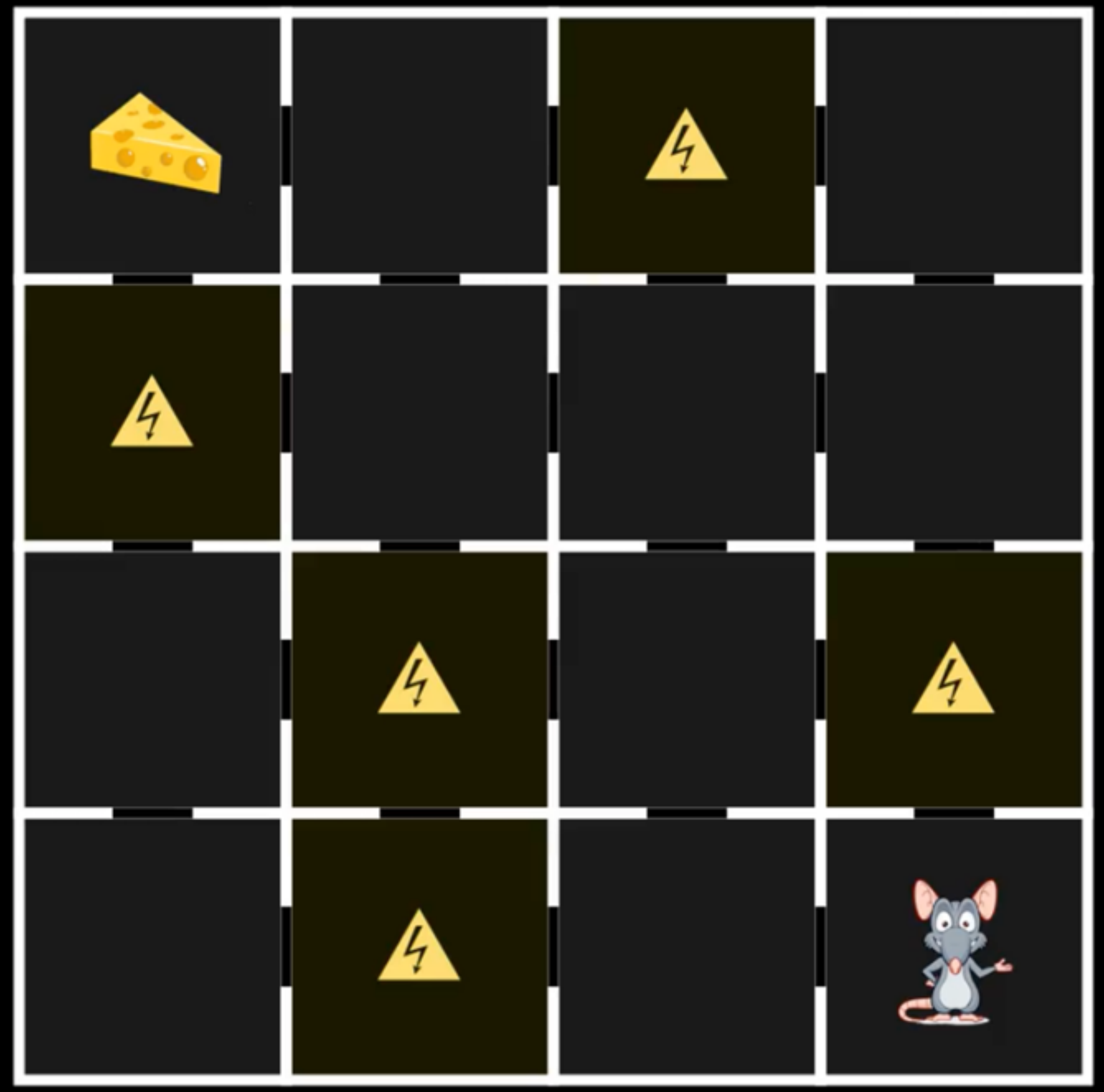
Lecture 31: Introduction to Reinforcement Learning

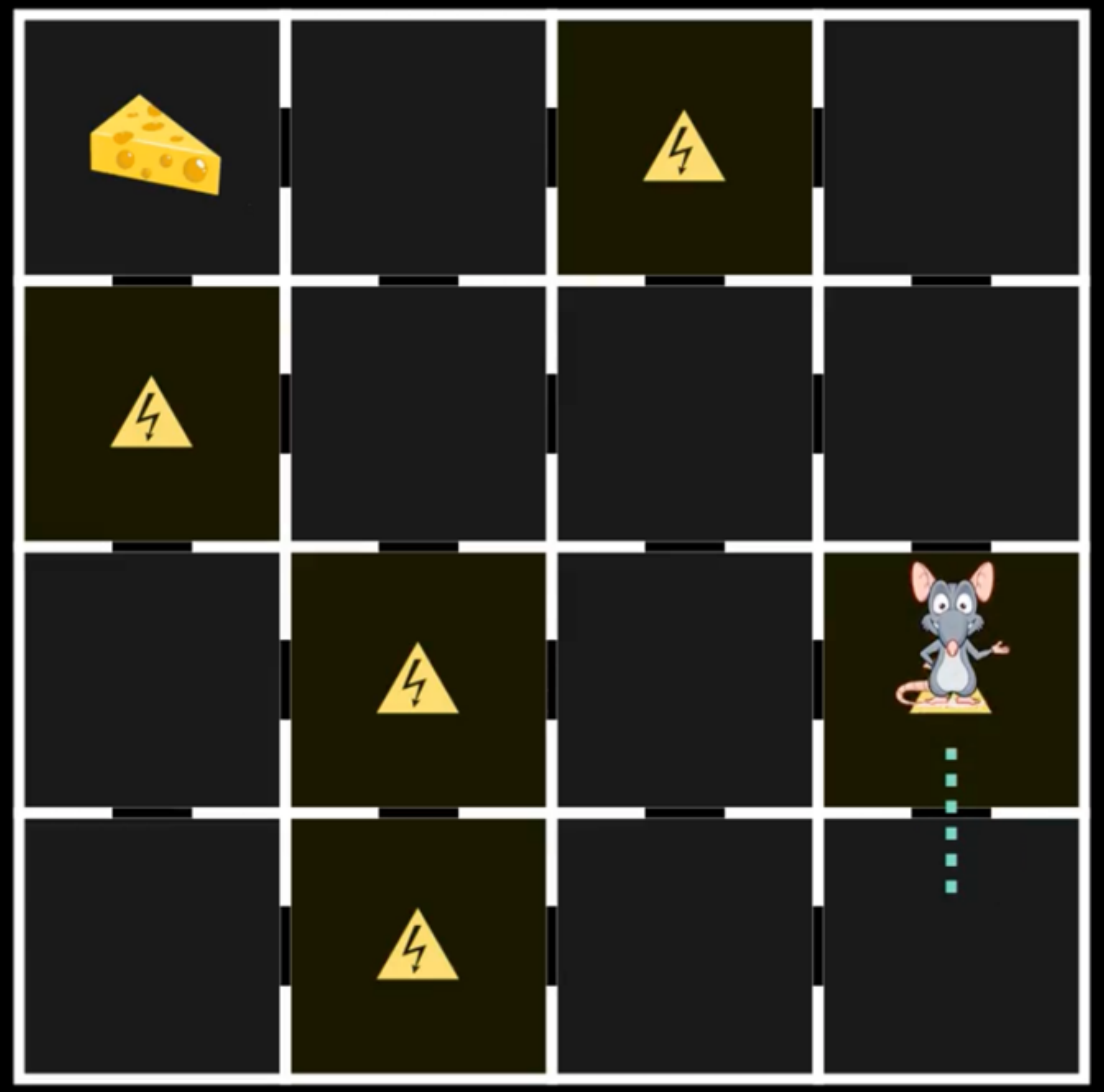
CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner

