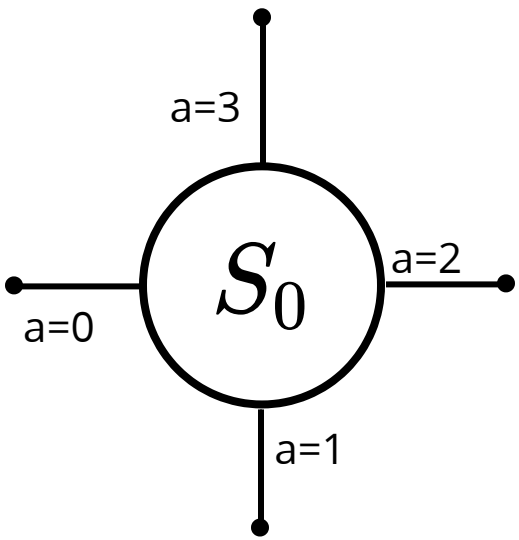
Left = 0

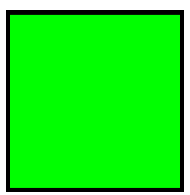
Down = 1

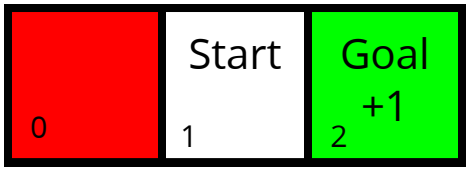
Right = 2

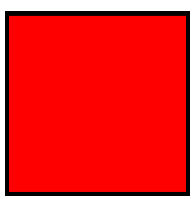
Up = 3

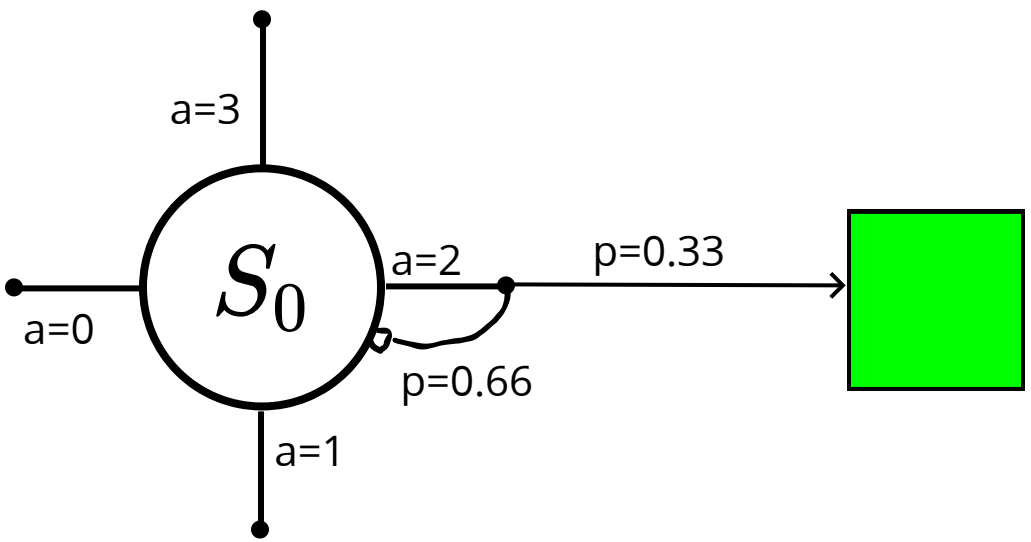


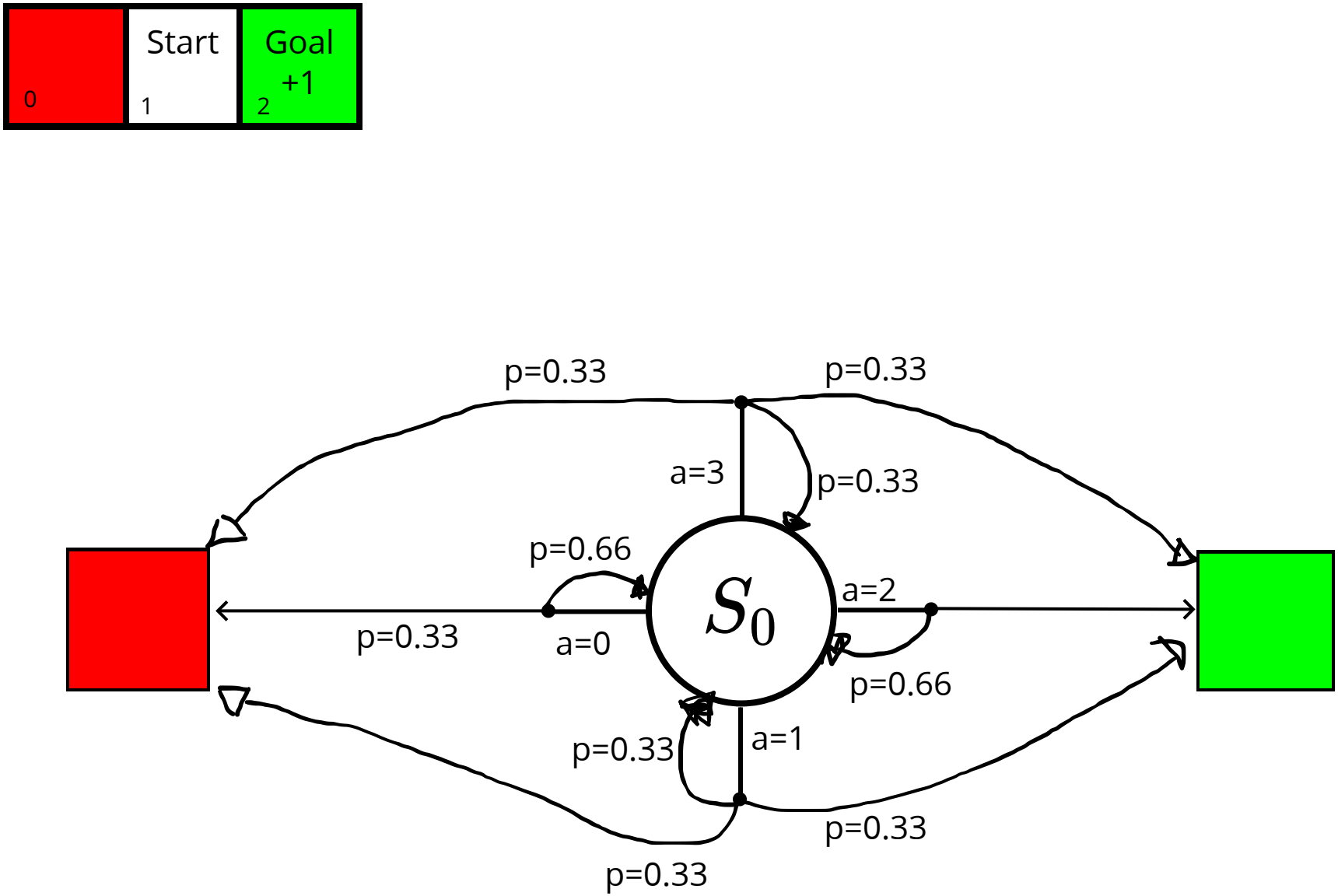


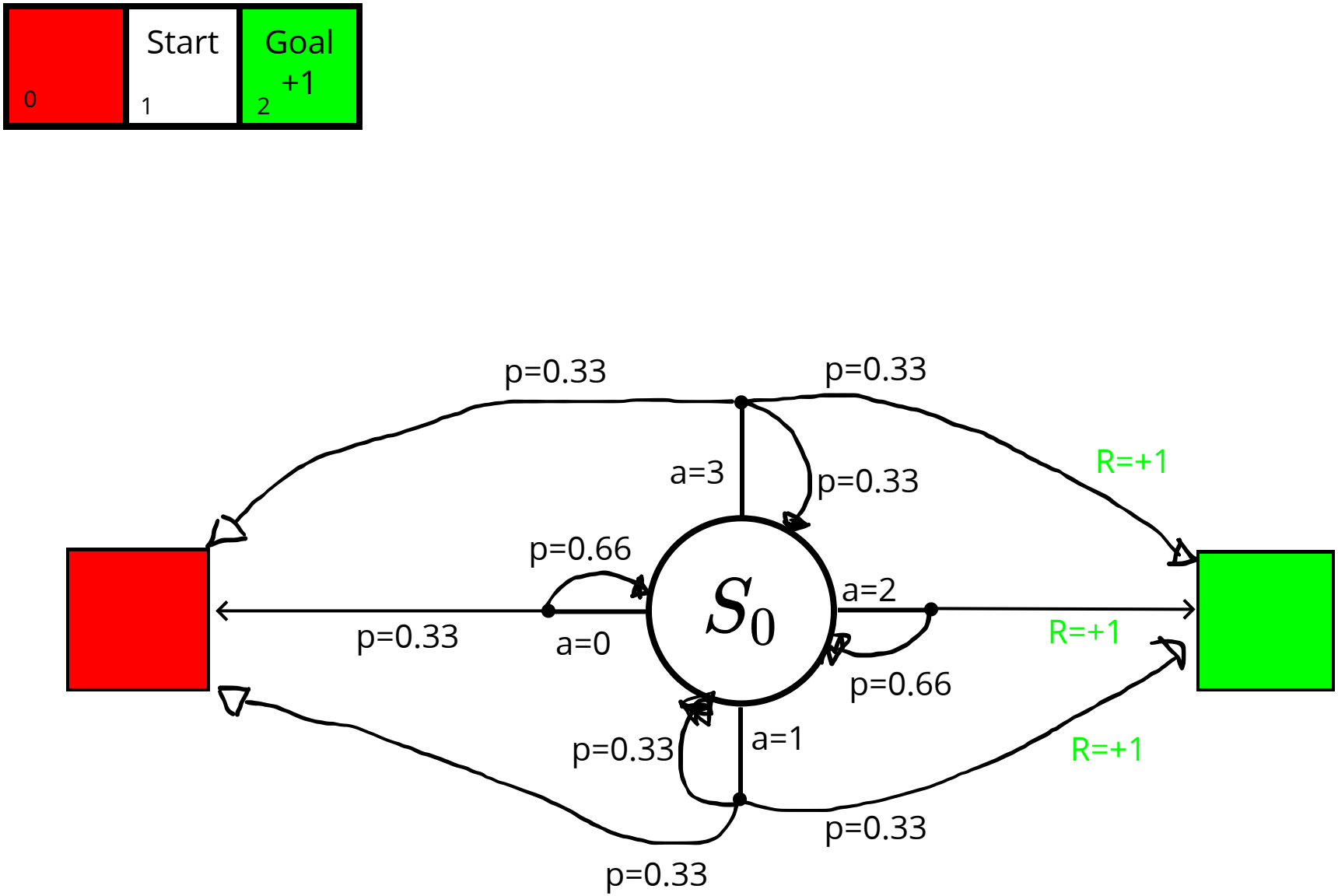



Start
Goal
+1
0
1
2




a=3
S_0
a=2
p=0.33
a=0
p=0.66
a=1


Start
Goal
+1
0
1
2
p=0.33
p=0.33
a=3
p=0.33
p=0.66
S_0
a=2
p=0.33
a=0
p=0.66
p=0.33
a=1
p=0.33
p=0.33


Start
Goal
+1
0
1
2
p=0.33
p=0.33
a=3
p=0.33
R=+1
p=0.66
S_0
a=2
p=0.33
a=0
R=+1
p=0.66
p=0.33
a=1
R=+1
p=0.33
p=0.33

Solving the MDP: The RL agent

An RL agent may include one or more of the following components:

- Policy: Agents' plan/behavior.
- Utility function: A function to evaluate how good/bad a state is.
- Model: A model predicts what the environment will do next.

RL methods can be classified as either utility-based or policy-based and as either model-based or model-free.

Policy

Recall: "A policy defines an agents' behavior"

It is a map from a state to an action, and is denoted π

For MDPs we want a plan for **all** possible states.

Formally:

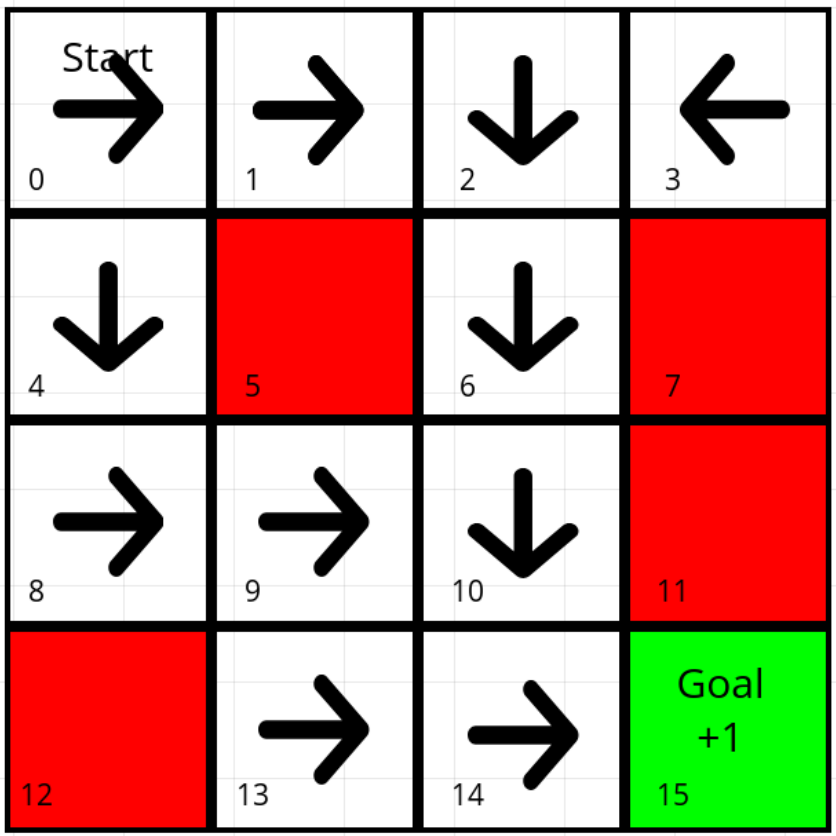
A policy: $\pi : S \rightarrow A$

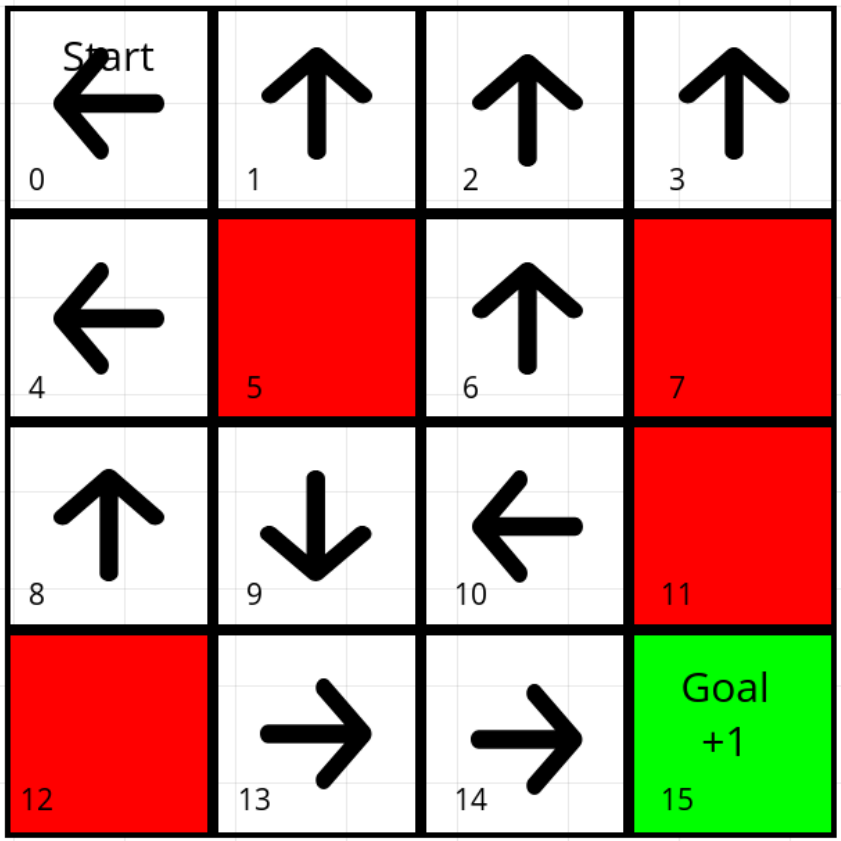
gives an action for each state

An optimal policy: $\pi^* : S \rightarrow A$

gives an action for each state that maximizes the expected utility.