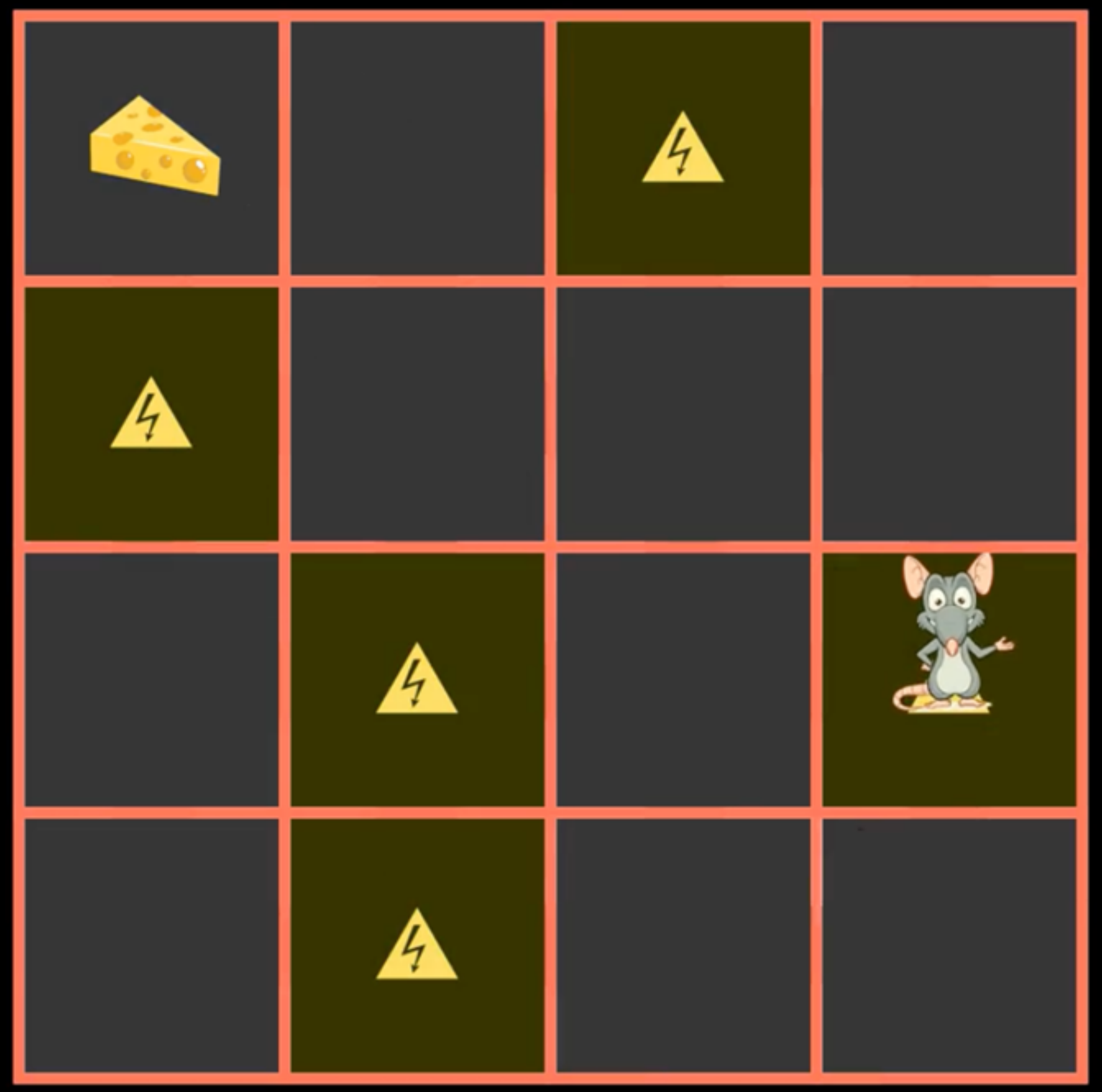
The Basic Intuition



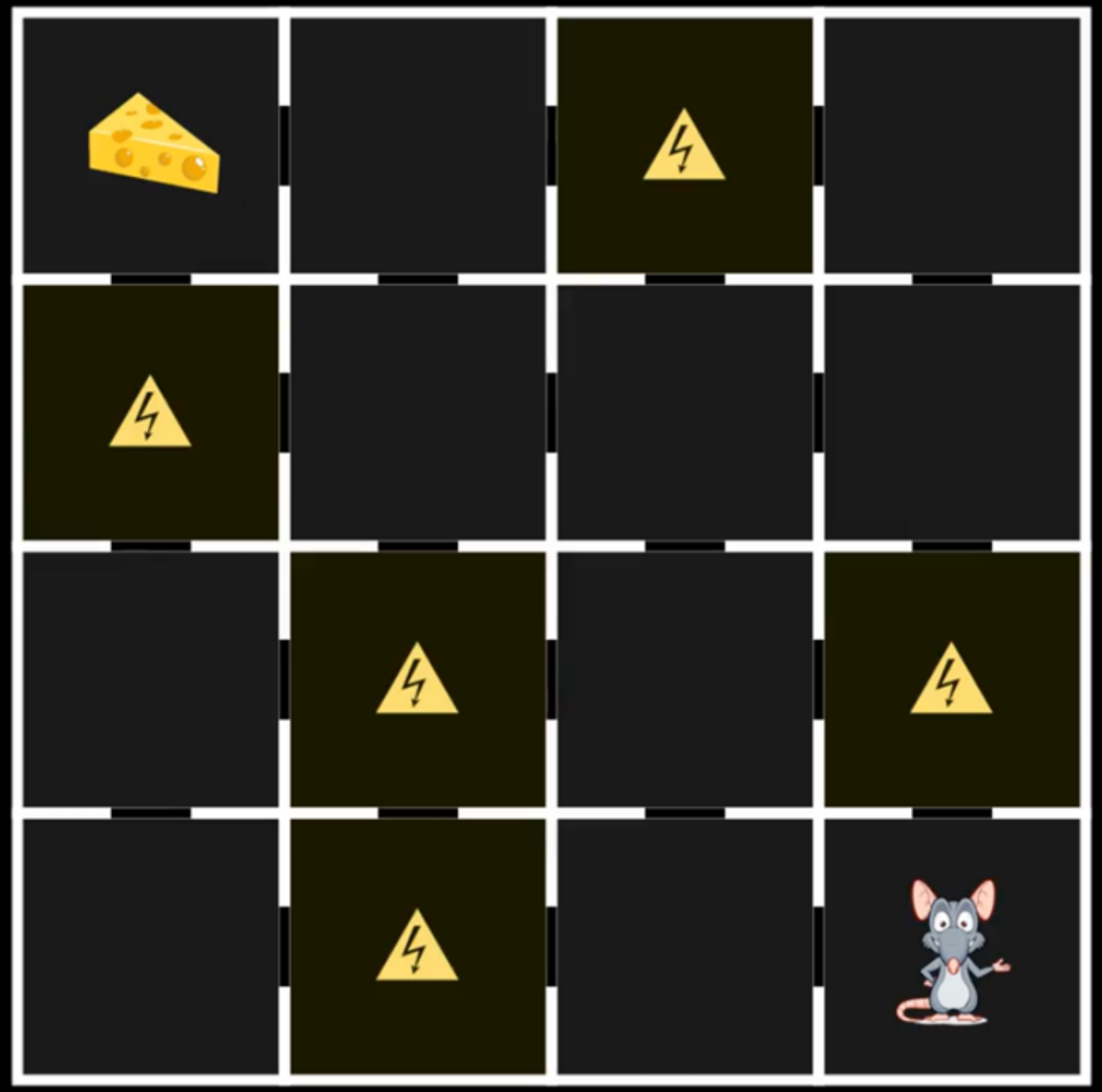
Attempt 1



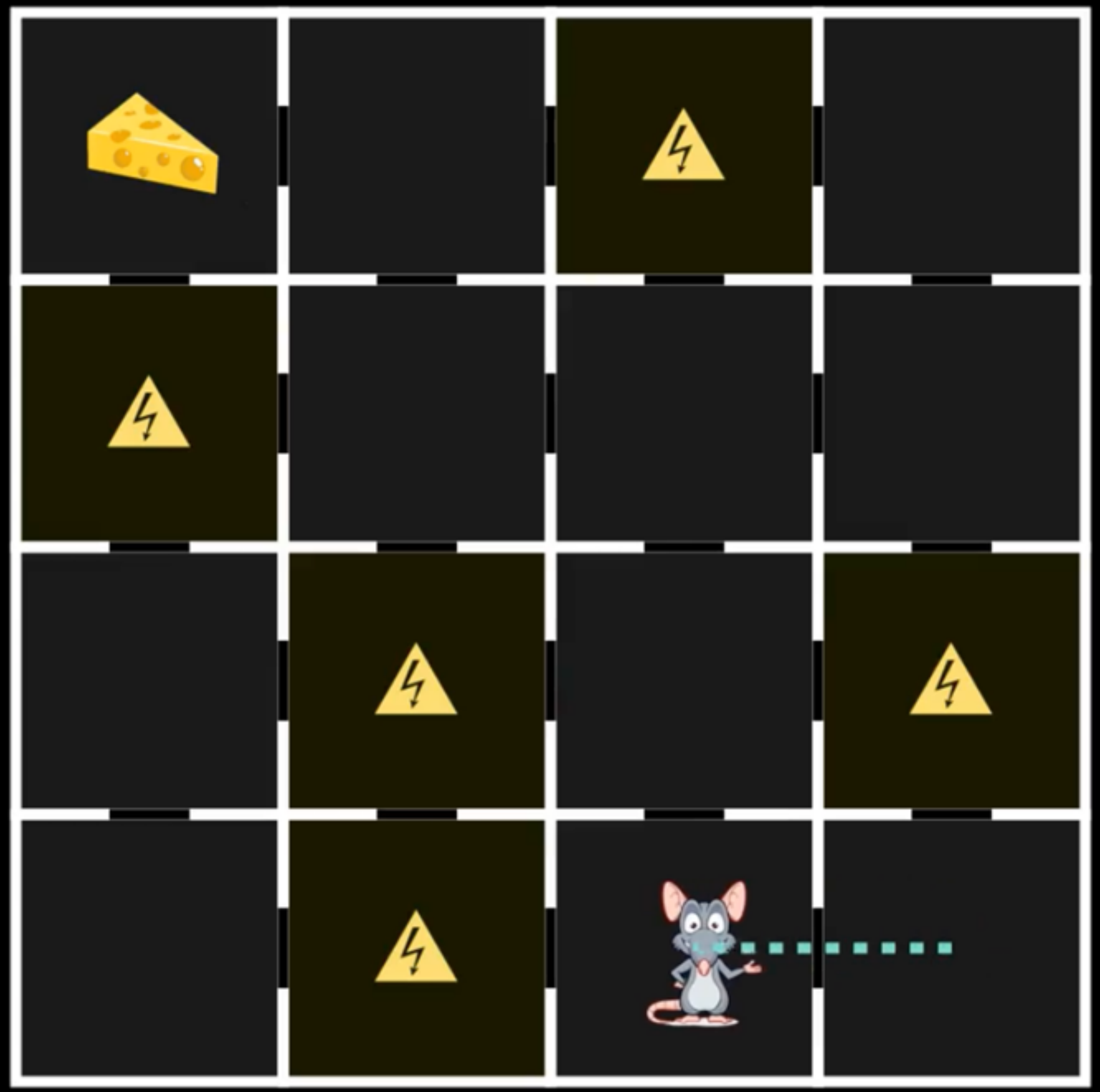
Attempt 1



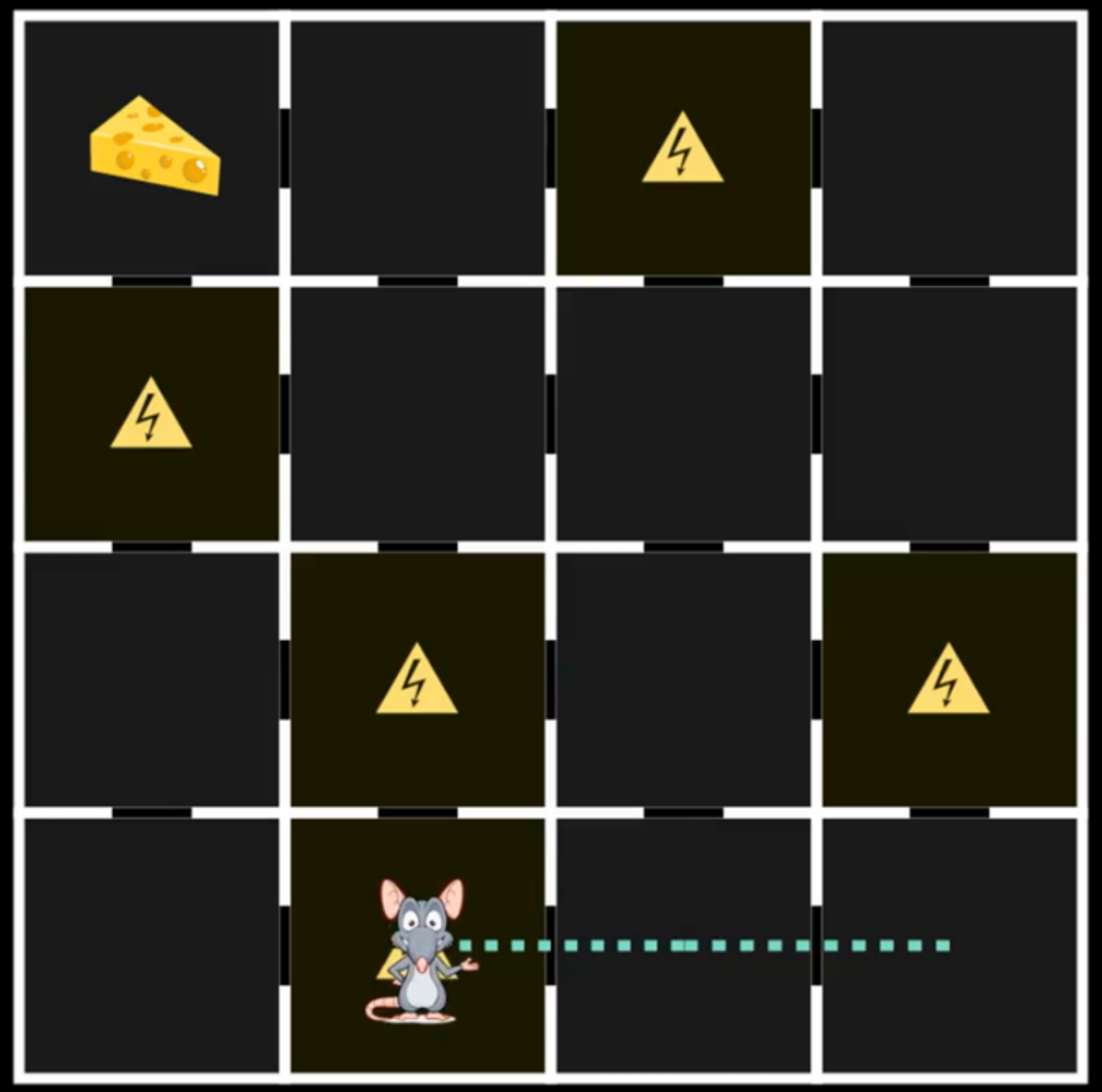
Attempt 2



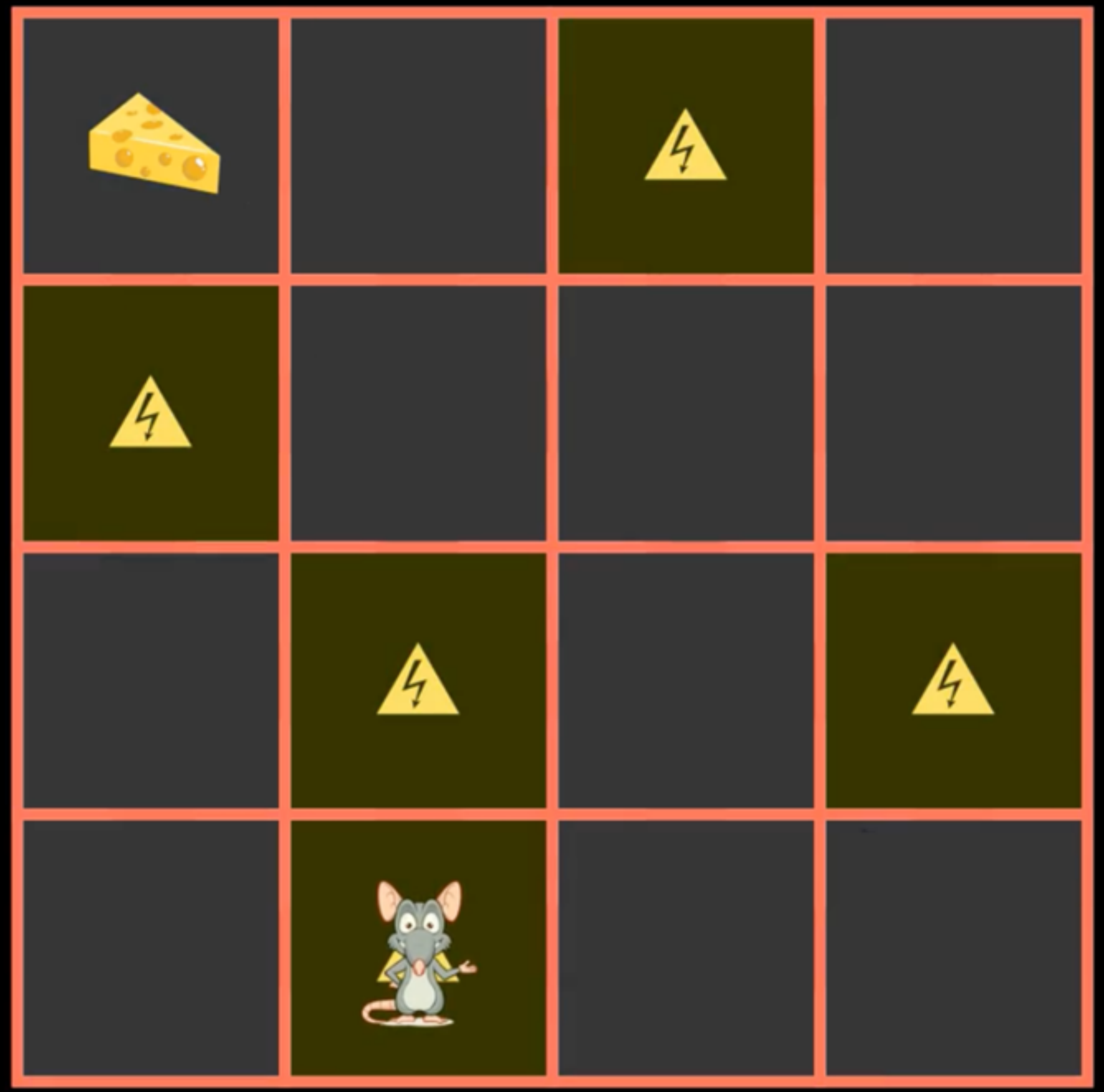
Attempt 2



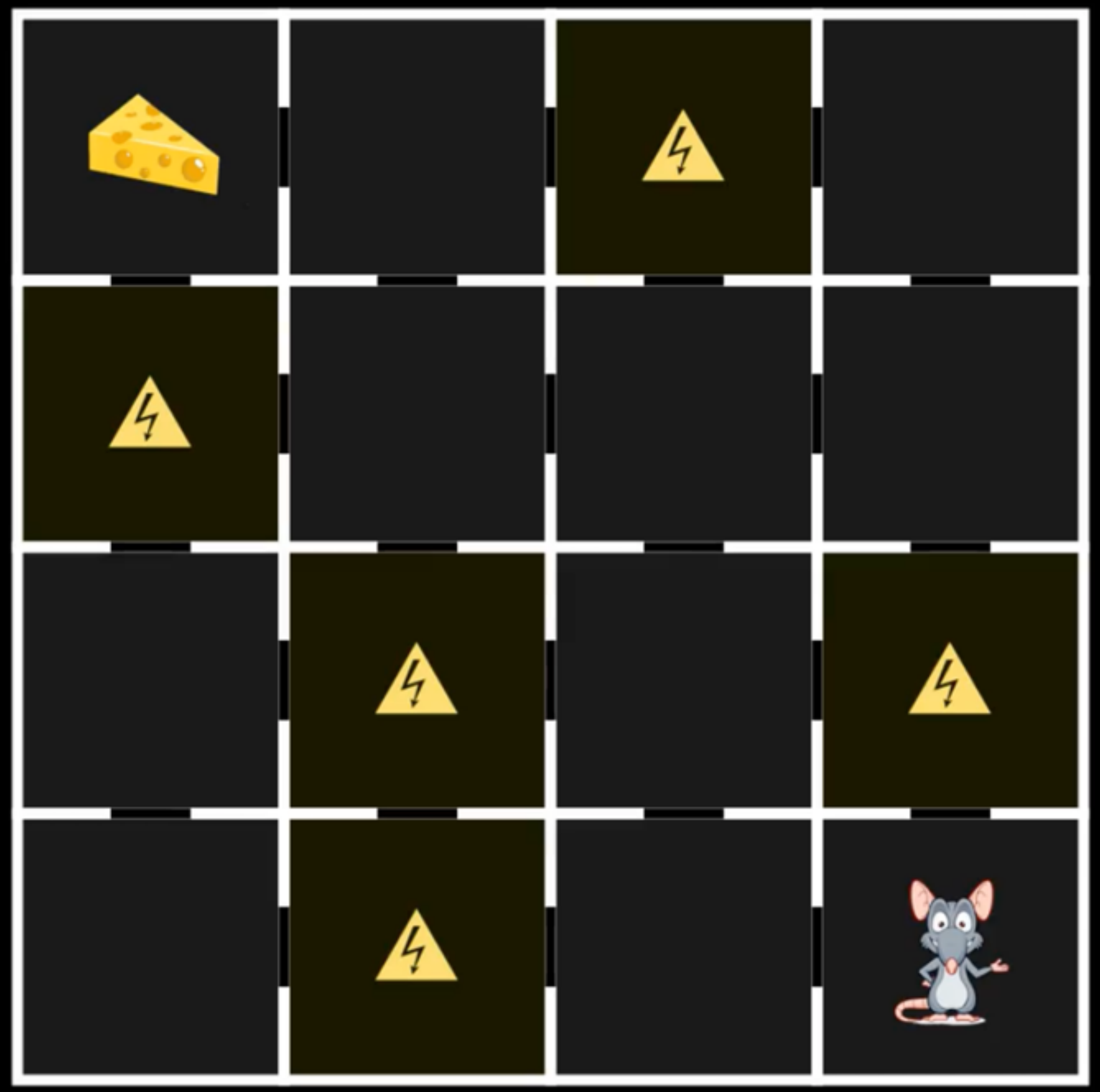
Attempt 2



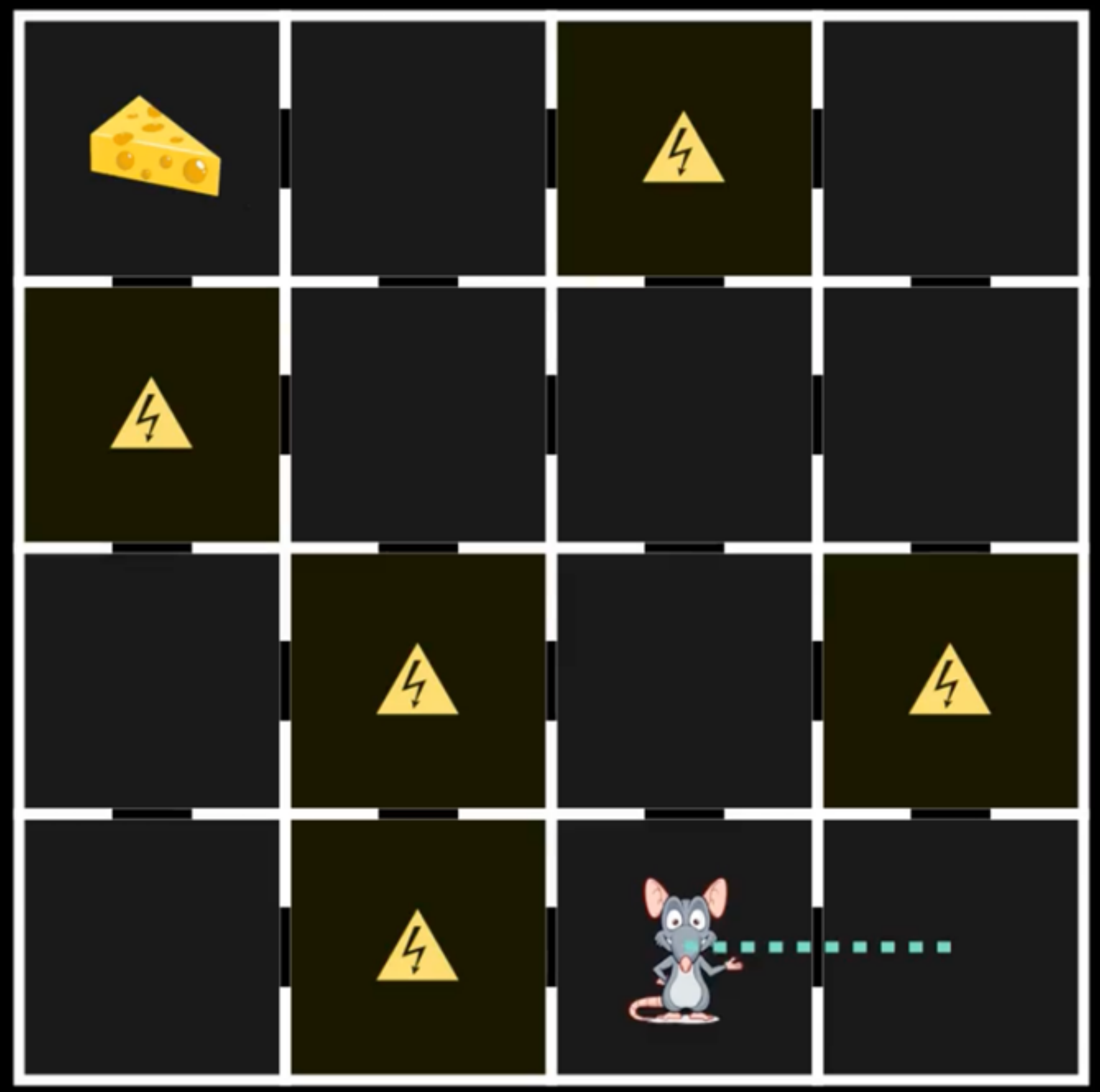
Attempt 2



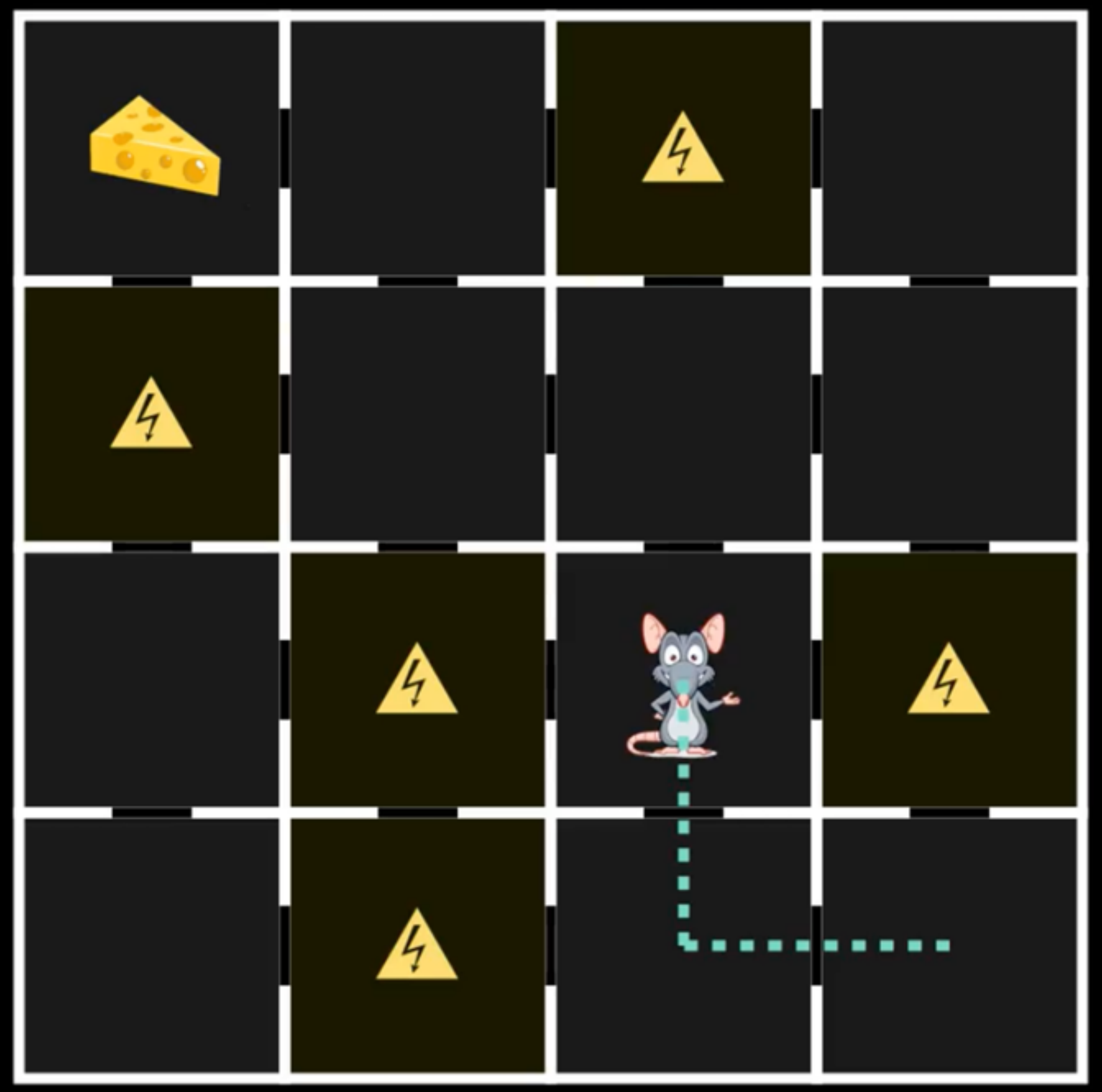
Attempt 3



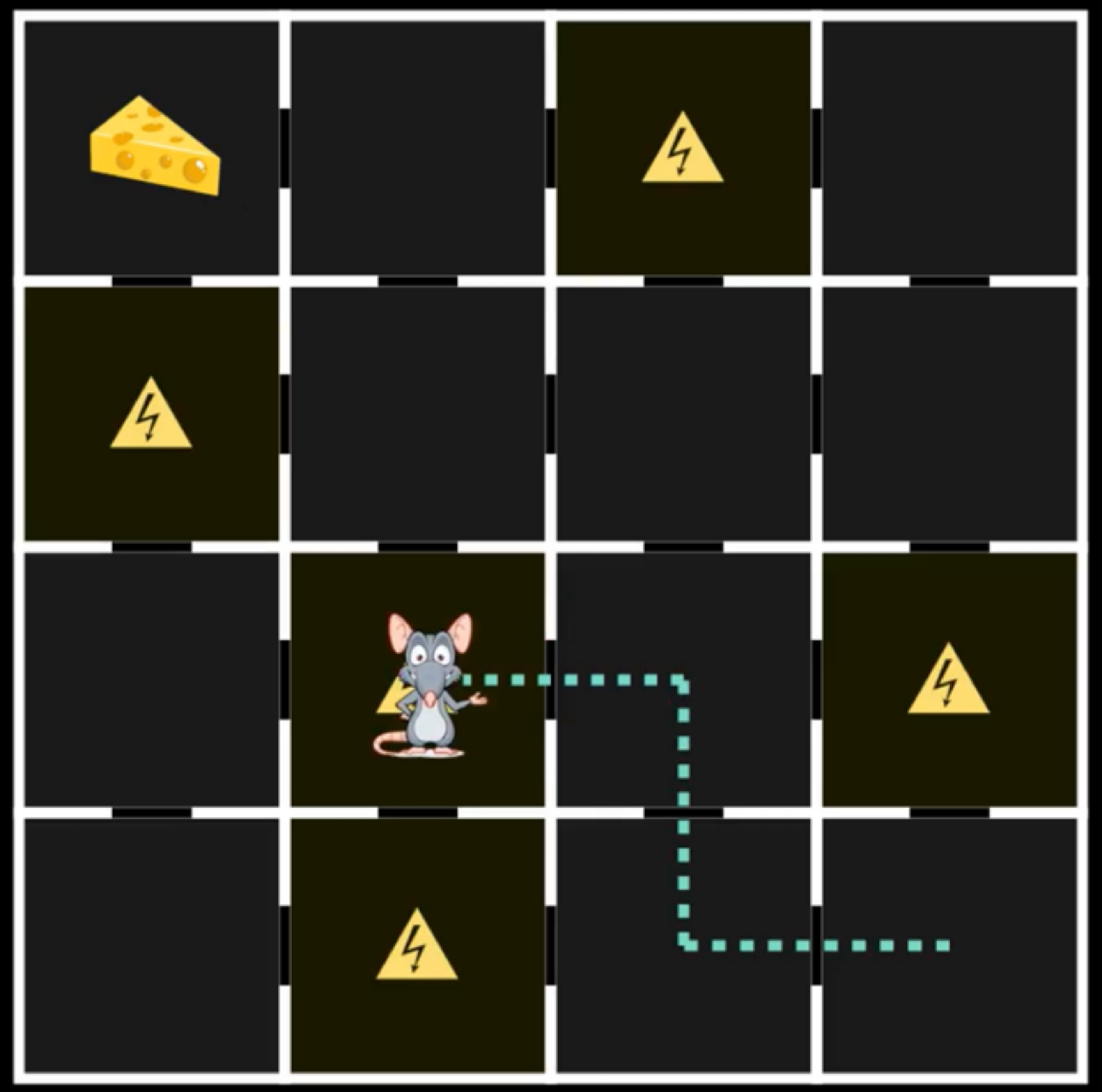
Attempt 3



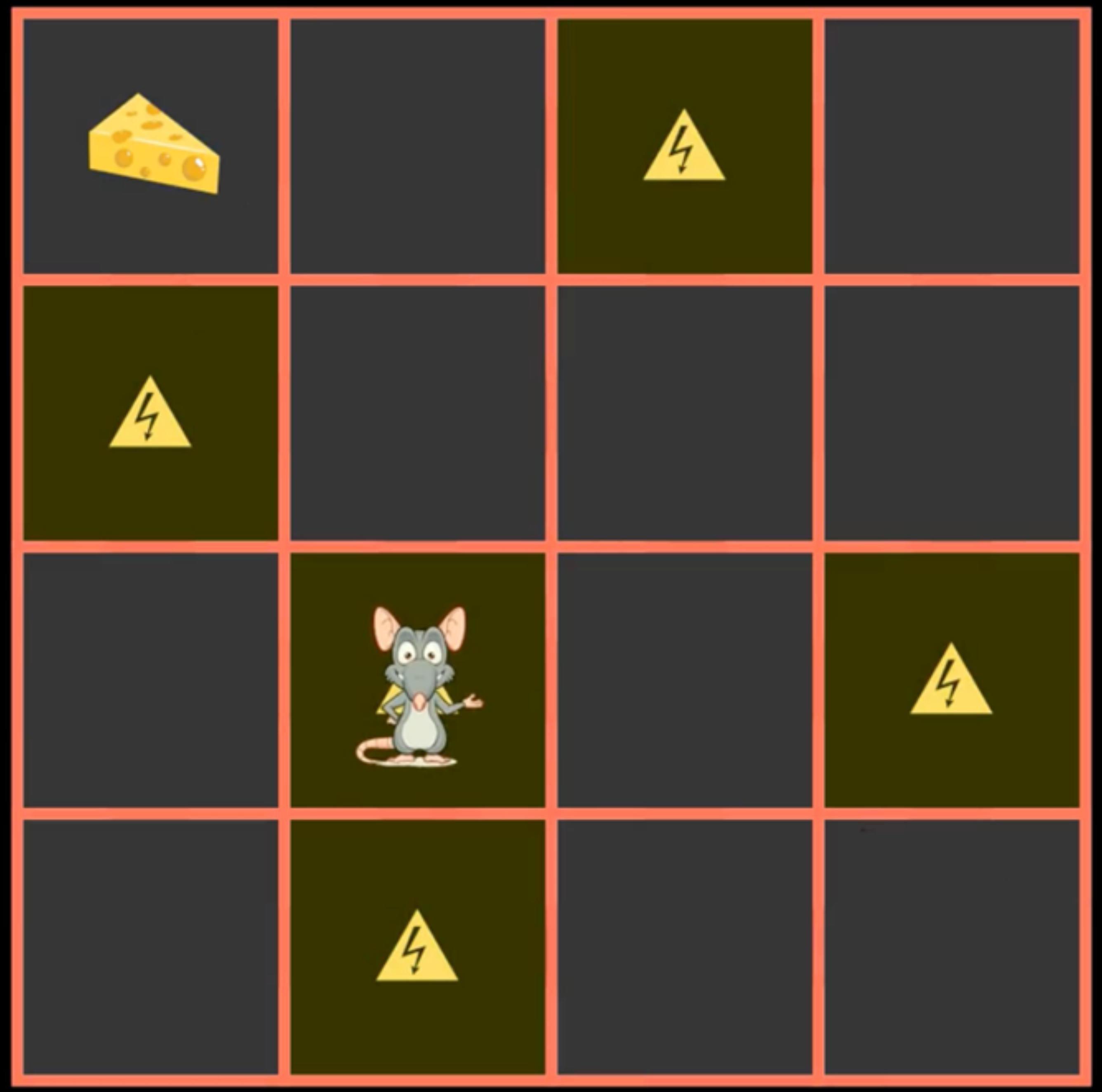