Example policies



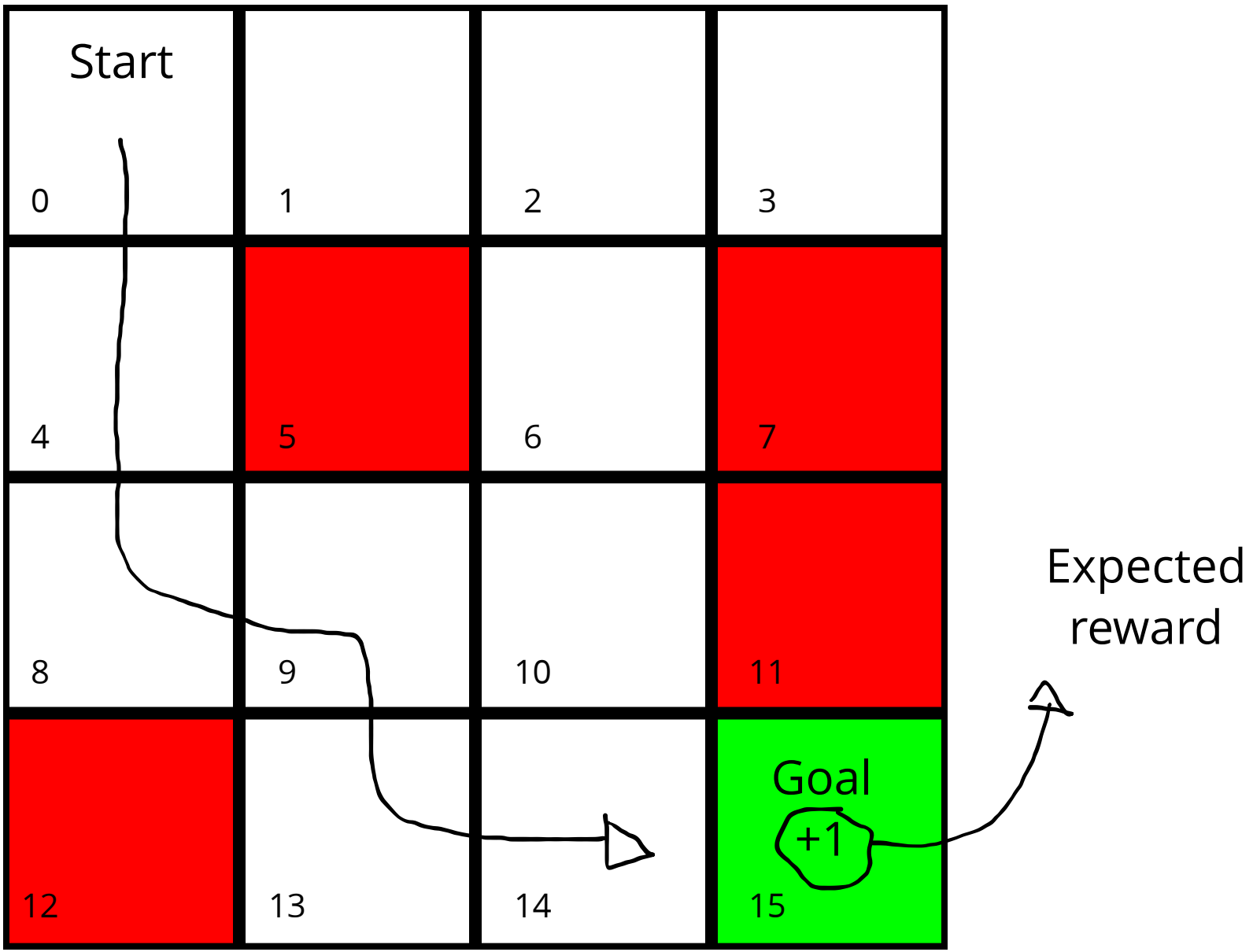


Utility Value

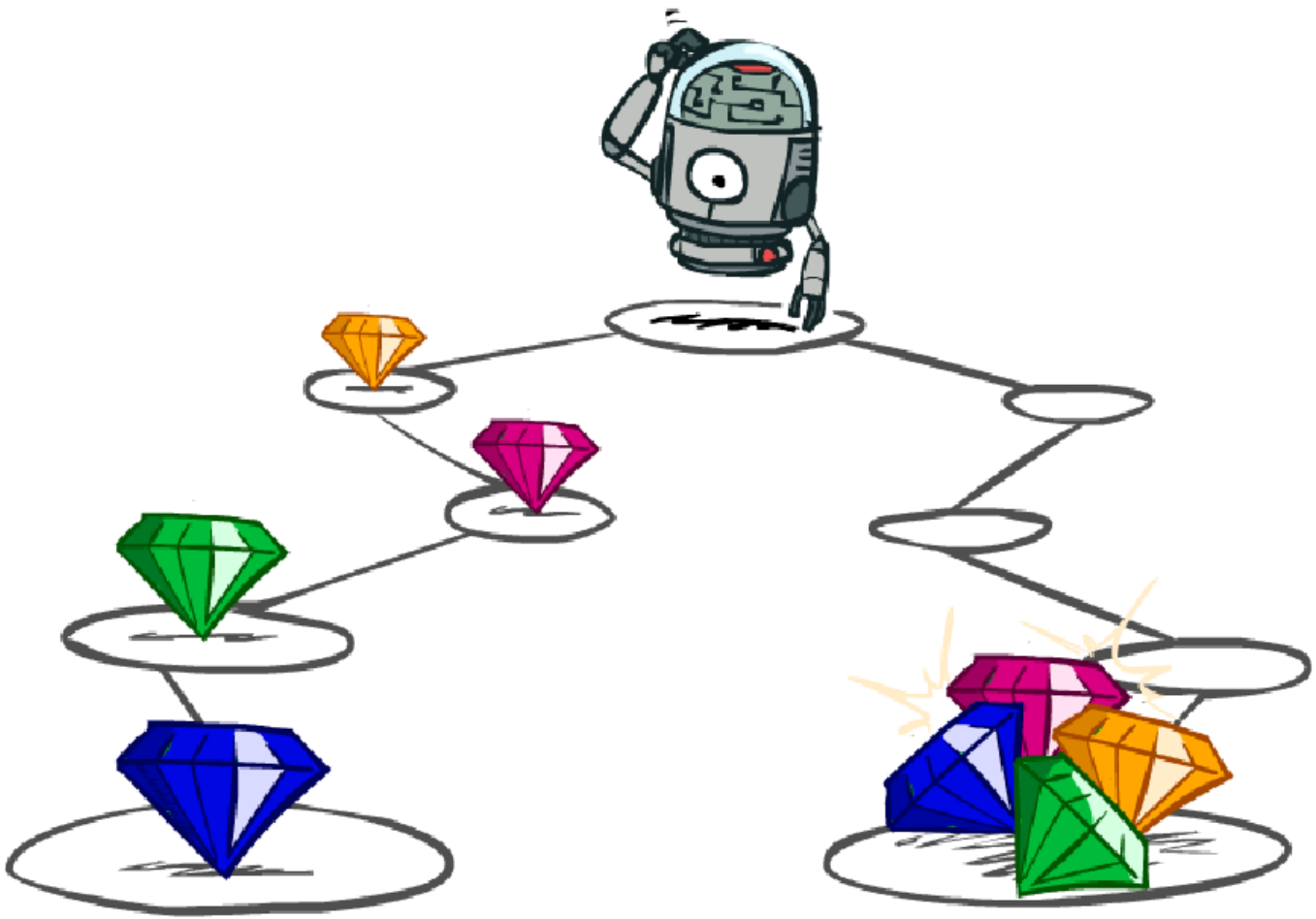
In Reinforcement Learning, the utility is an expectation of the long-term reward.

The utility can be used to:

- Evaluate the desirability of states
- Select between different actions


Start
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
Goal
+1
15
Expected
reward

Utilities of Sequences



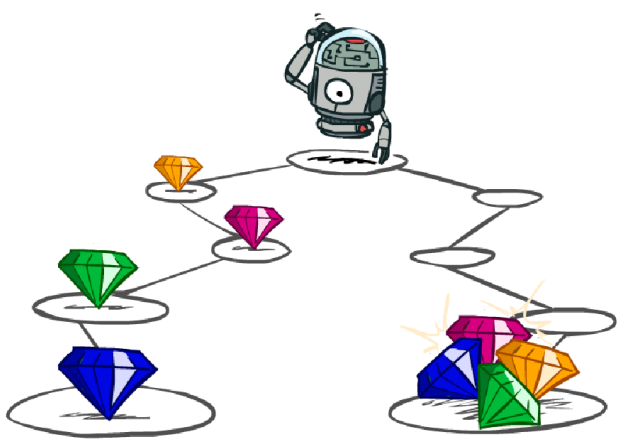
Utlities of Sequences

What preferences should an agent have over reward sequences?

- More or less? [1,2,2] vs [2,3,4]

  What if more increases risk?

- Now or later? [0,0,1] vs [1,0,0]



Utlities of Sequences

Utlities of Sequences

What preferences should an agent have over reward sequences?

- More or less? [1,2,2] vs [2,3,4]

  What if more increases risk?