Attempt 3



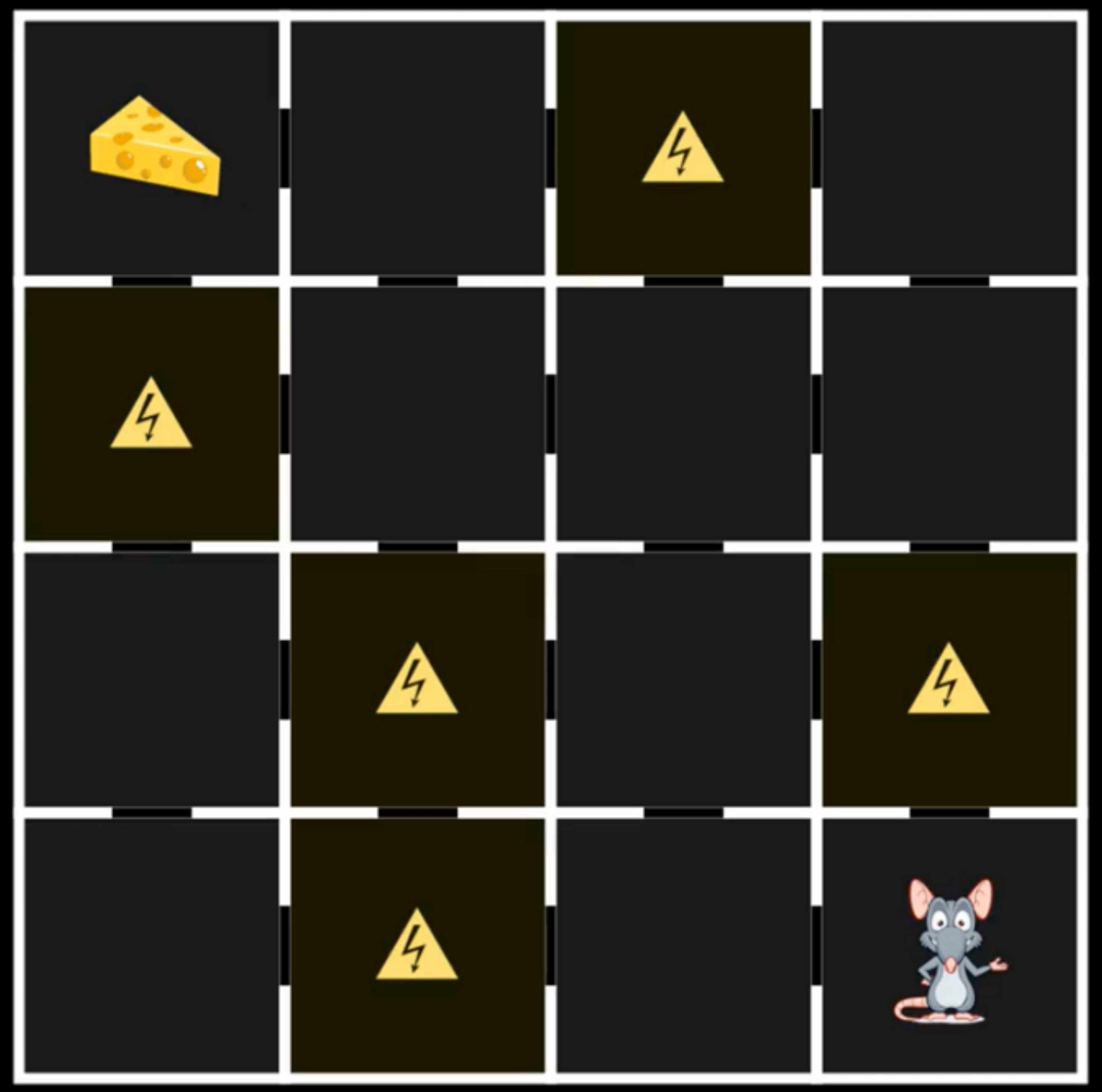
Attempt 3



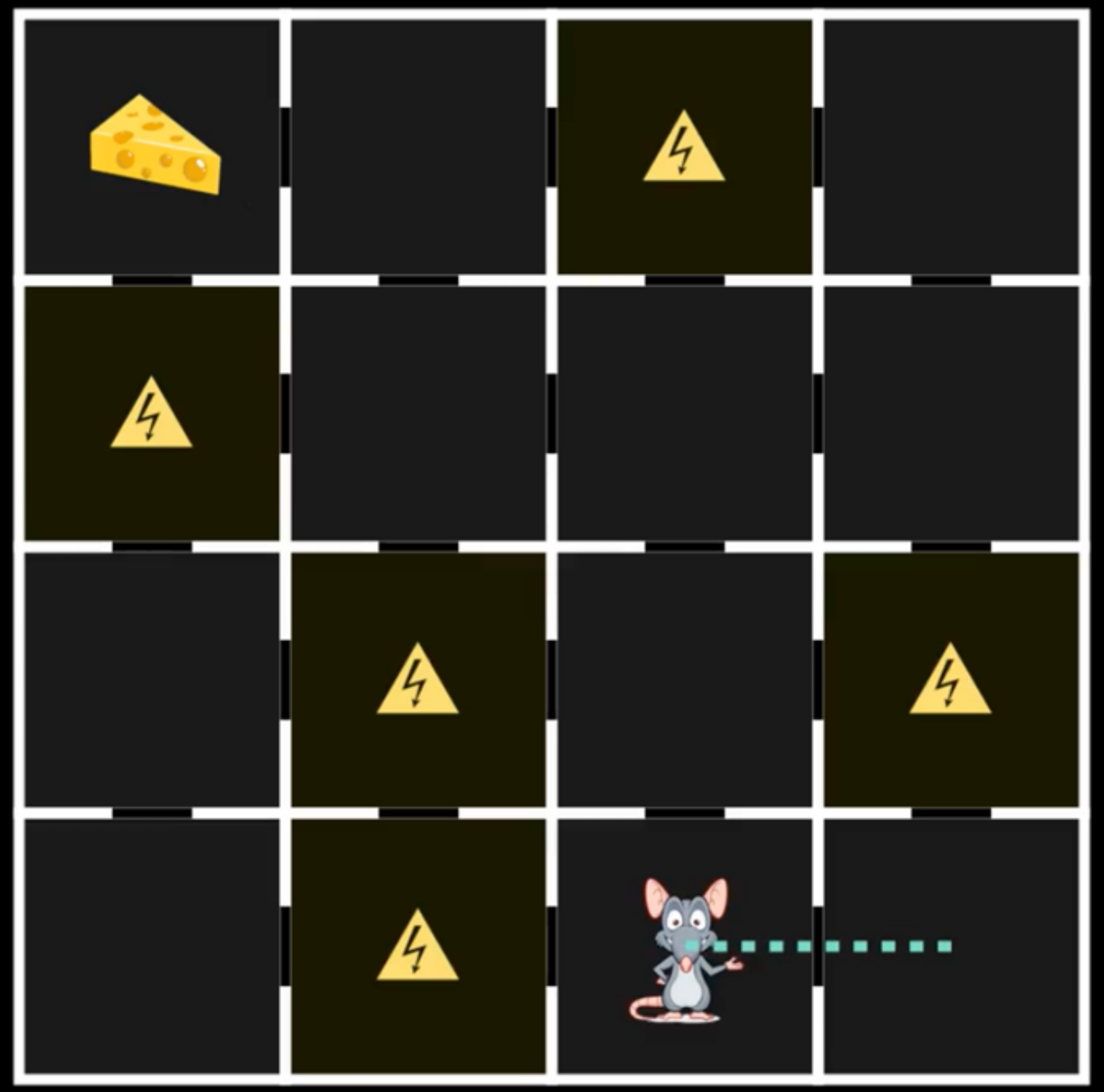
Attempt 3



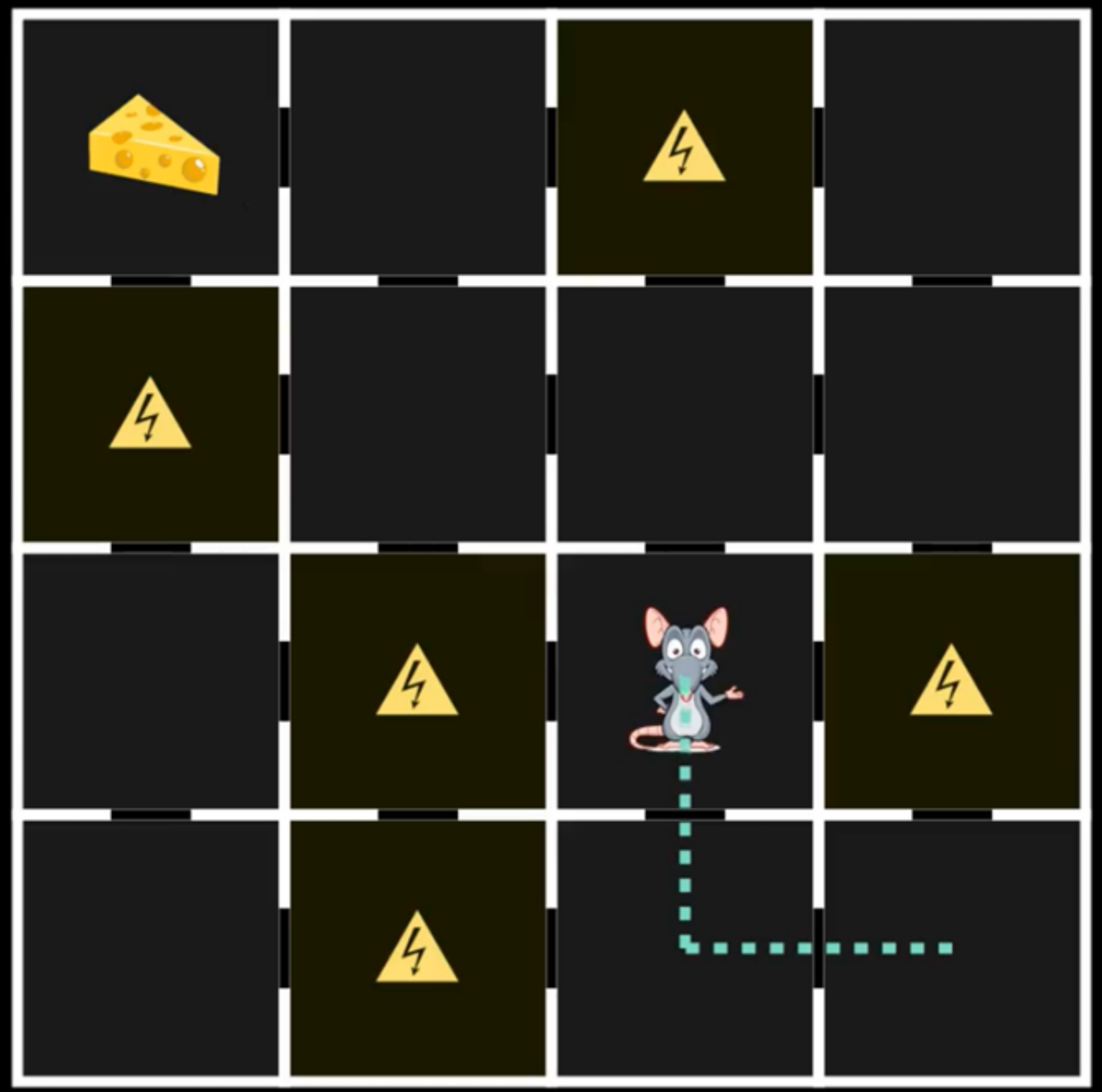
Attempt 4



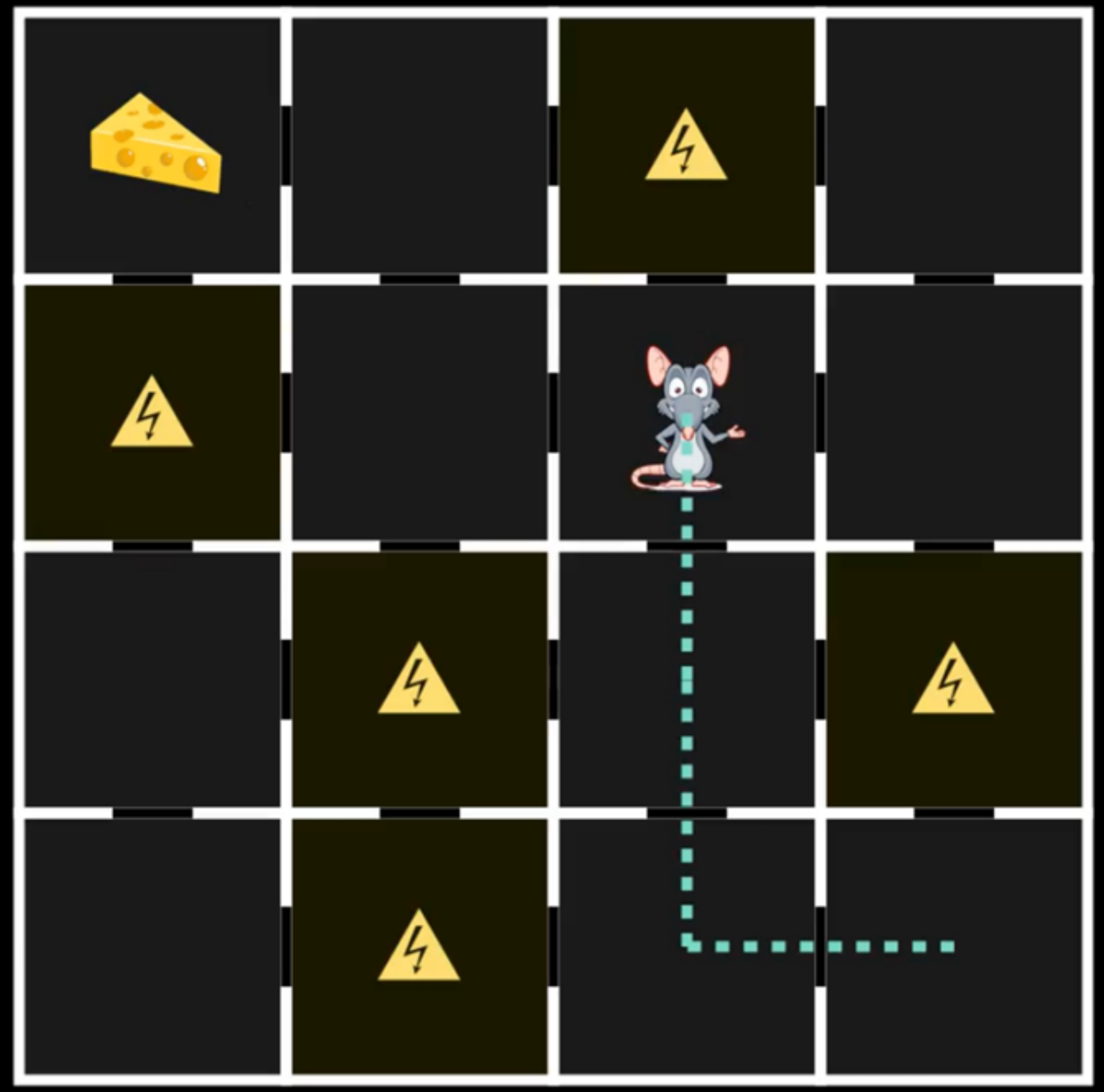
Attempt 4



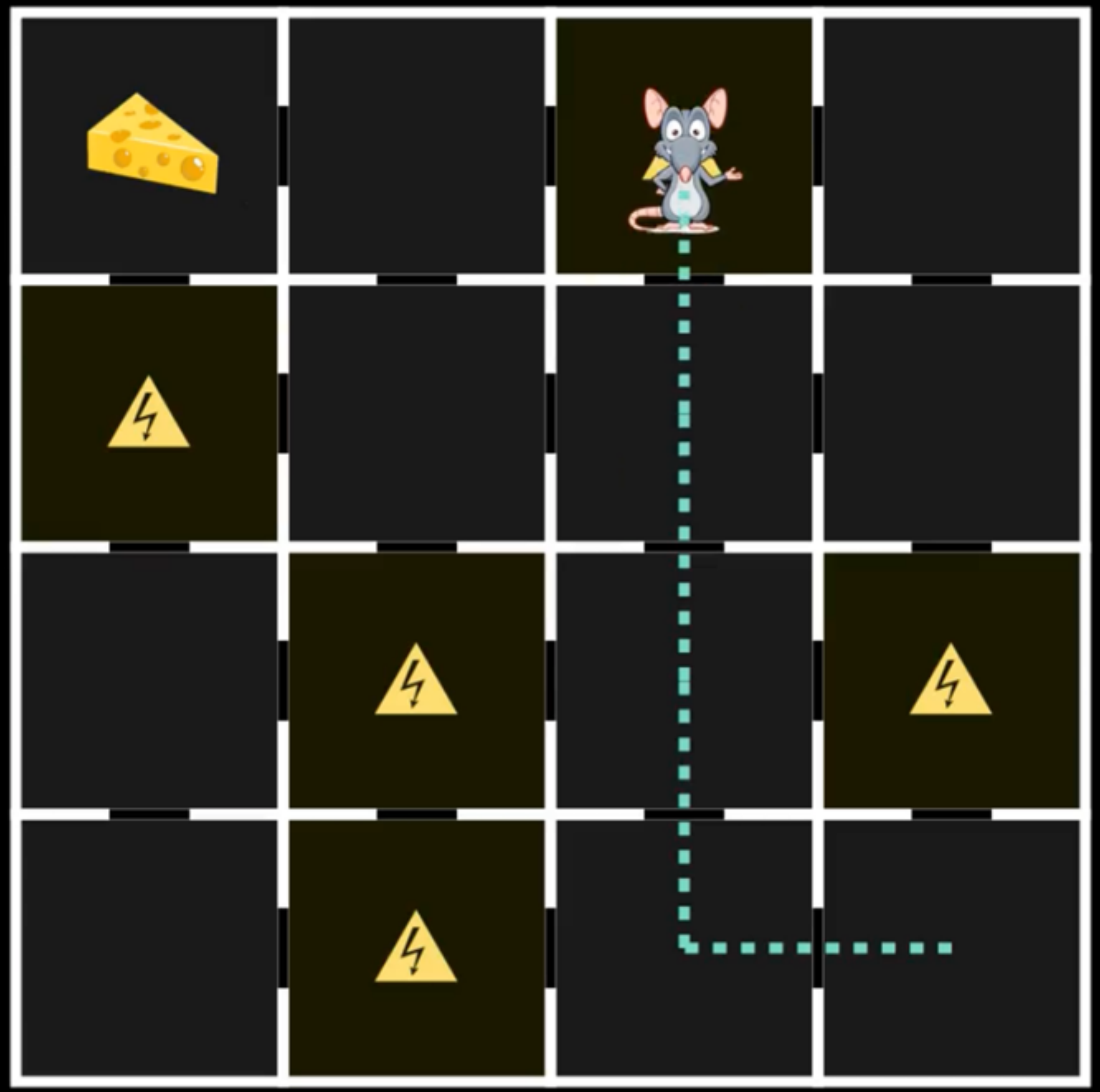
Attempt 4



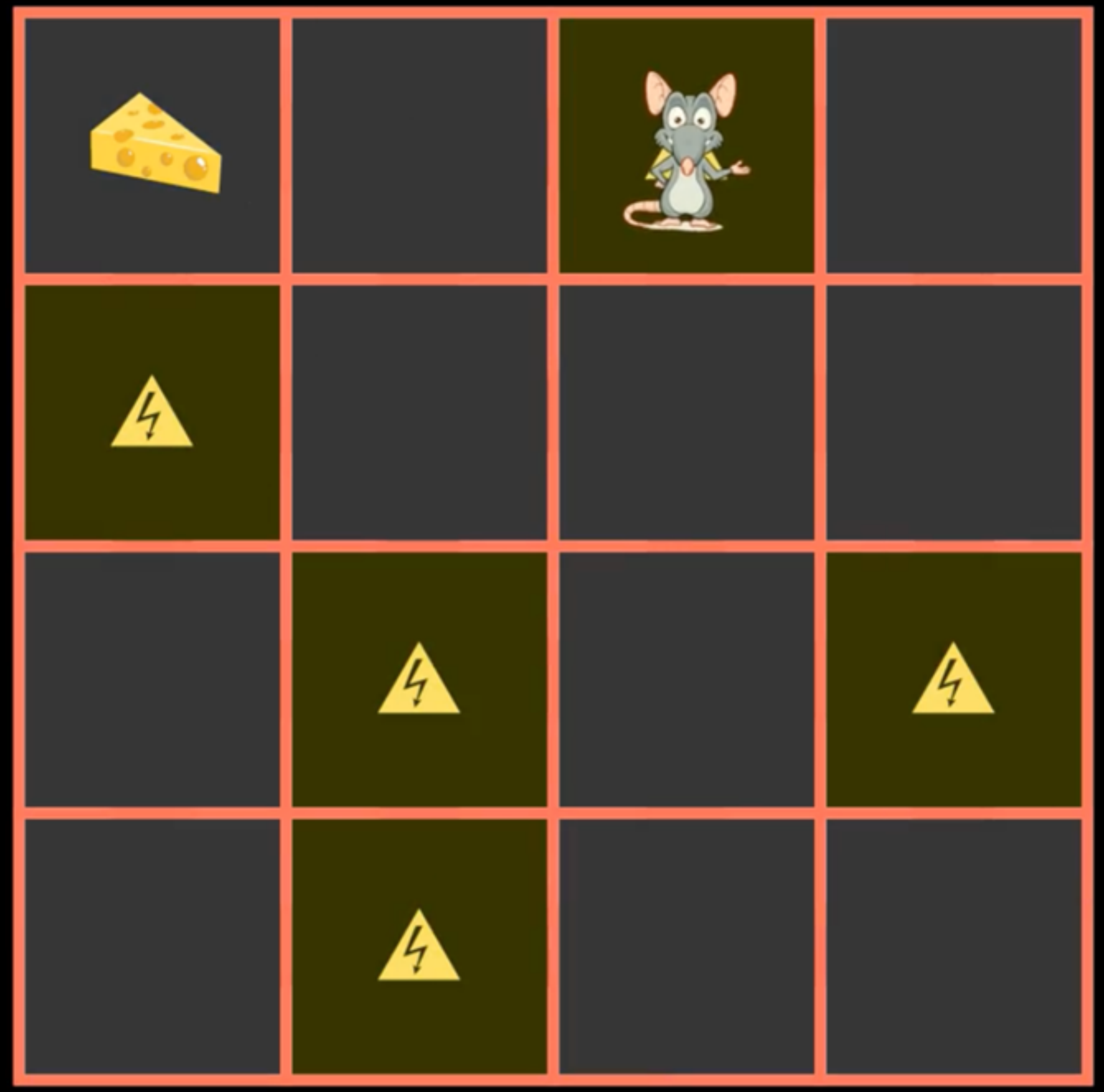
Attempt 4



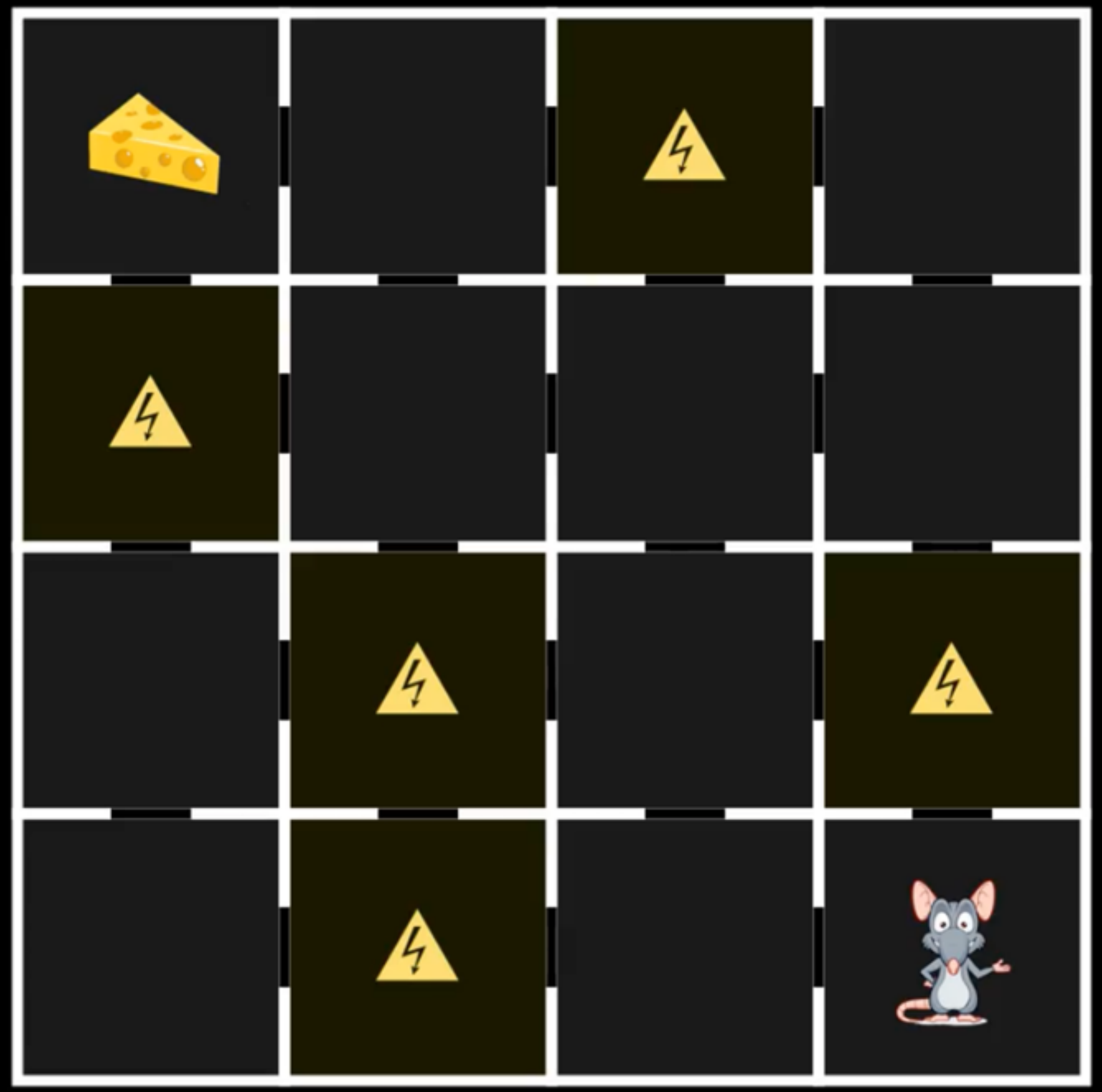
Attempt 4



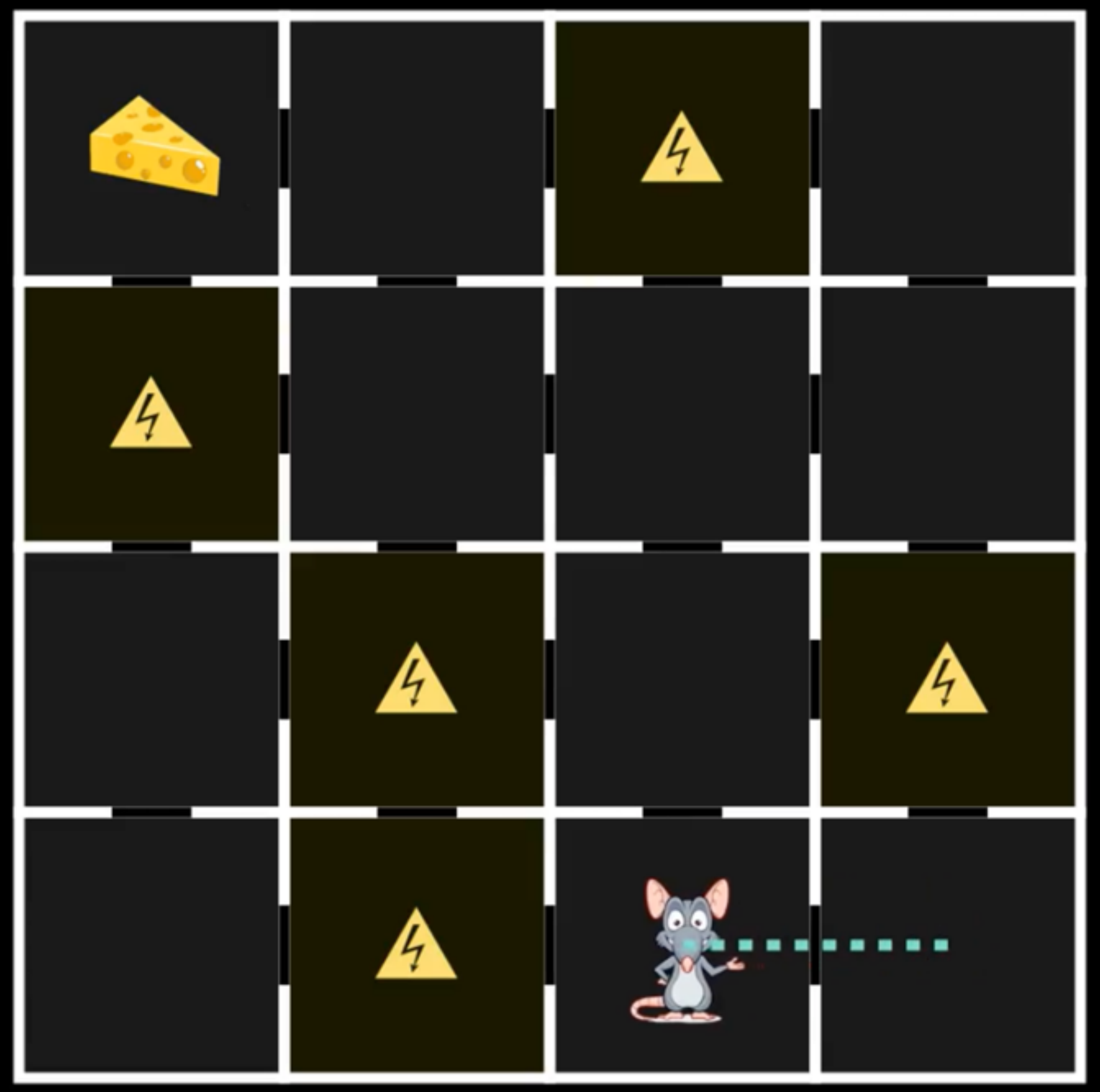
Attempt 4



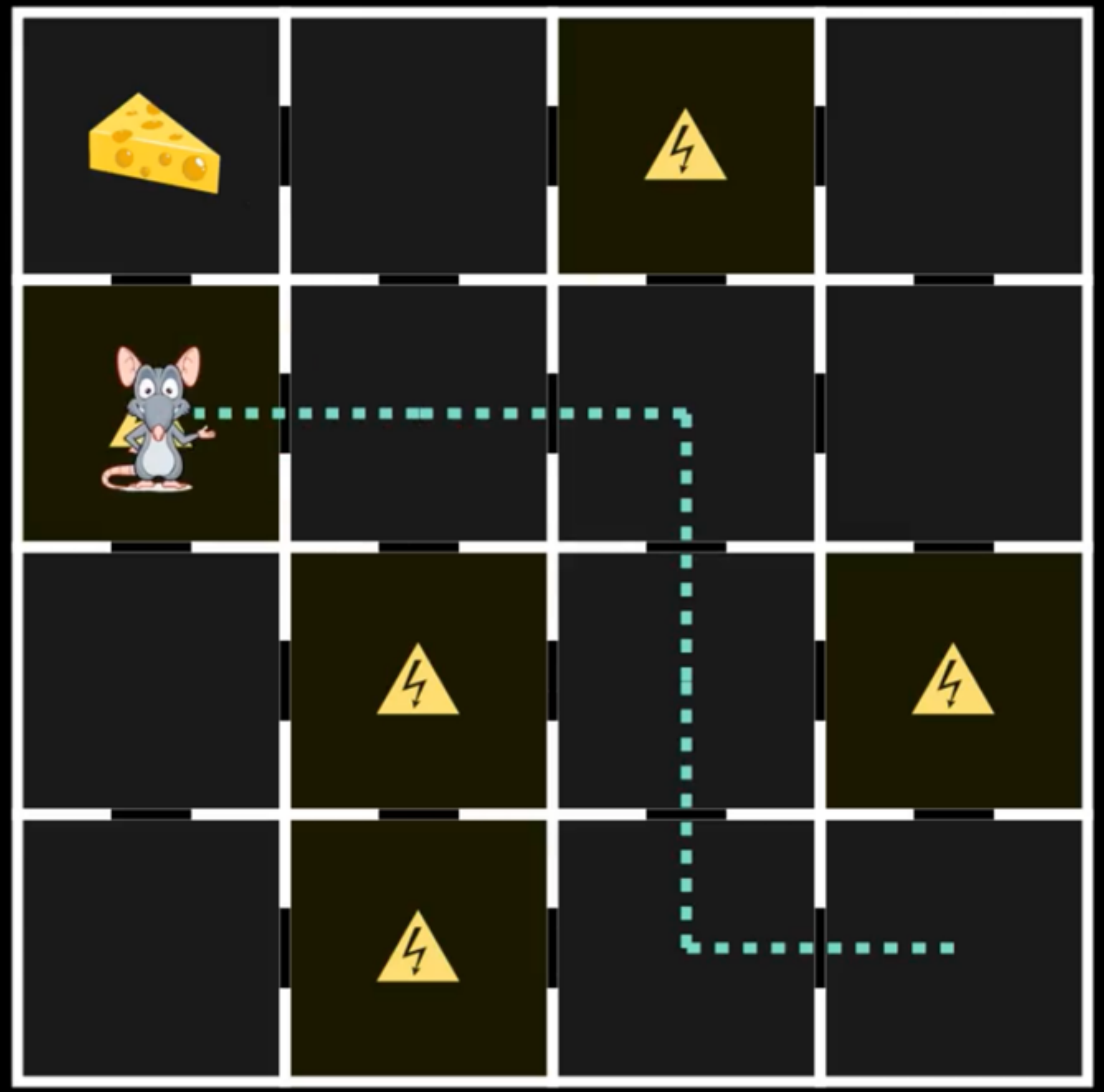
Attempt 5



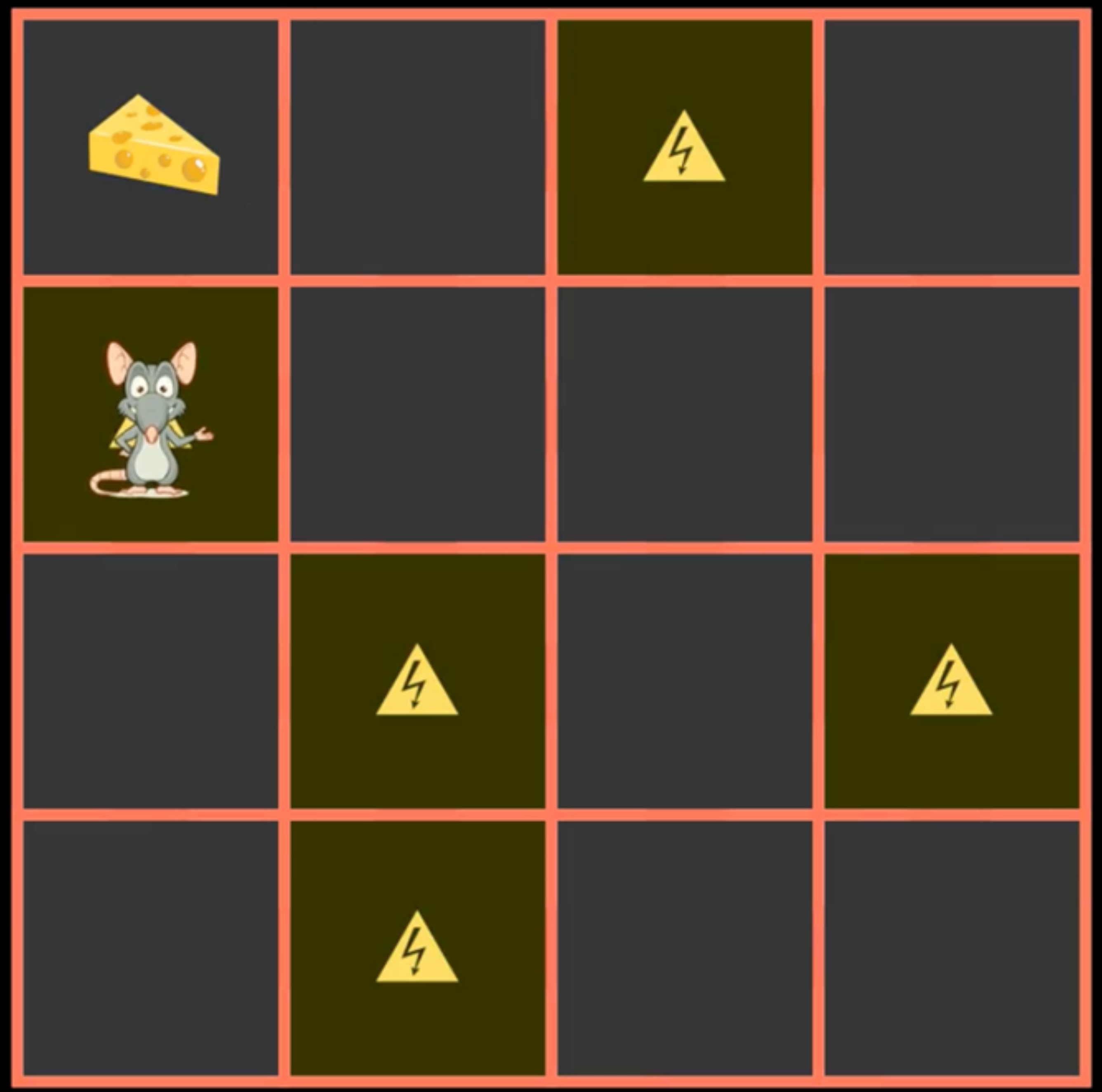
Attempt 5



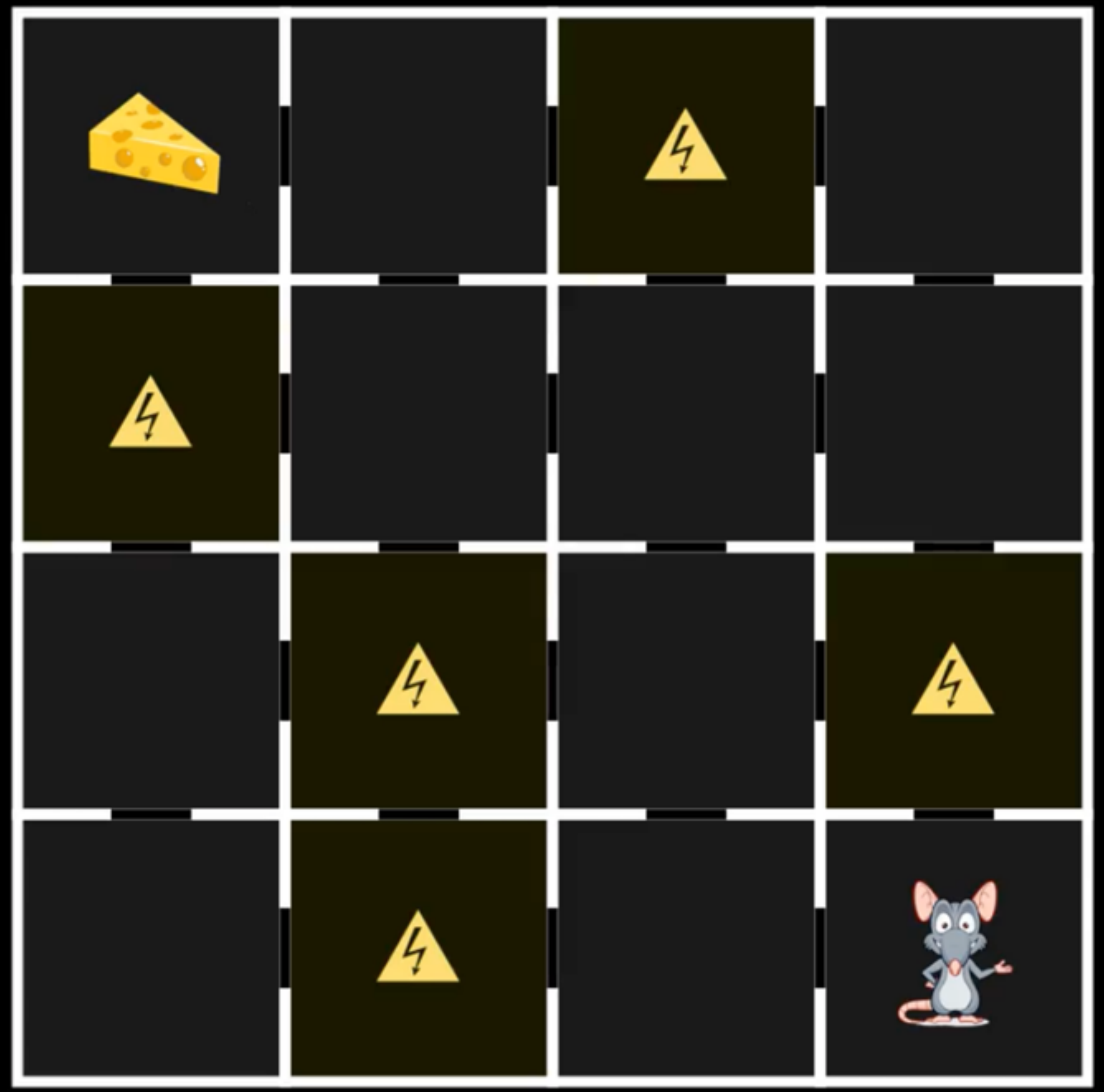
Attempt 5



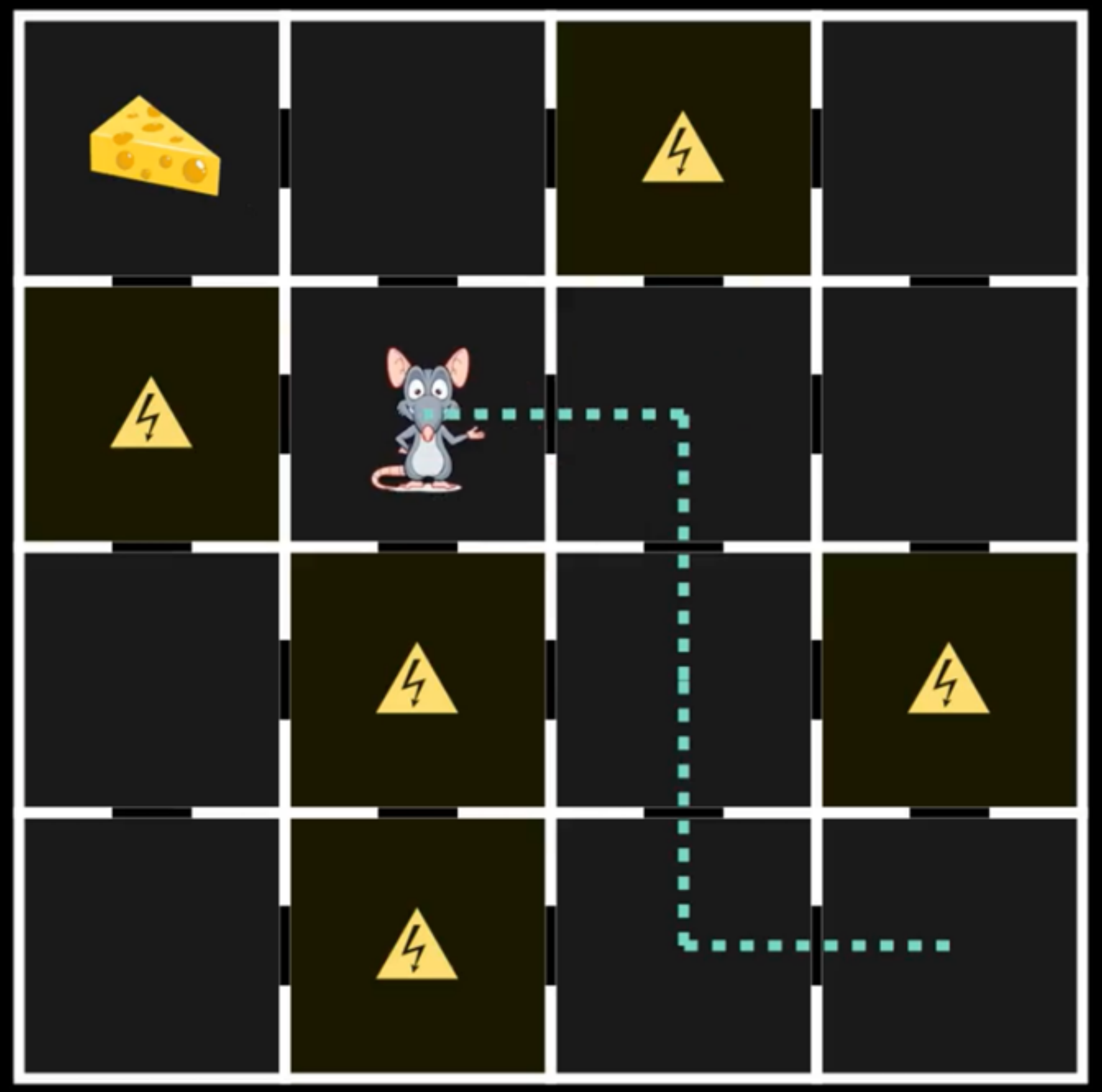
Attempt 5



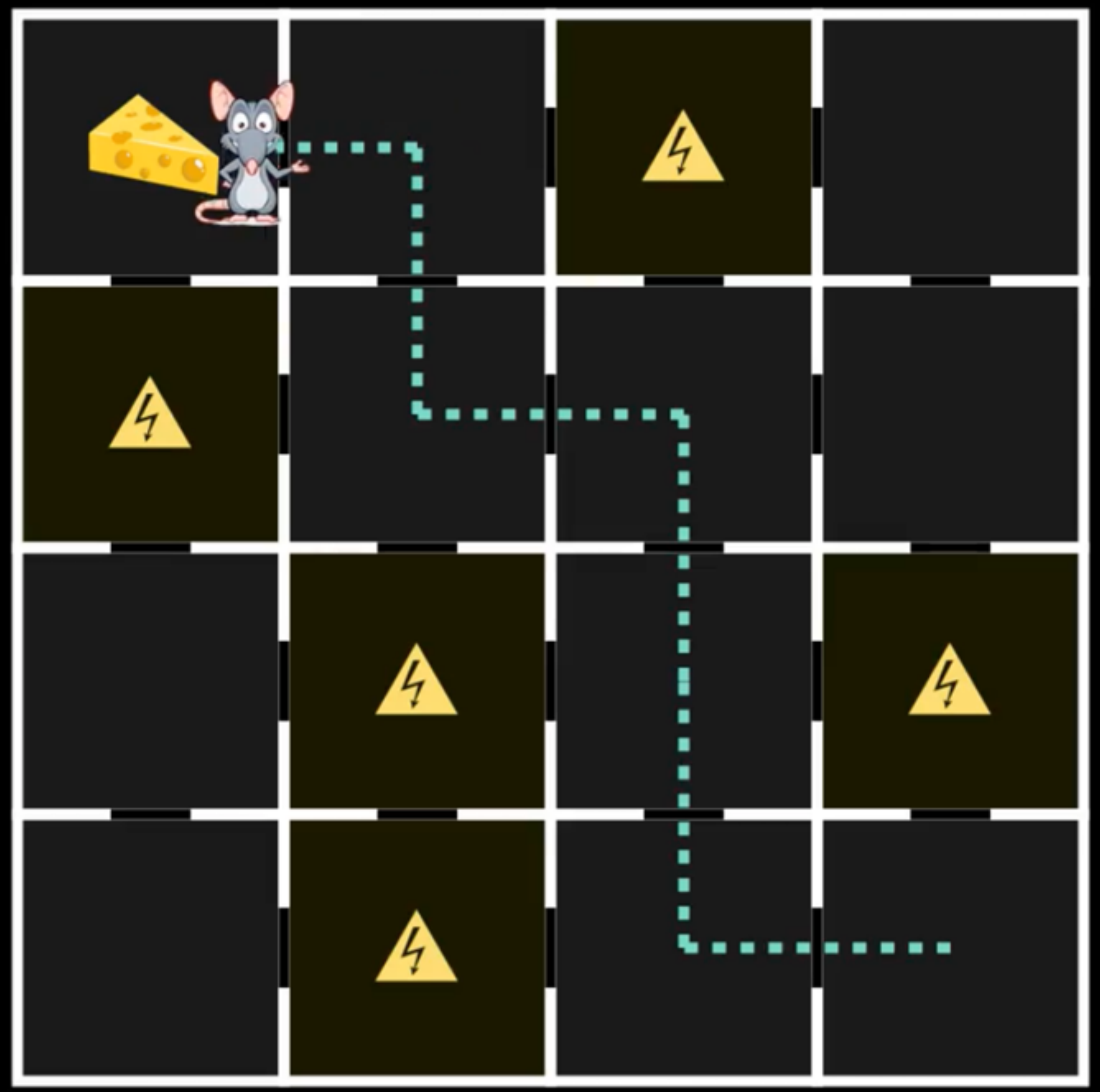
Attempt 6



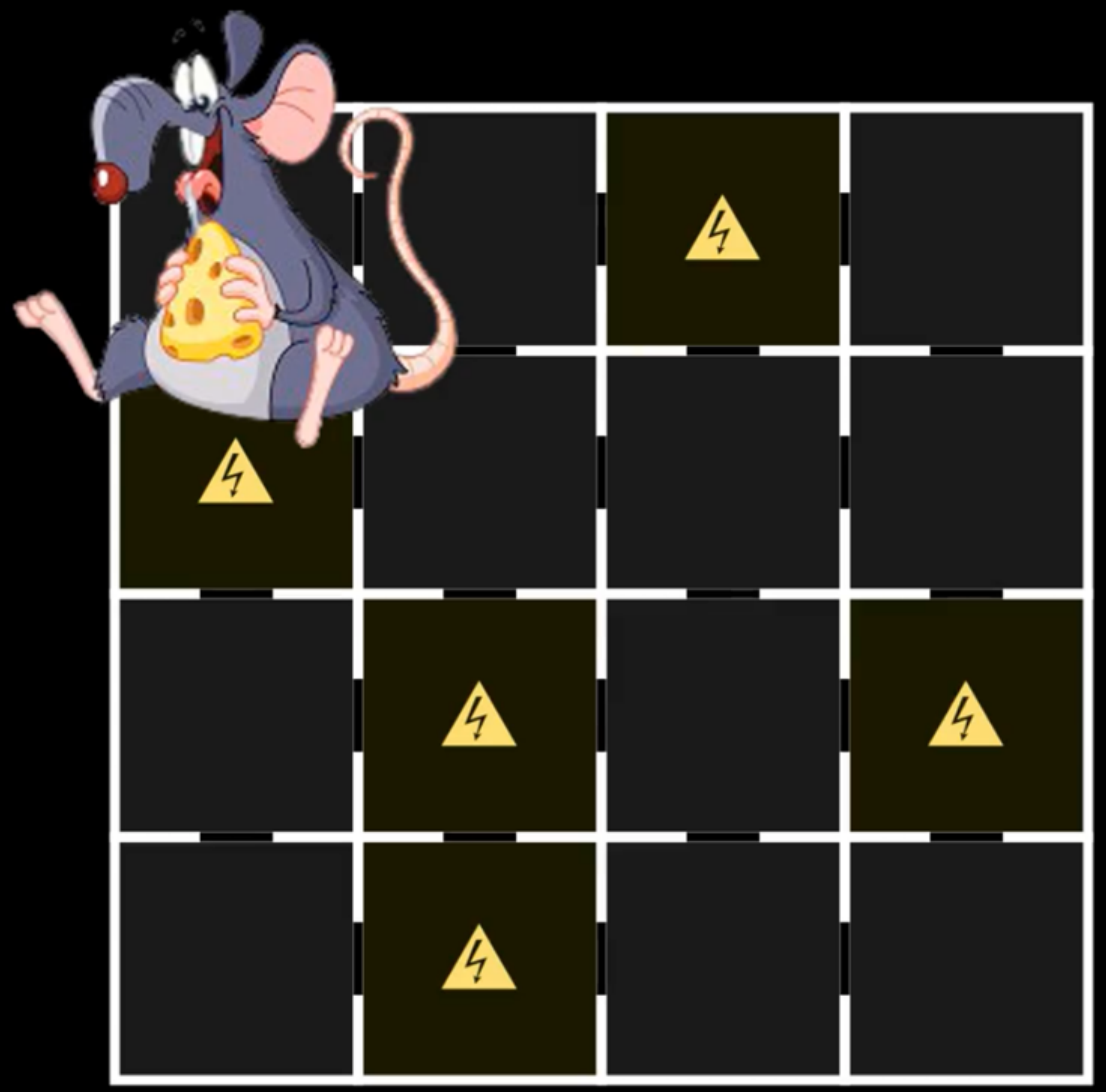
Attempt 6



Attempt 6



Attempt 6



A formal definition

Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward.

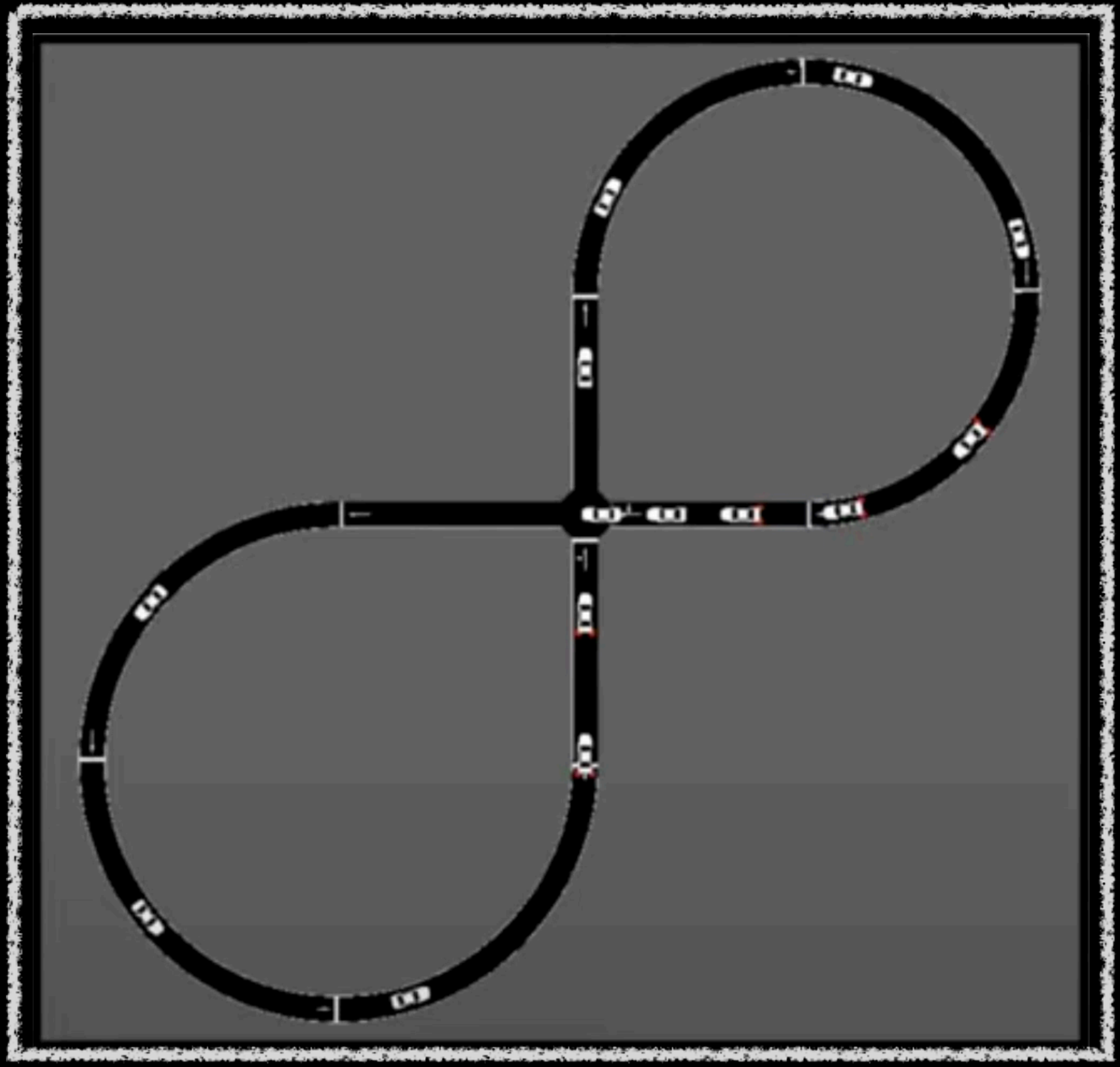
Reinforcement Learning (RL) is a type of machine learning technique that enables an agent to learn in