- Now or later? [0,0,1] vs [1,0,0]

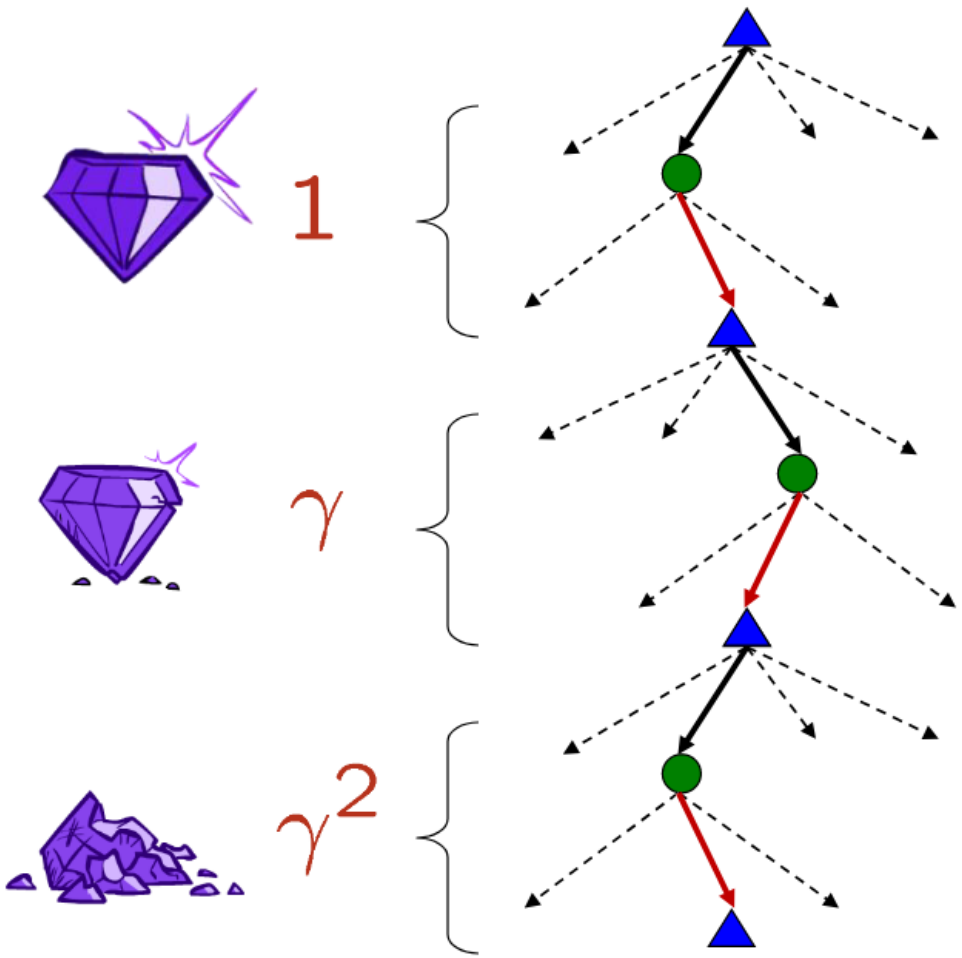
It is generally reasonable to maximize the sum of rewards.

It is also reasonable to prefer rewards now to rewards later.

One solution is to let the values decay exponentially for every timestep by discounting the reward.

Discounting

- We introduce a discount factor $\gamma \in [0, 1]$
- The discount factor trades off the importance of immediate vs. long-term reward
- For each timestep, we multiply the discount once



Example, $\gamma = 0.5$:

$$U_h([1, 2, 3]) = 1 * \gamma^0 + 2 * \gamma^1 + 3 * \gamma^2$$

$$U_h([1, 2, 3]) = 1 * 1 + 2 * 0.5 + 3 * 0.25$$

$$U_h([1, 2, 3]) < U_h([3, 2, 1])$$

Finite/Infinite Horizon

Problem: What if the game lasts forever? Do we get infinite rewards?

A Solution: Finite Horizon:

- Terminate episodes after a fixed T steps (or set all rewards to 0)
- Policy π depends on how much time is left

A policy that depends on the time is called nonstationary.

A policy that does not depend on the time is called stationary.

Discussion

Could you solve the problem with the boat agent above by changing the horizon or the discount value?

Assume:

- Getting first place would, on average, reward more points than looping for the entire game.
- Getting second place would, on average, reward fewer points than looping for the entire game.
- The current horizon is set to when the second-to-last player reaches the goal + 100 steps.

Additive Discounted Rewards

Imagine that you are following some policy π, thus producing a trajectory of states, actions, and rewards $(s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, ...)$

We can calculate the utility of a trajectory using additive discounted rewards:

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

$$= \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})$$

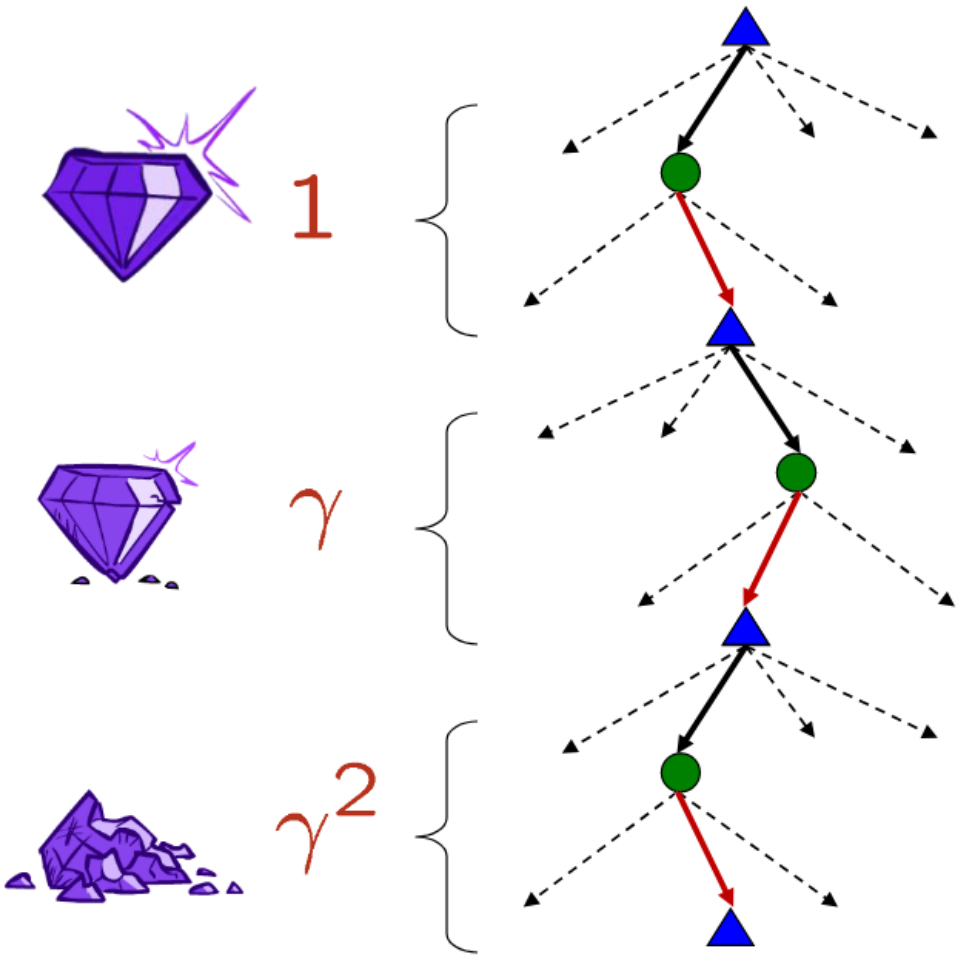
Additive Discounted Rewards

Example, $\gamma = 0.5$:

$U_h([1,2,3]) = 1 * \gamma^0 + 2 * \gamma^1 + 3 * \gamma^2$

$U_h([1,2,3]) = 1 * 1 + 2 * 0.5 + 3 * 0.25$

$U_h([1,2,3]) < U_h([3,2,1])$



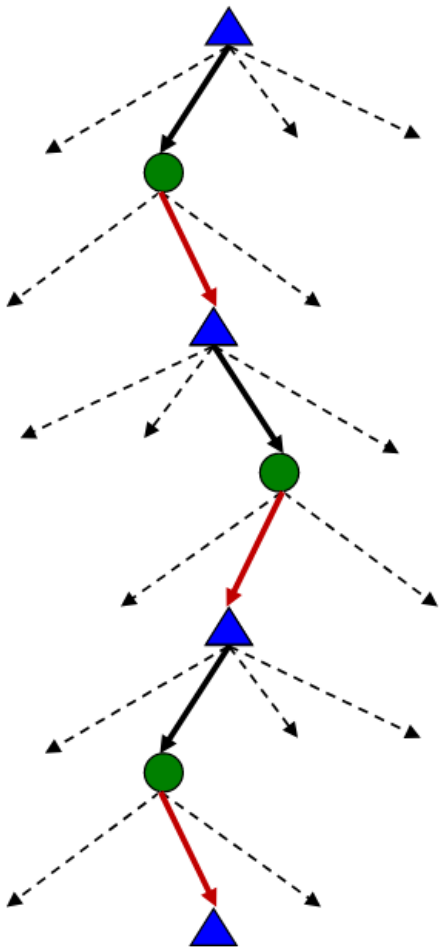
Utility Function

The expected reward from following a policy π starting in state s is given by the function

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

Here the expectation E is with respect to the probability distribution over the state sequences determined by s and π