an interactive environment by trial and error using feedback from its own actions and experiences.

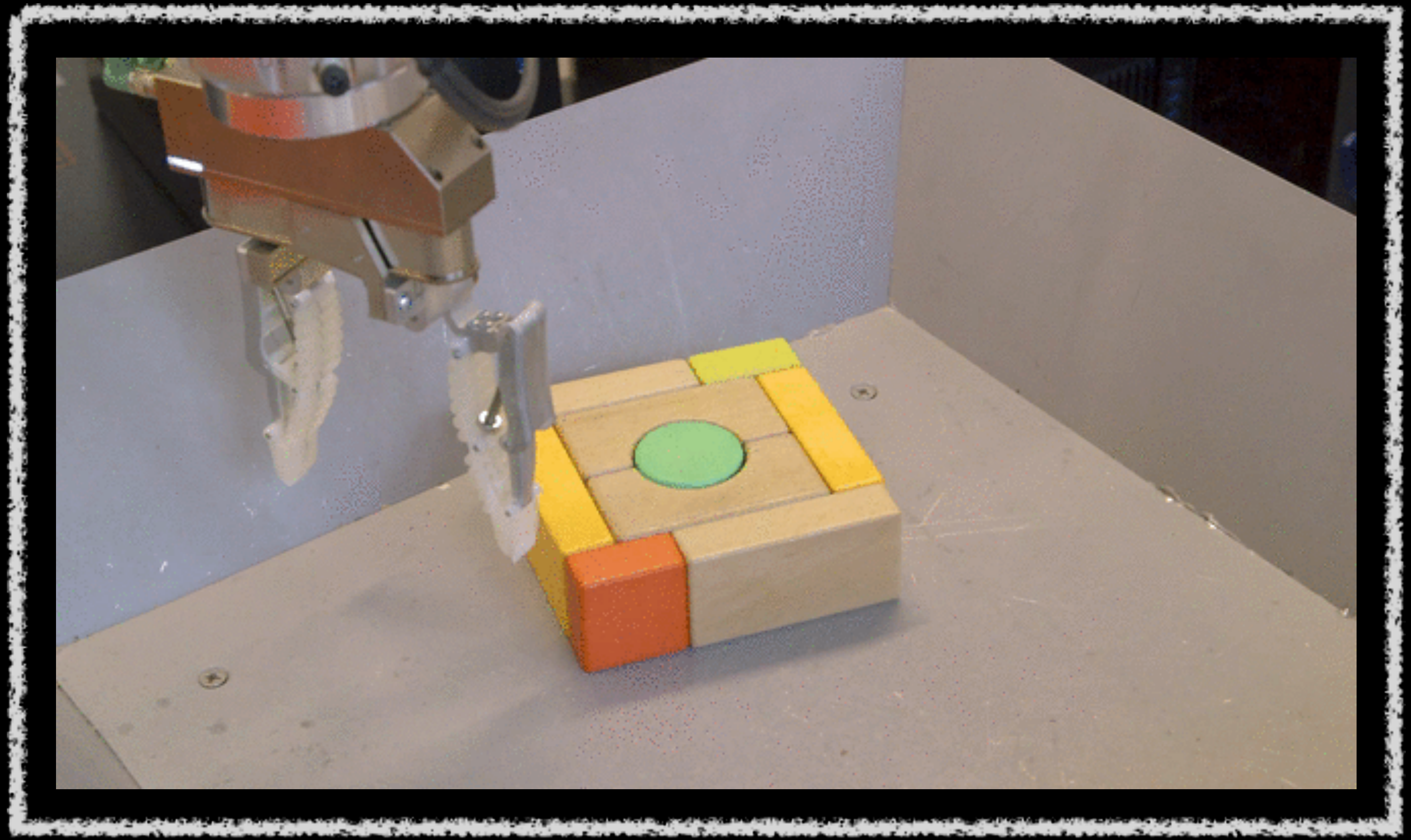
A mathematical formalisation of a decision making process.

Where is it used?

Games

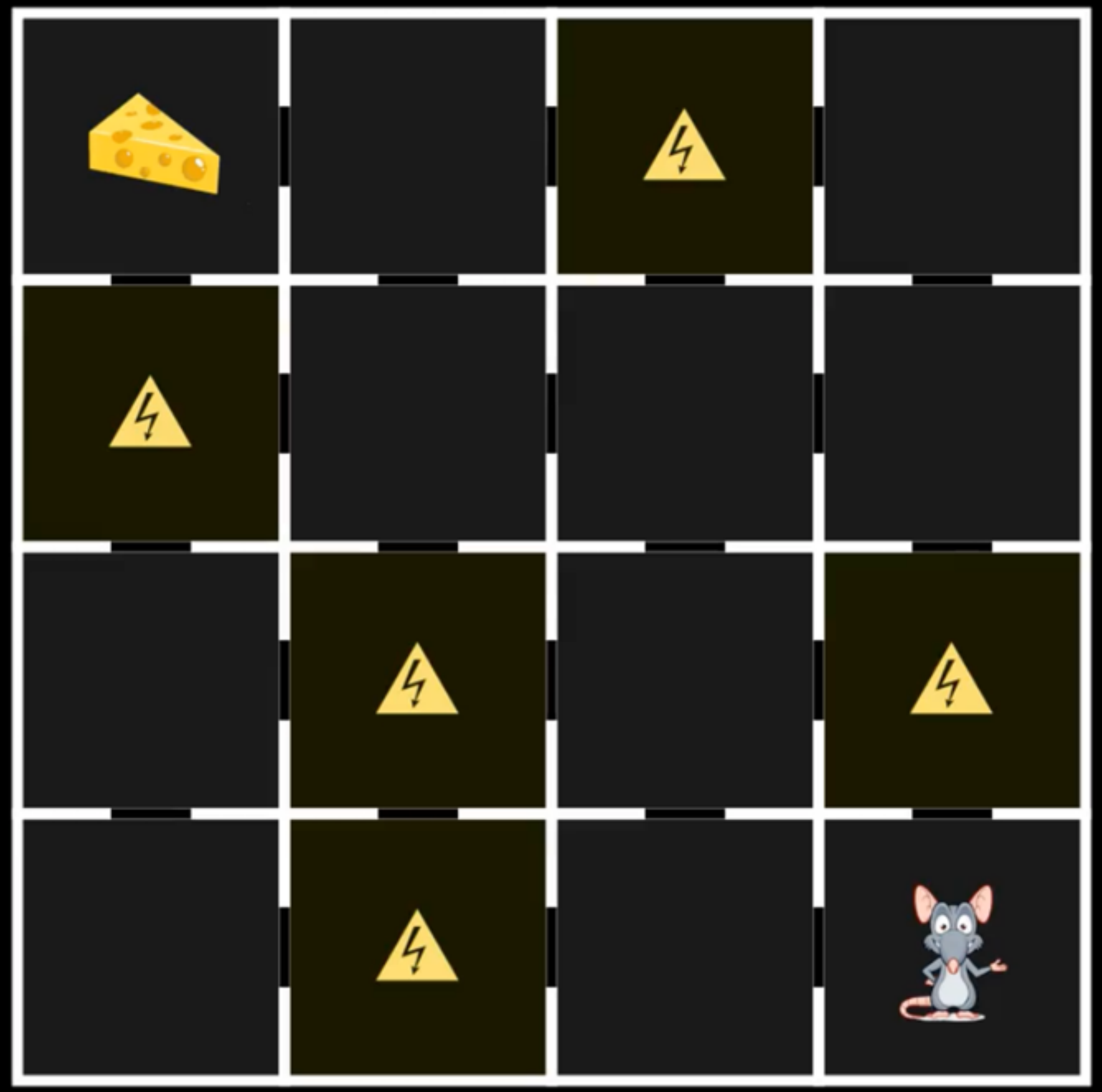
Traffic

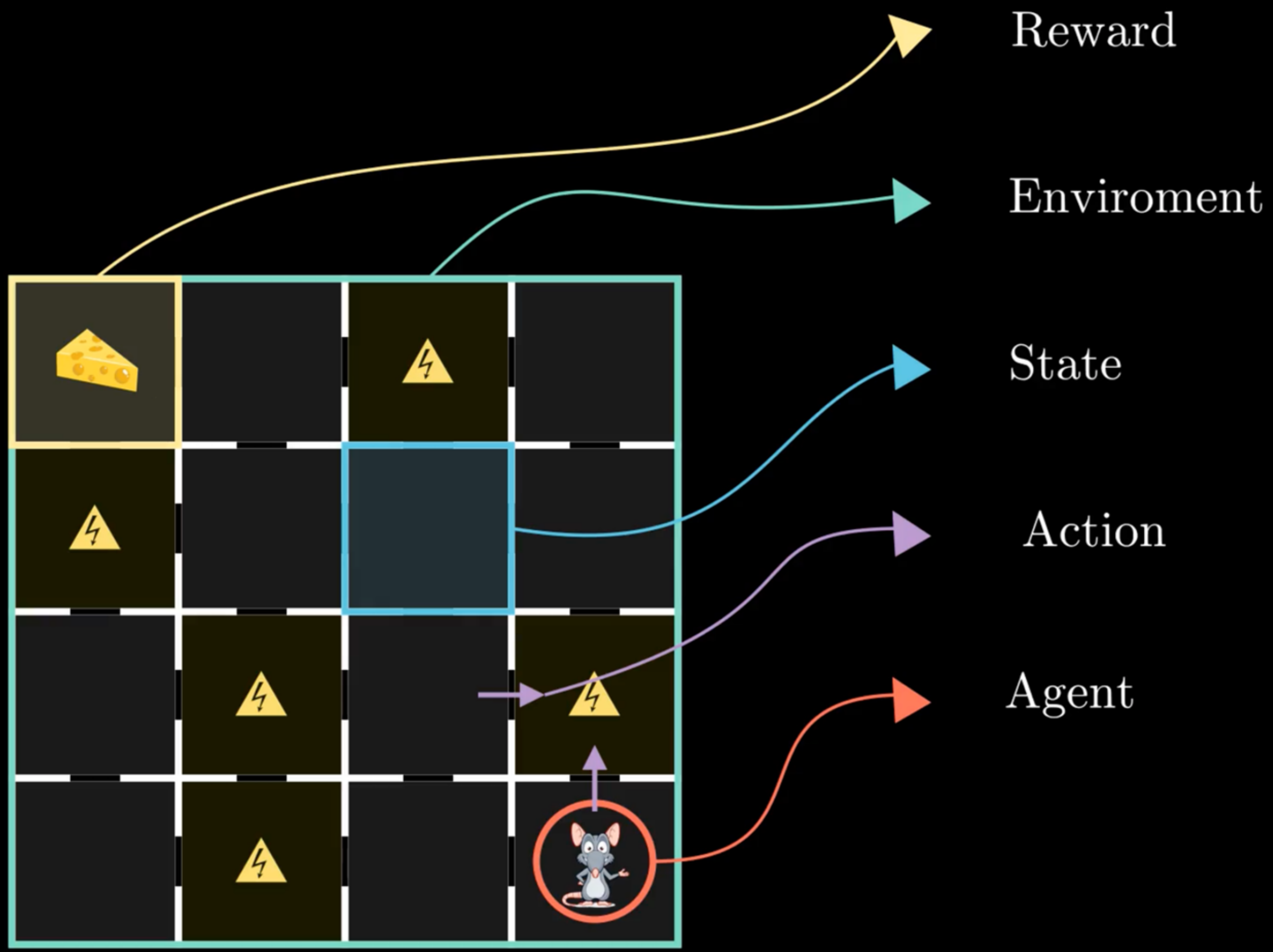




Robotics

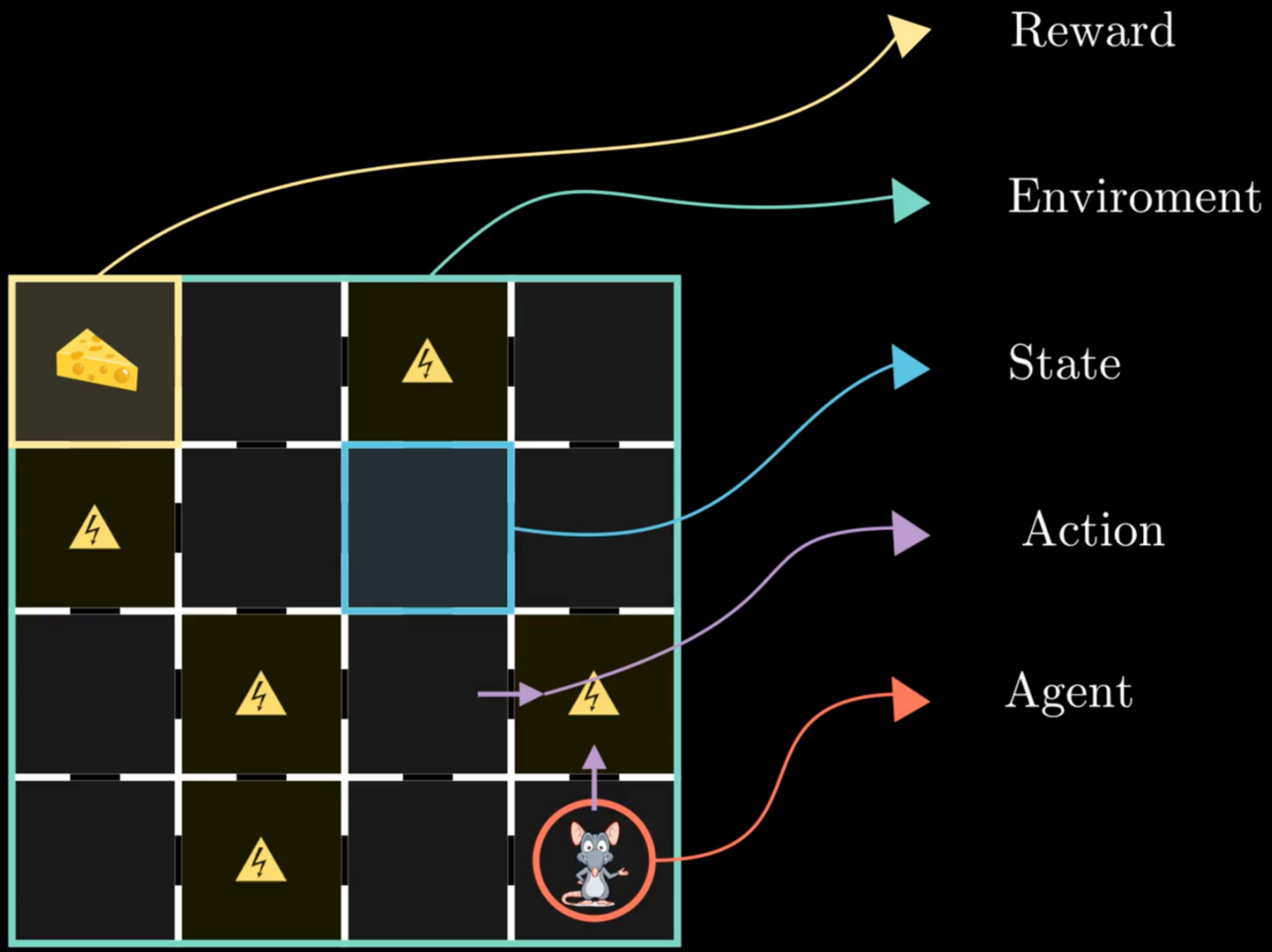
Elements of Reinforcement Learning




Reward
Enviroment
State
Action
Agent

Agent

An entity that takes a set of actions
to fulfill a set goal.

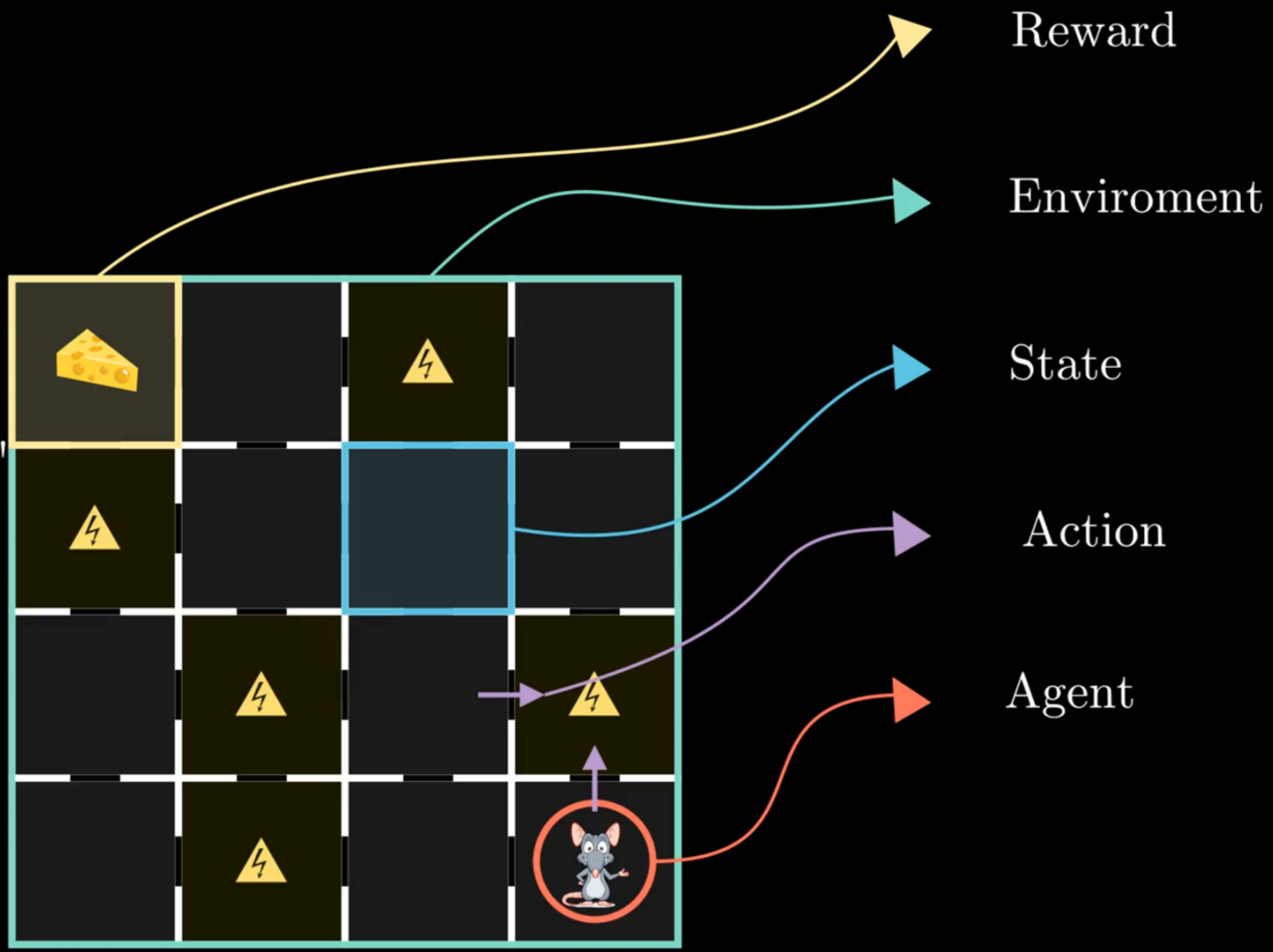


Agent

An entity that takes a set of actions to fulfill a set goal.

Environment

Everything the agent interacts with. "The agent may or may not know the dynamics of the entire environment."



Agent

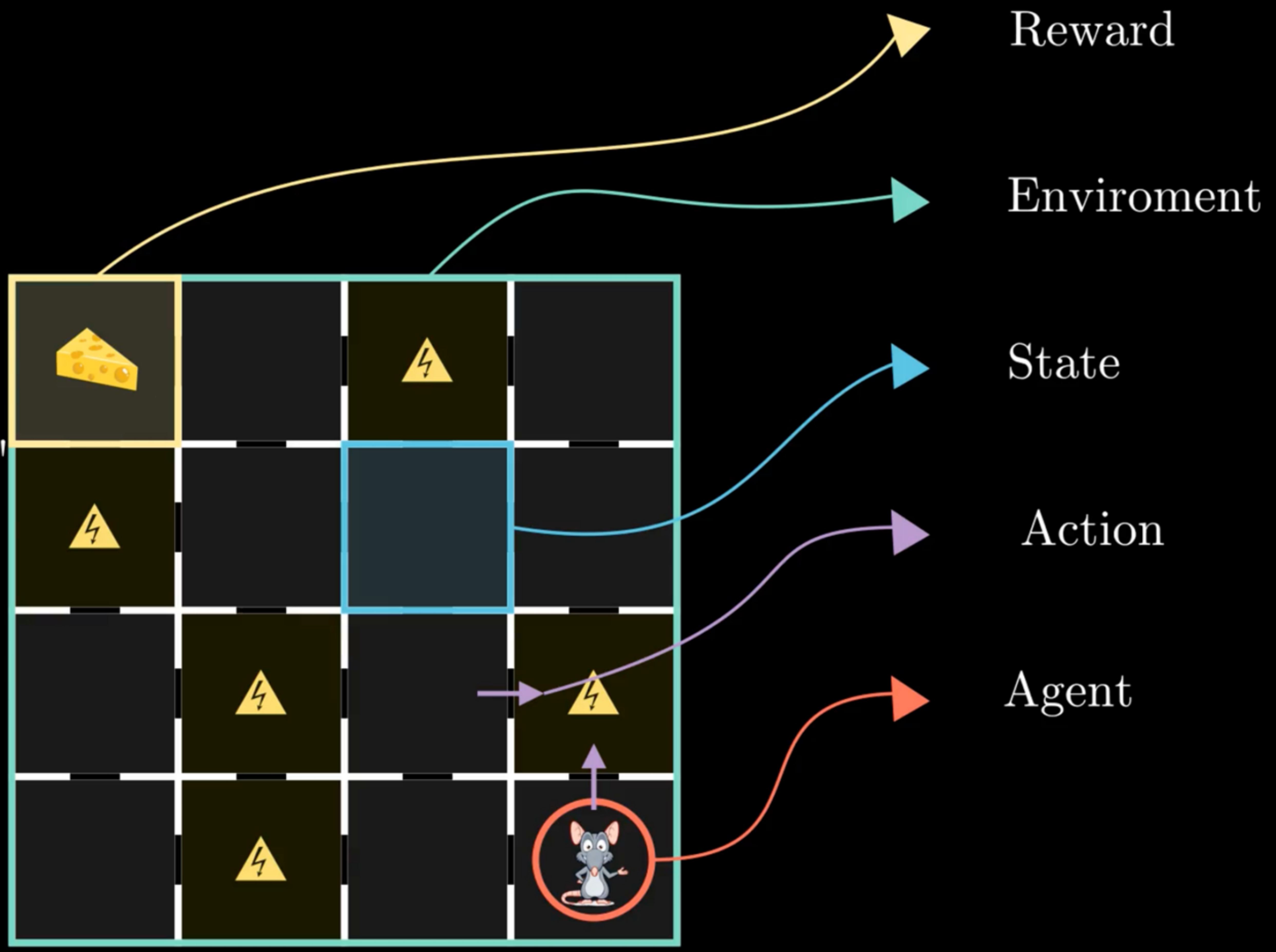
An entity that takes a set of actions
to fulfill a set goal.

Environment

Everything the agent interacts with.
"The agent may or may not know
the dynamics of the entire environment."

State

It is a signal that tells the agent how
the enviroment is at a given time.



Agent

An entity that takes a set of actions to fulfill a set goal.

Environment

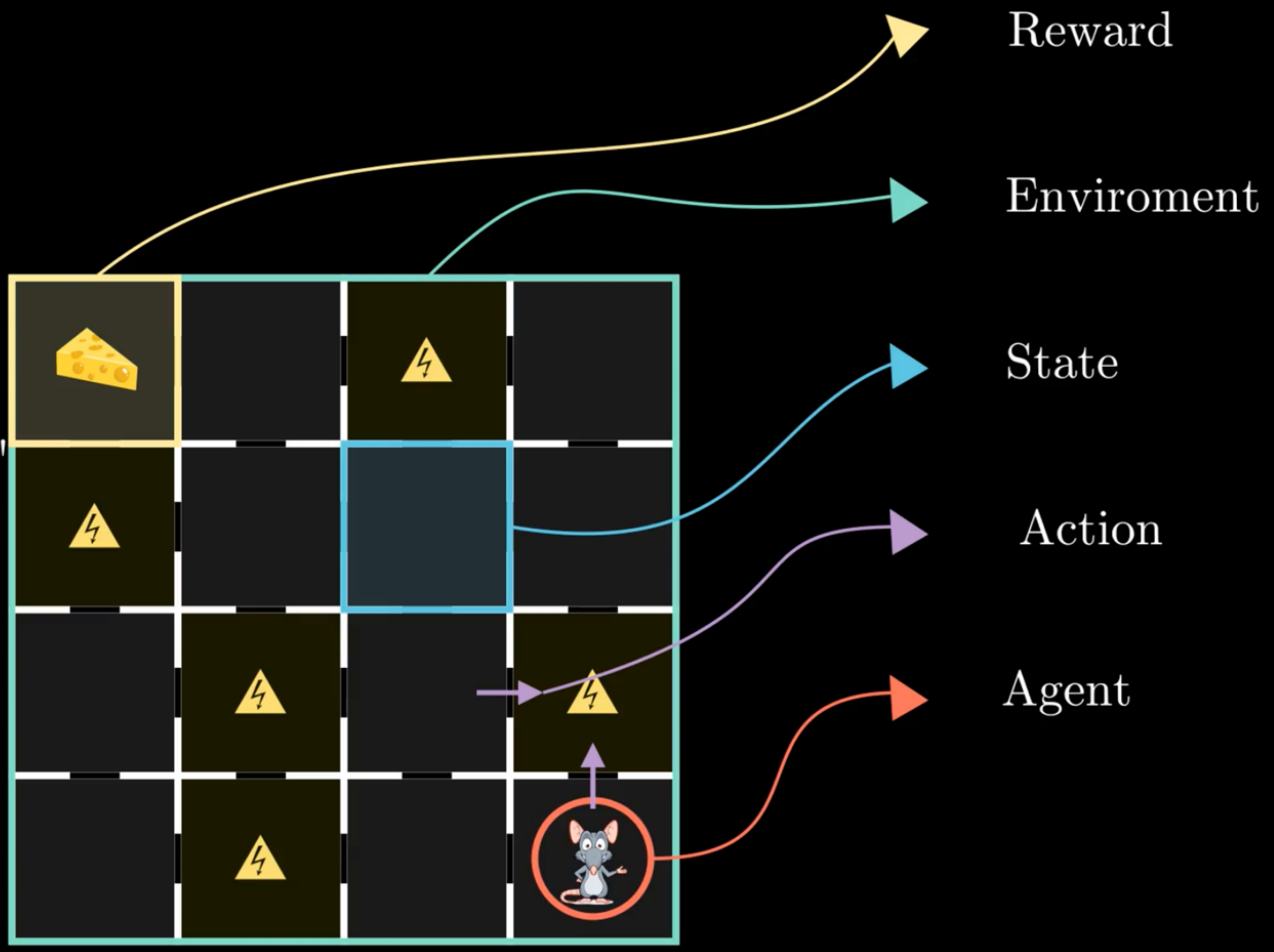
Everything the agent interacts with. "The agent may or may not know the dynamics of the entire environment."

State

It is a signal that tells the agent how the enviroment is at a given time.

Action

The set of possible actions an agent can take in an enviroment.



Agent

An entity that takes a set of actions to fulfill a set goal.

Environment

Everything the agent interacts with. "The agent may or may not know the dynamics of the entire environment."

State

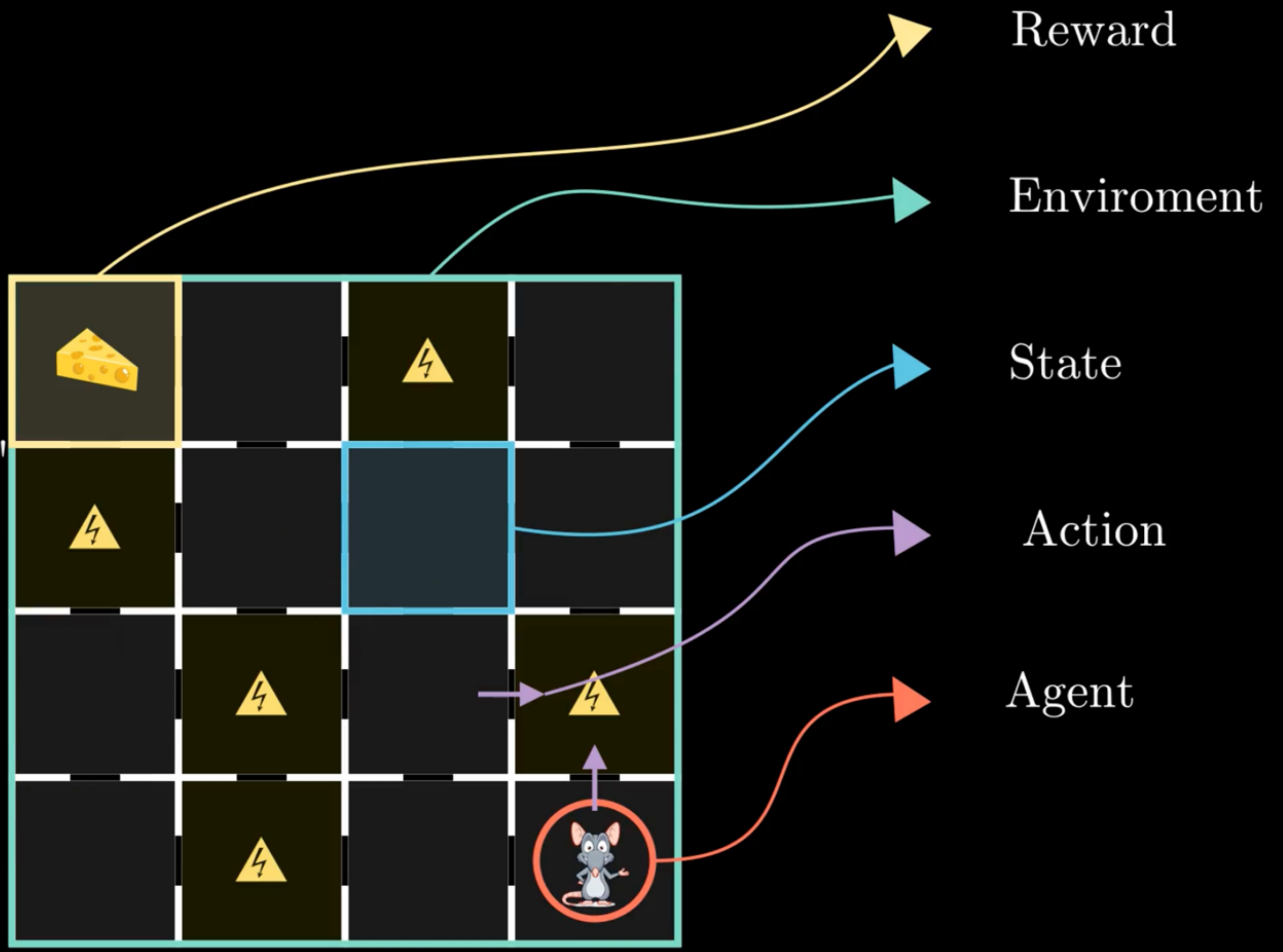
It is a signal that tells the agent how the enviroment is at a given time.

Action

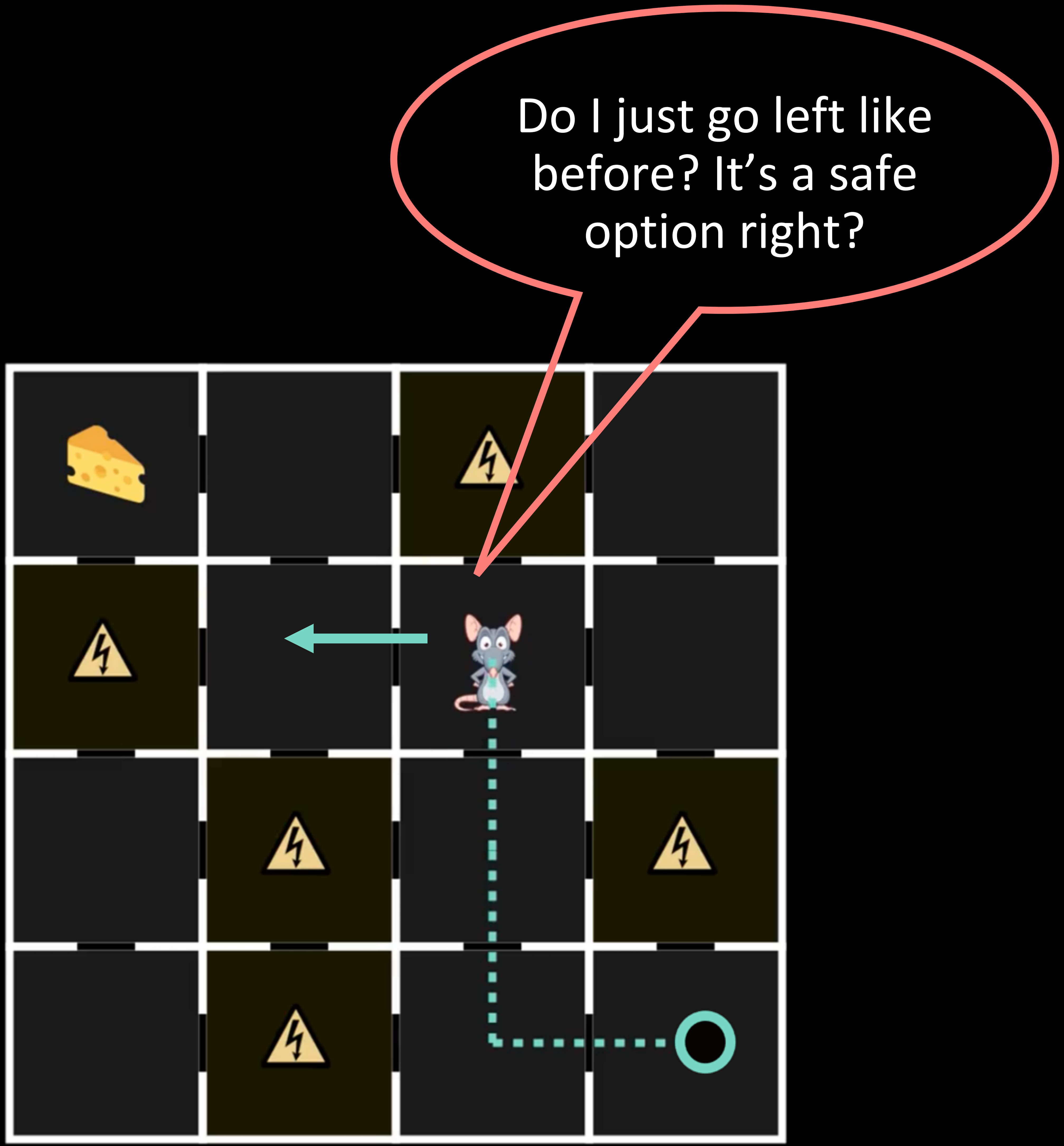
The set of possible actions an agent can take in an enviroment.

Reward

A numerical value that forms the basis for an action to be taken. It is returned by the environment.

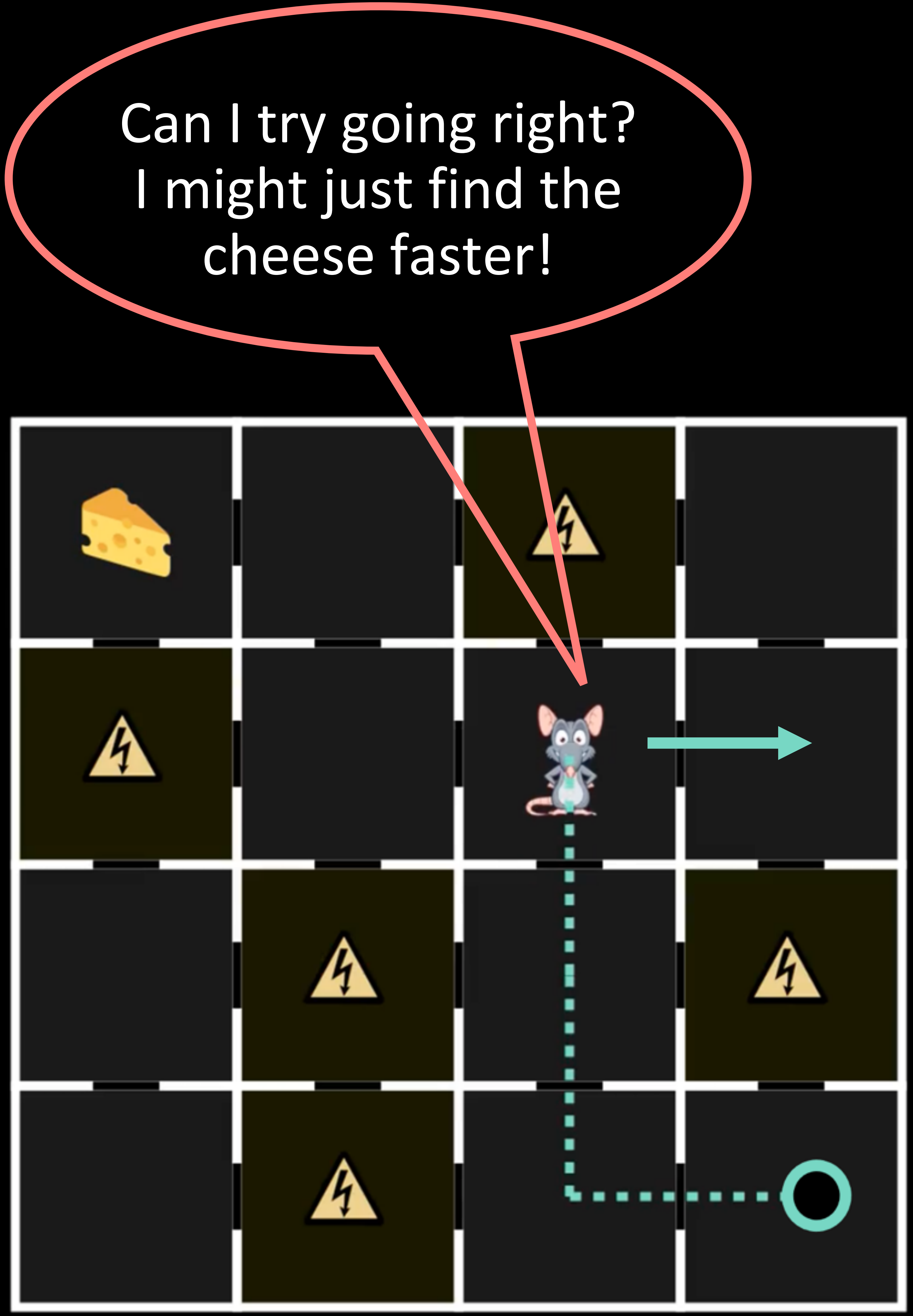


Exploration vs Exploitation



Exploitation

Better immediate reward



Exploration

Long term return is increased

Markov Decision Process

Markov Property

The future is independent of the past given the present.