S_t is the state the agent reaches by time t by following the policy π



Utility Function

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

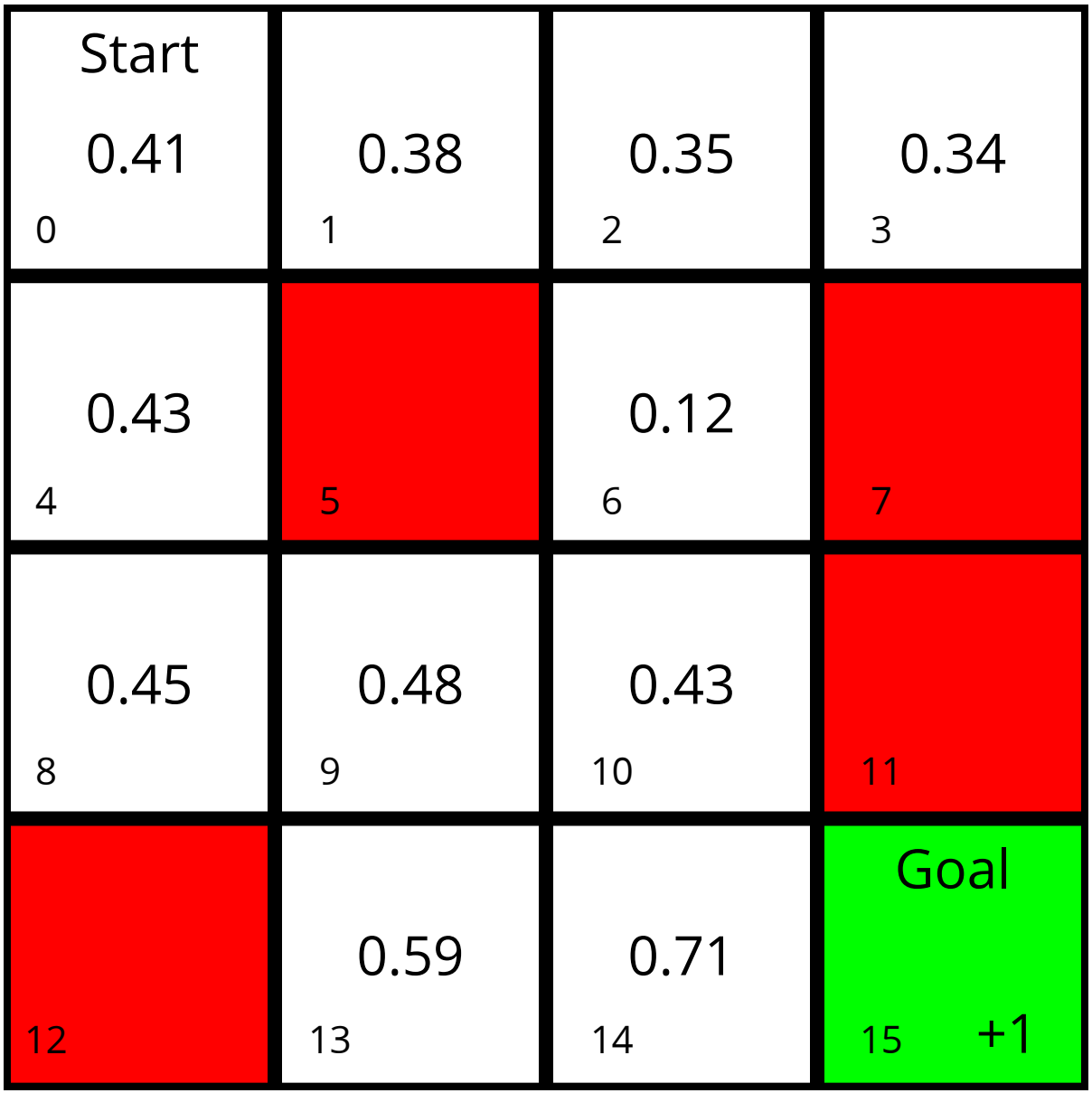
Utility Function:

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

This is also called a Bellman Equation, after Richard E. Bellman

Optimal policy:

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


Start
0.41
0
0.38
1
0.35
2
0.34
3
0.43
4
5
0.12
6
7
0.45
8
0.48
9
0.43
10
11
12
0.59
13
0.71
14
Goal
15
+1

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

With $\gamma = 0.9$

$$U(10) = max\{[0.33(0 + \gamma U(6)) + 0.33(0 + \gamma U(9))0.33(0 + \gamma U(14))],$$
$$[0.33(0 + \gamma U(9)) + 0.33(0 + \gamma U(14))0.33(0 + \gamma U(11))],$$
$$[0.33(0 + \gamma U(14)) + 0.33(0 + \gamma U(11))0.33(0 + \gamma U(6))],$$
$$[0.33(0 + \gamma U(11)) + 0.33(0 + \gamma U(6))0.33(0 + \gamma U(9))]\}$$

$$U(10) = max\{[0.33(0.9 * 0.12) + 0.33(0.9 * 0.48)0.33(0.9 * 0.71)], ...\}$$
$$U(10) = max\{[0.39], [0.35], [0.25], [0.18]\}$$
$$U(10) = 0.39$$

$\pi^*(s) = 0$, which maps to "*left*"

Q-Function

(Action-utility function)

Q-Function w/r to the utility function:

$$U(s) = \max_a Q(s, a)$$

Optimal policy w/r the Q-function:

$$\pi^*(s) = \arg\max_a Q(s, a)$$

The Q-function as a Bellman Equation:

$$Q(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma \max_{a'} Q(s', a')]$$

With $\gamma = 0.9$

$$Q(10, 0) = 0.33(0 + 0.90.12) + 0.33(0 + 0.9 * 48) + 0.33(0 + 0.9 * 71)$$

$$= 0.39$$

Expected utility after taking action 0 (left) in state 10.

Utility Functions in Reinforcement Learning

- Agents often approximate utility functions.
- With an accurate utility function, we can behave optimally.
- With suitable approximations, we can act well, even in intractably large domains.
- We will discuss and learn some algorithms later in the course.

- Introduction to Reinforcement Learning
- Markov Decision Processes (MDPs)