A state S_t is said to be Markov if and only if

$$\mathbb{P}[S_{t+1}|S_1, S_2, S_3 \ldots, S_t] = \mathbb{P}[S_{t+1}|S_t]$$

Hence we require the state to encapsulate all the necessary information from the history

State Transition Probability

The probability of going from a Markov state *s* to a state *s'* is given by

$$P_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

State Transition Matrix

Given the previous probability we get a **State Transition Matrix** that gives the probabilities from all state *s* to all successor states *s'*.

Probability of going from s_1 to s_1

$$\begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & \vdots & \vdots \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

Each row characterises all possible transitions from one state s_1 to all permissible states s_n

Sum of each row is 1

Markov Process

Gives the entire dynamics of the environment

A Markov Process is a tuple $\langle S, P \rangle$

- S is the set of states (finite)
- P is the state transition matrix

Markov Process Chain

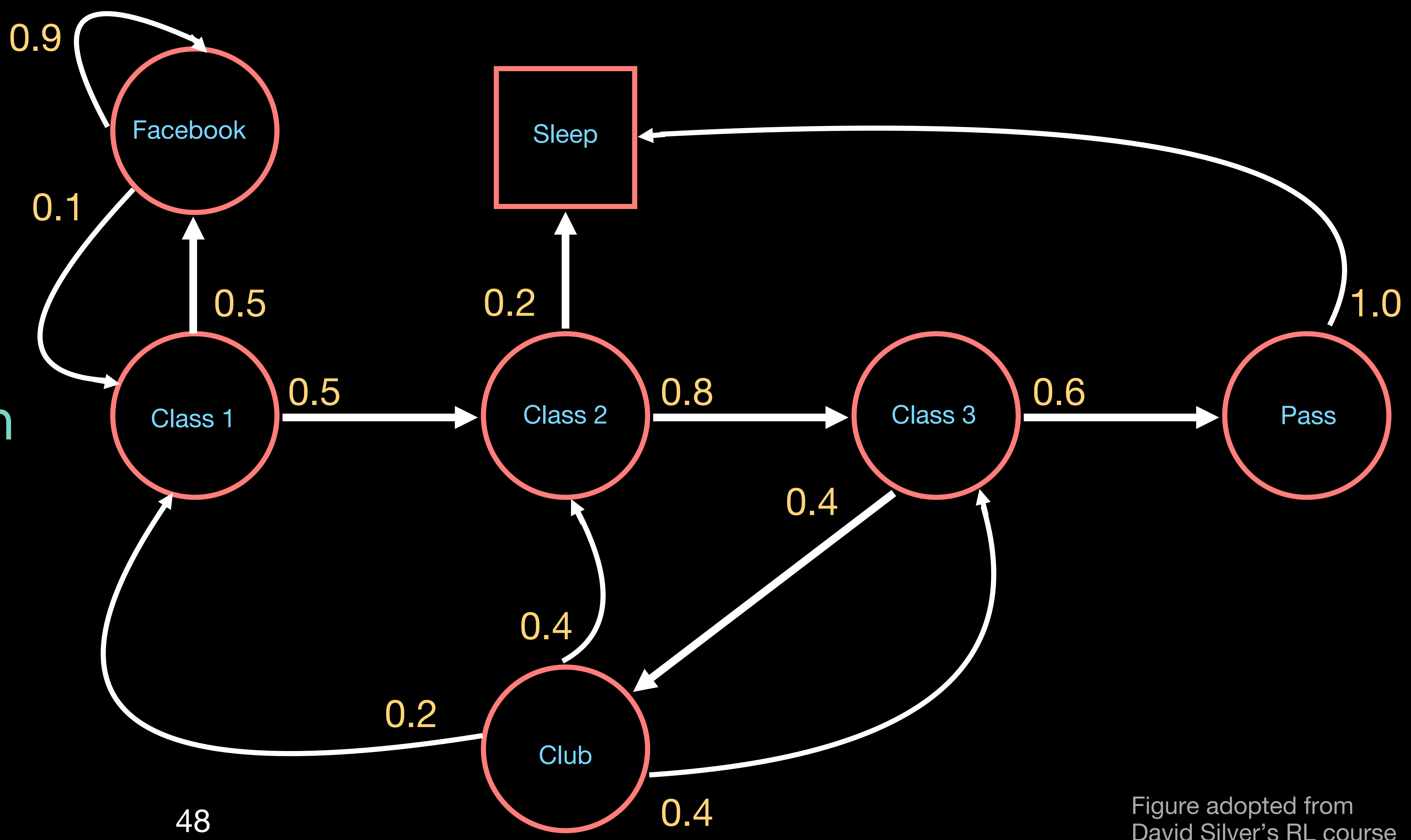
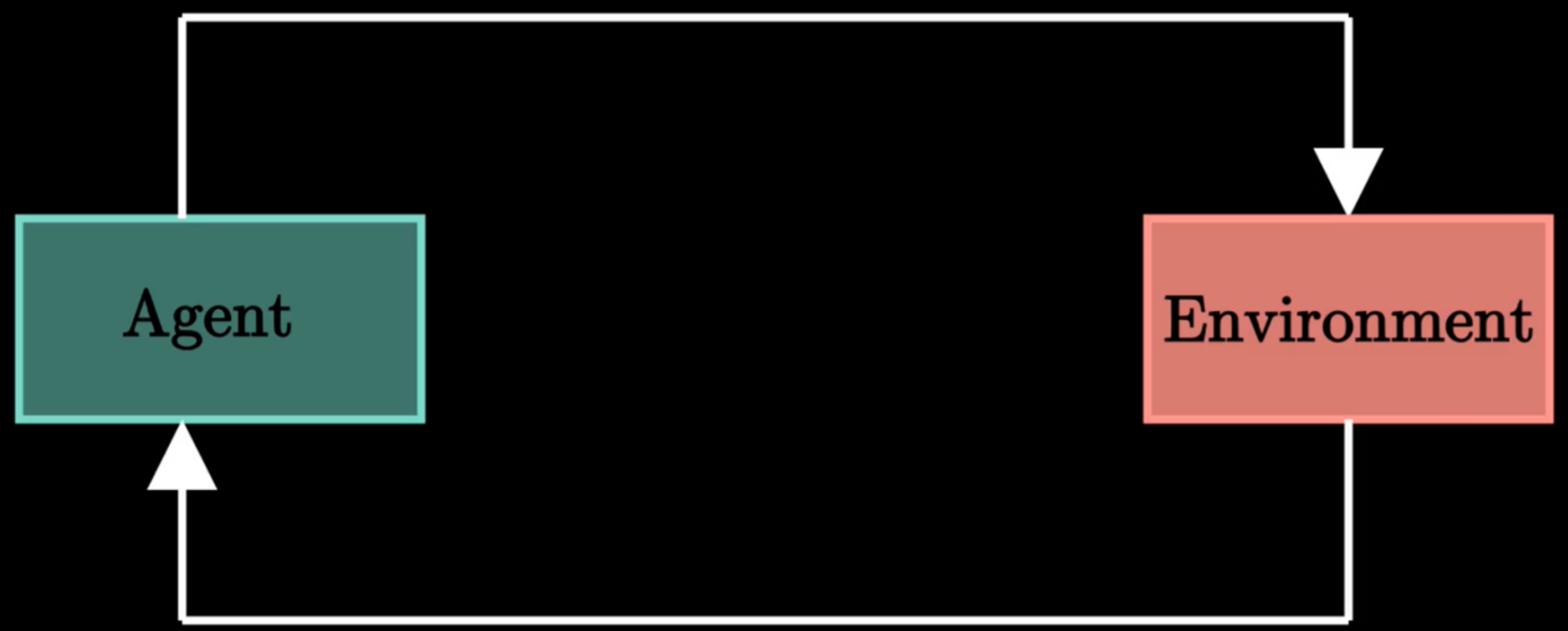


Figure adopted from
David Silver's RL course

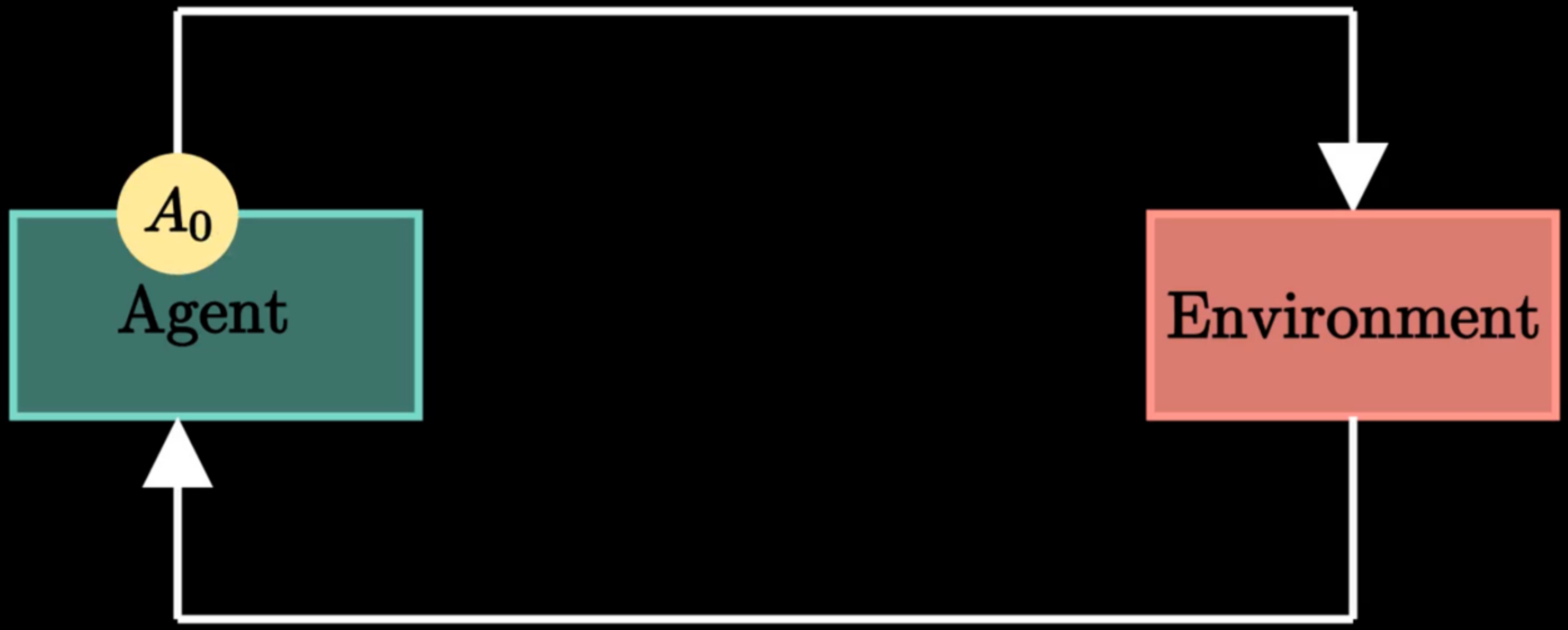
Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



Markov Decision Process (MDP)

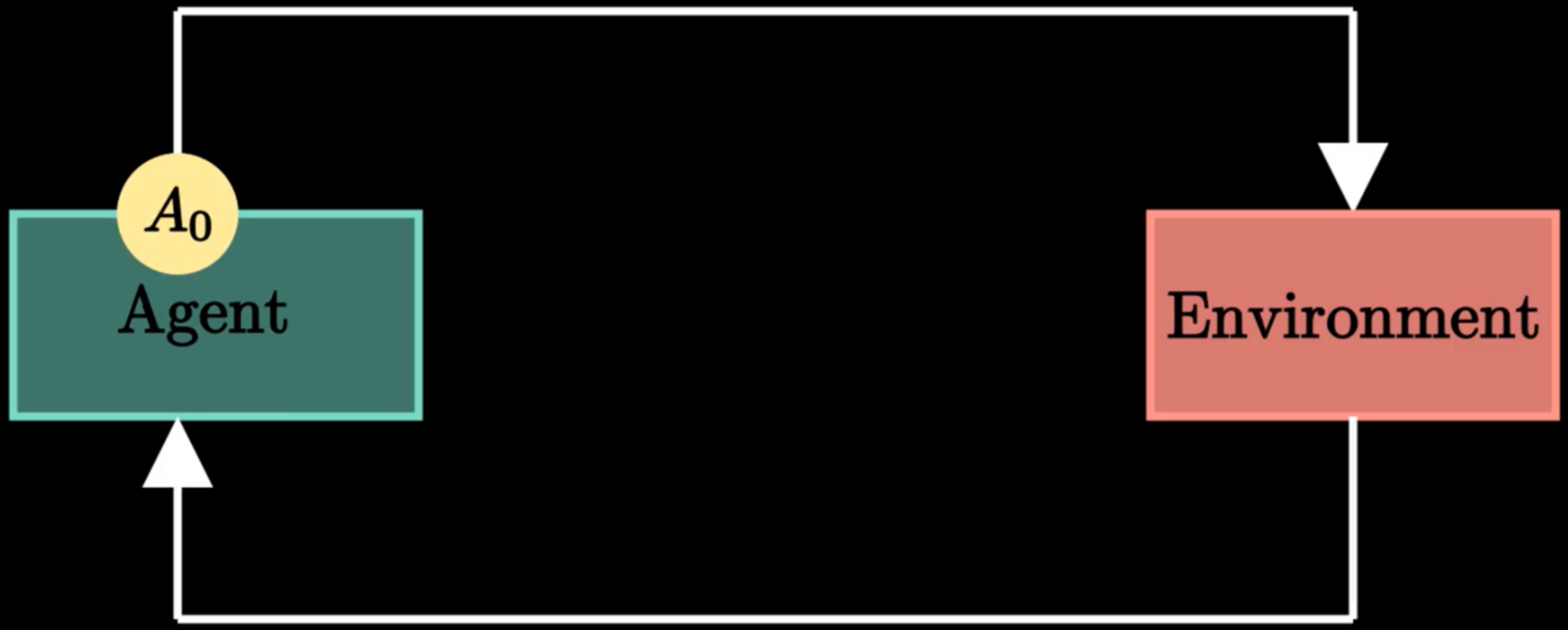
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0, \ A_0,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0, \quad A_0,$

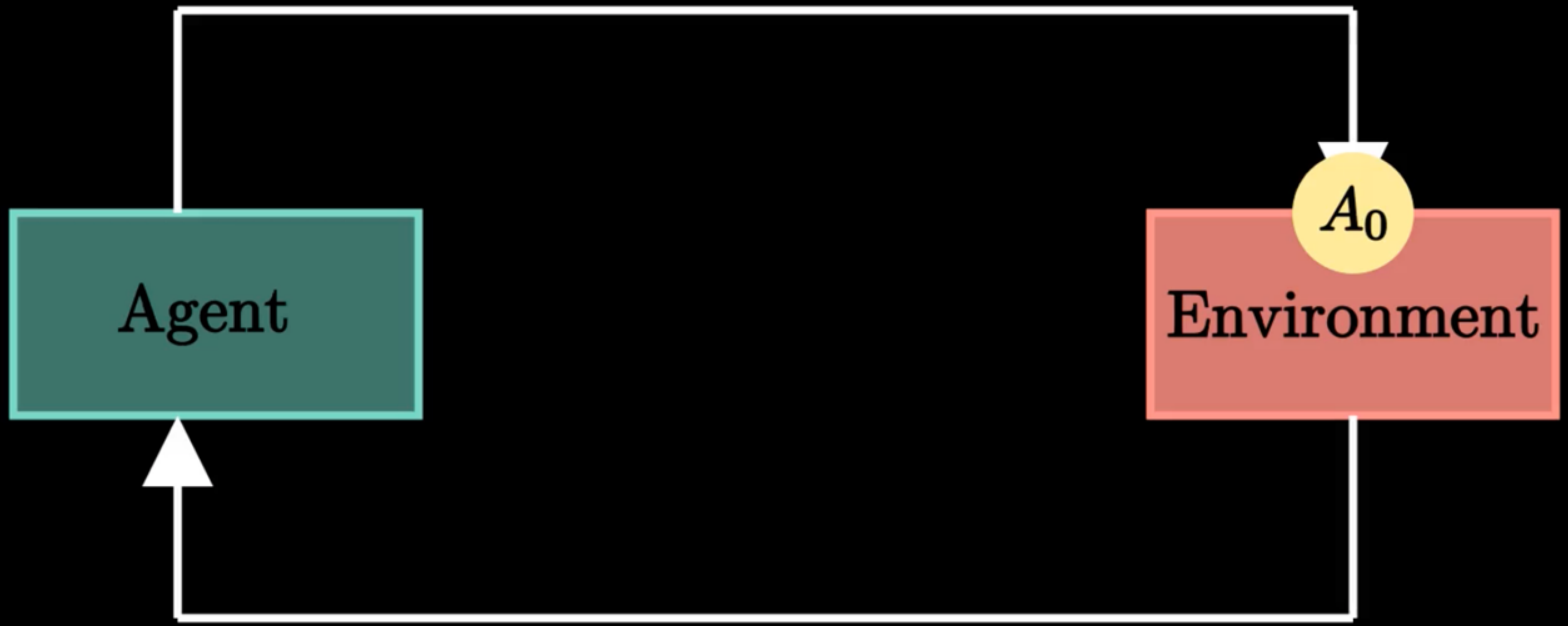
Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0,\ A_0,$

Markov Decision Process (MDP)

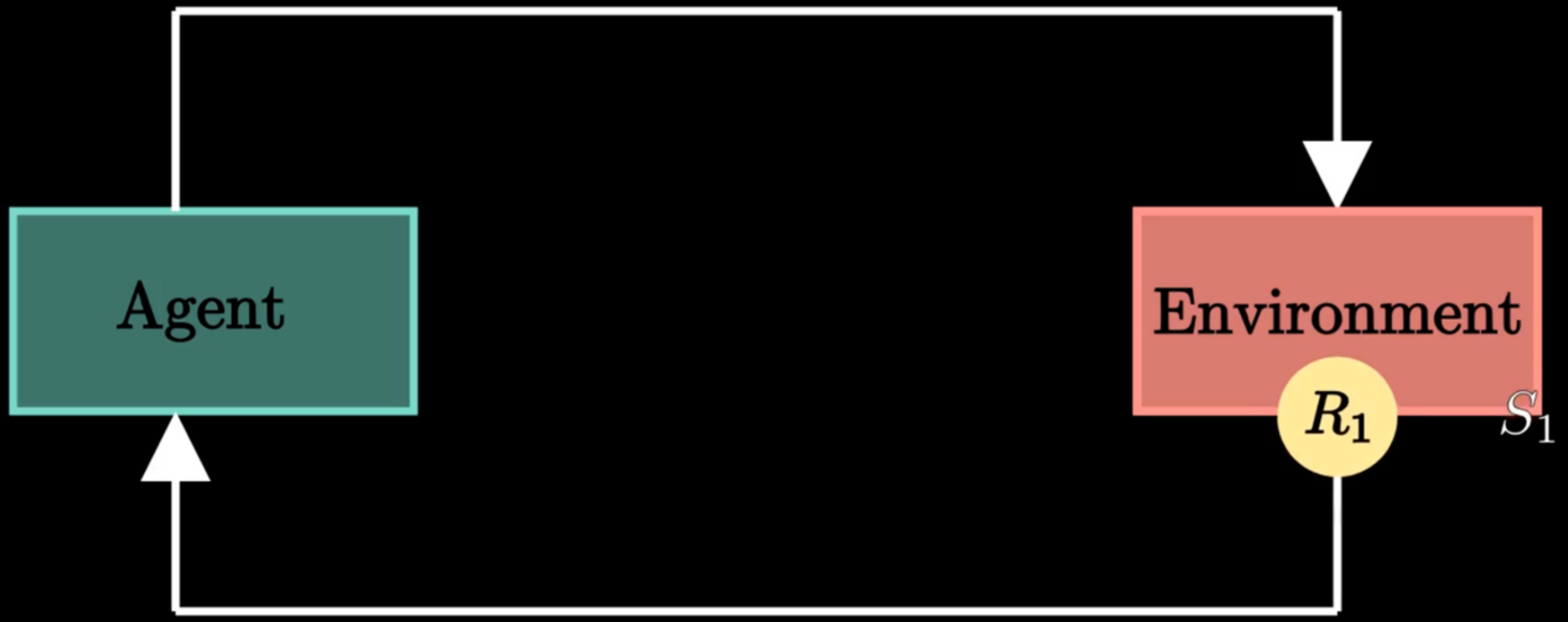
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0, \; A_0,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0,\ A_0,\ R_1,\ S_1,$

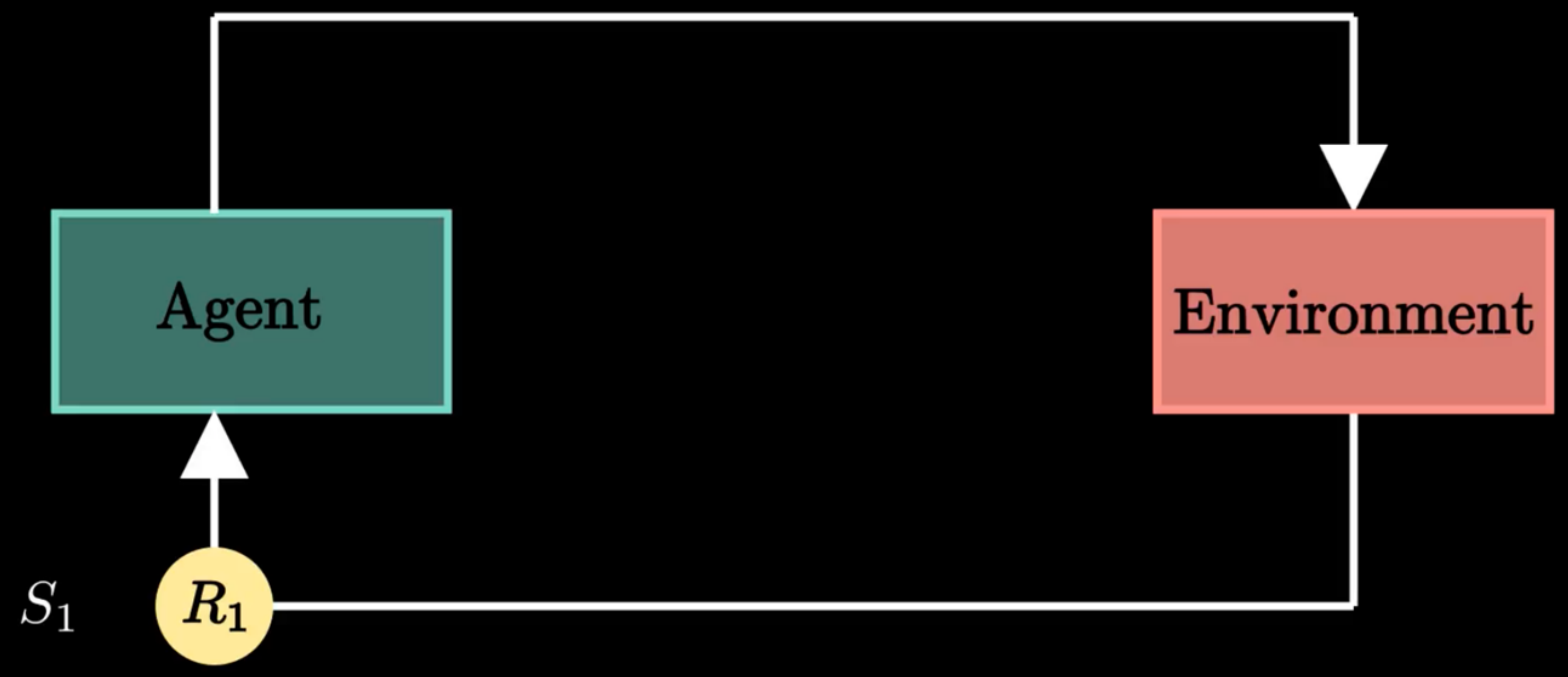
Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0, \ A_0, \ R_1, \ S_1,$

Markov Decision Process (MDP)

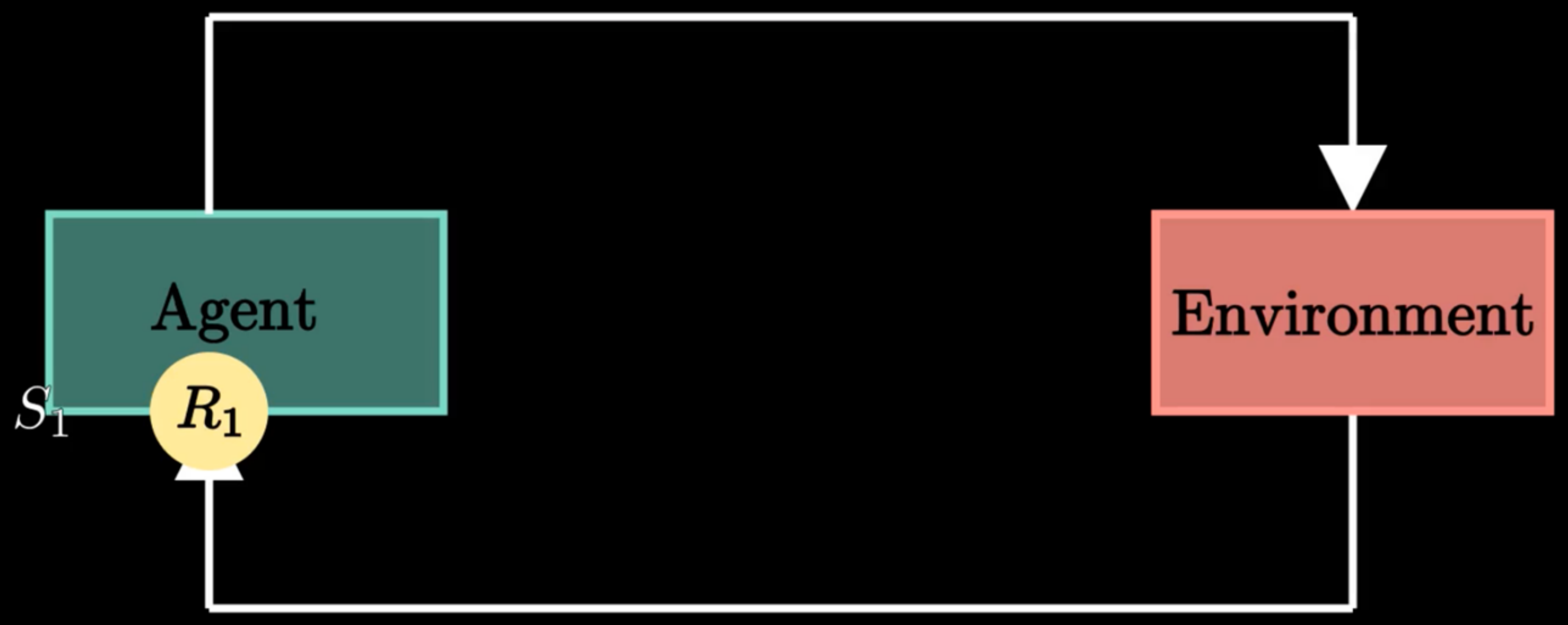
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0,\ A_0,\ R_1,\ S_1,$

Markov Decision Process (MDP)

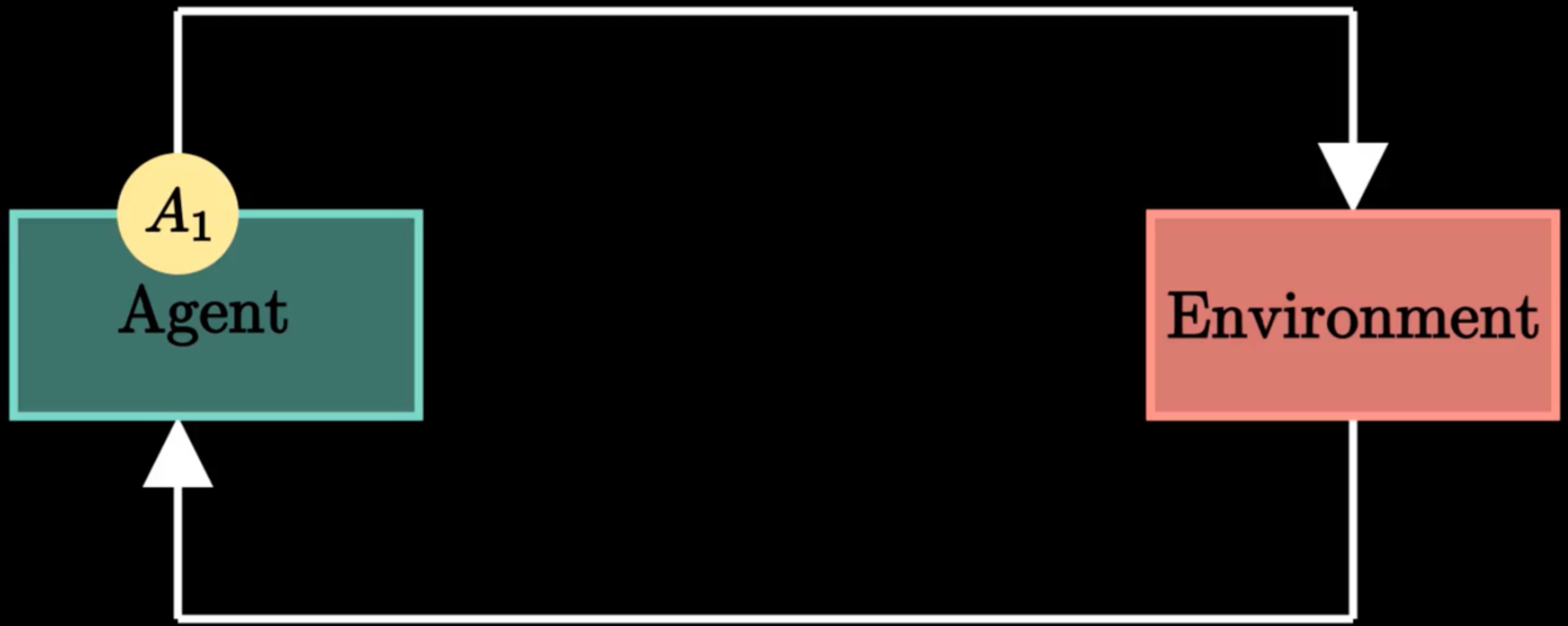
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0, \; A_0, \; R_1, \; S_1,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0, \ A_0, \ R_1, \ S_1, \ A_1,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0, \; A_0, \; R_1, \; S_1, \; A_1,$

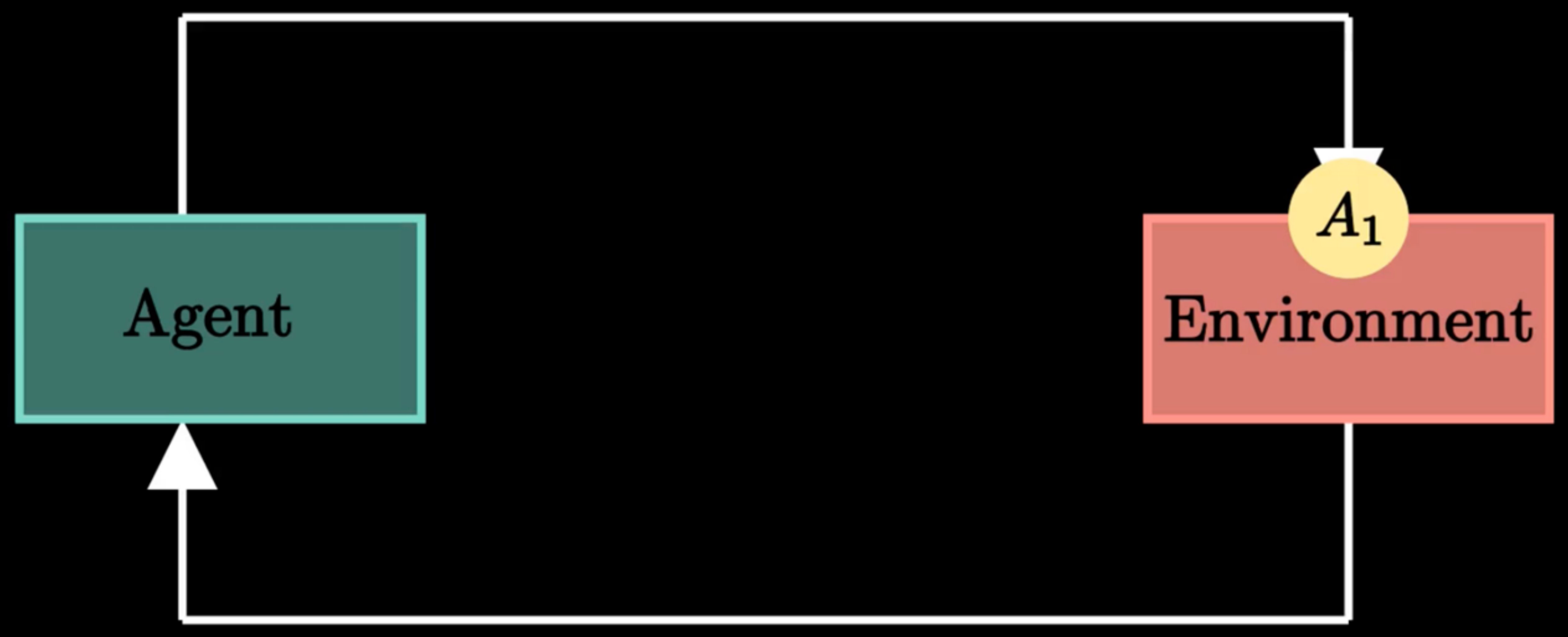
Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0, \; A_0, \; R_1, \; S_1, \; A_1,$

Markov Decision Process (MDP)

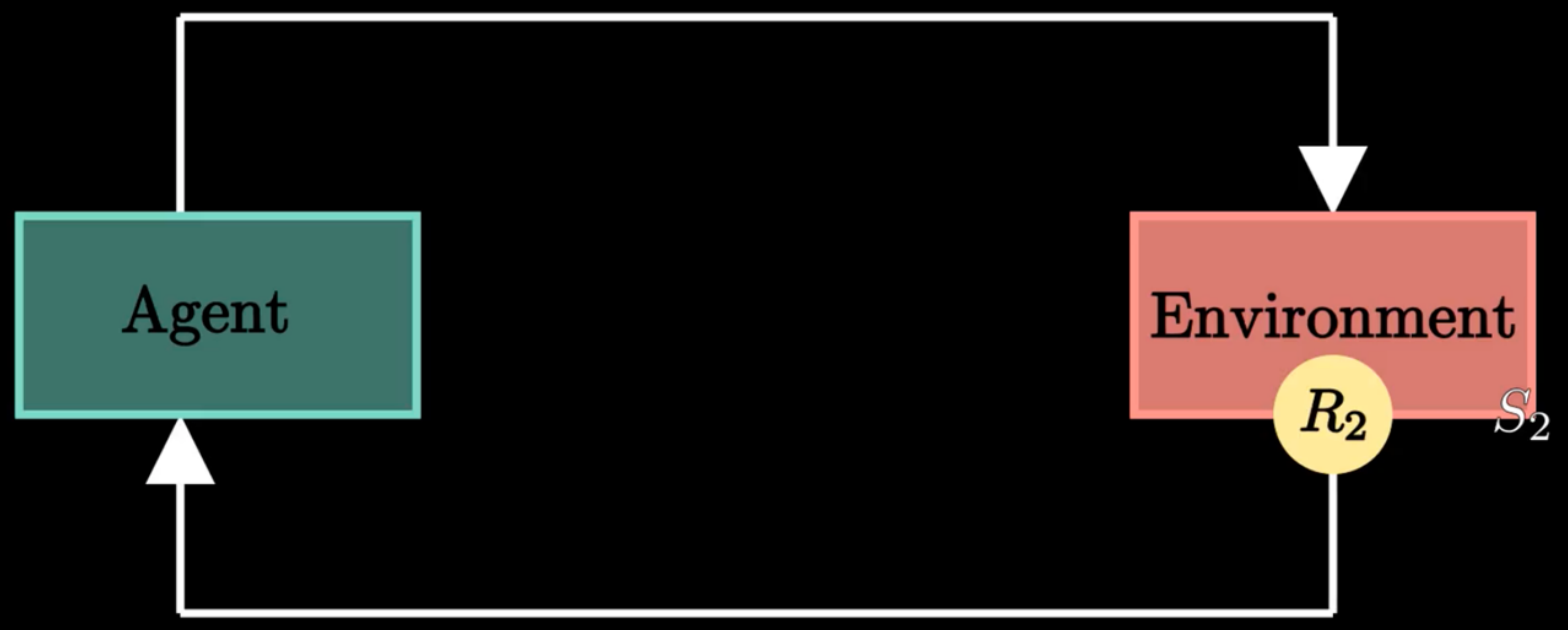
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$$S_0, \ A_0, \ R_1, \ S_1, \ A_1,$$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,$

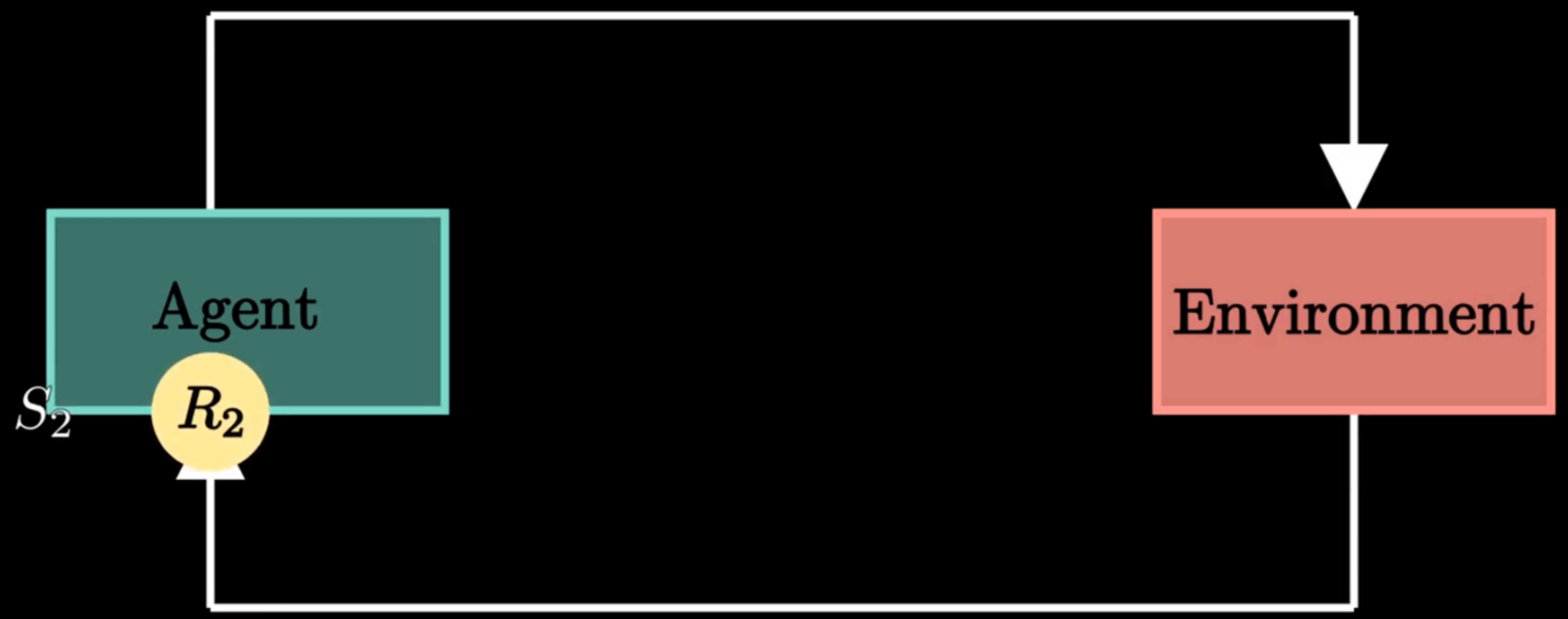
Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,$$

Markov Decision Process (MDP)

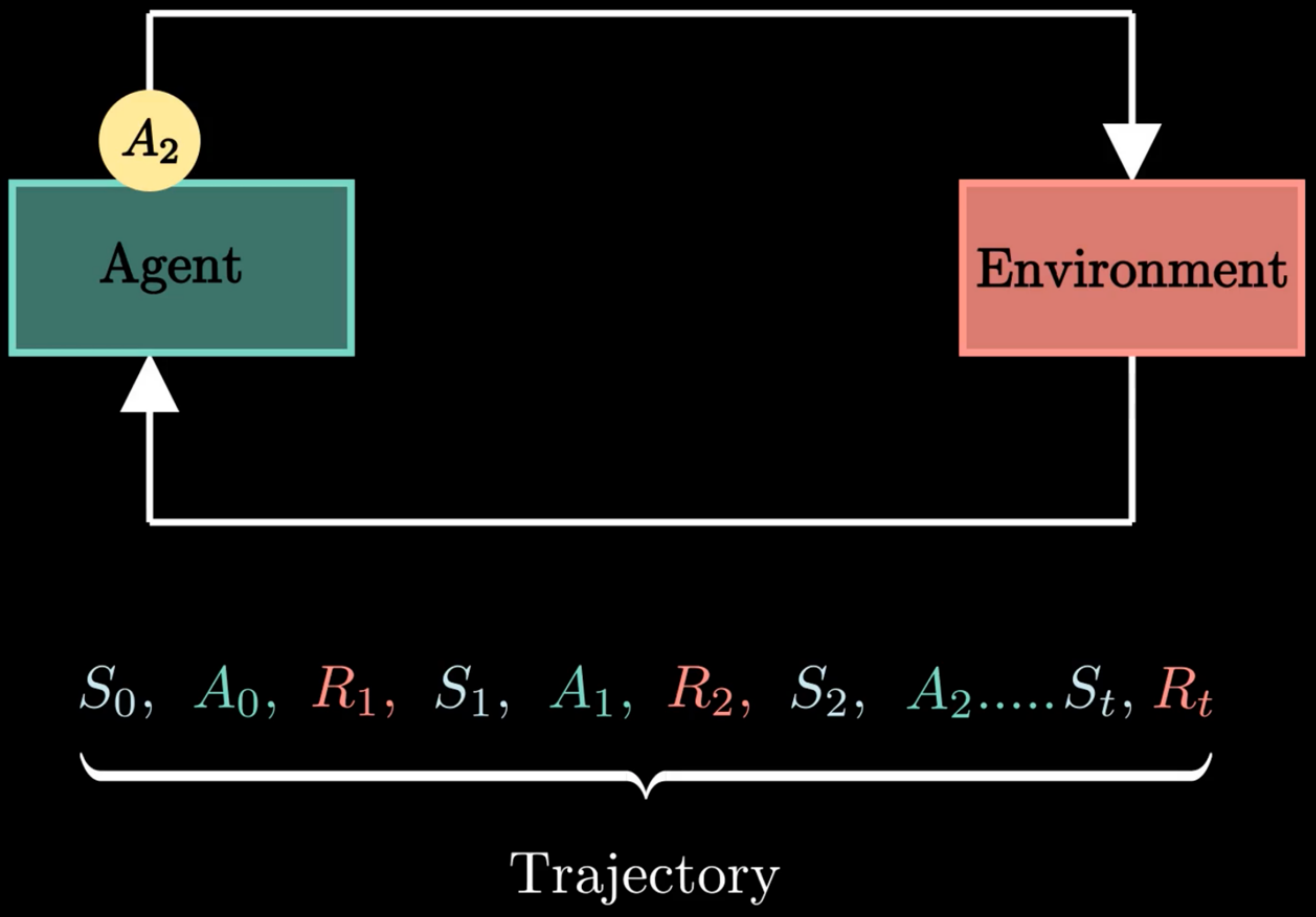
MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.



$S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,$

Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

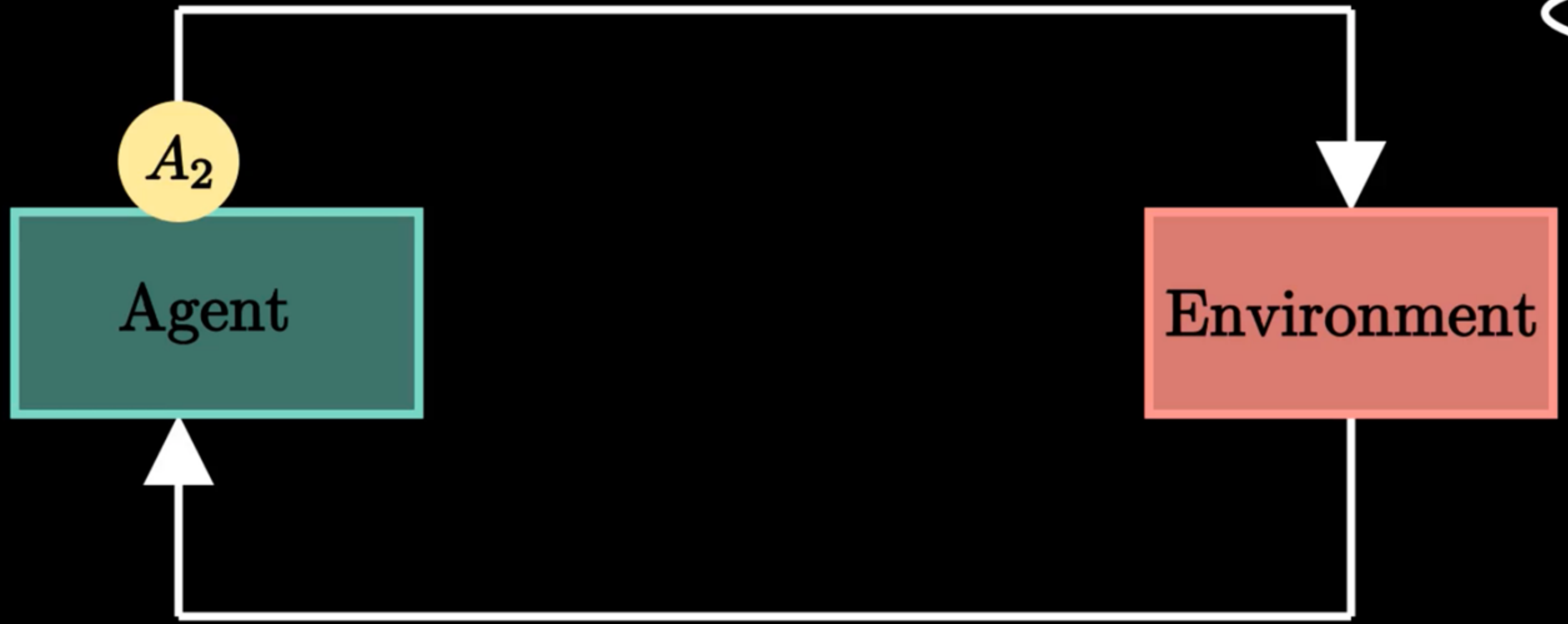


Markov Decision Process (MDP)

MDP provides a method to formalize sequential decision making where actions influence immediate rewards, subsequent states and future rewards.

$$p(s', r|s, a) = P_r\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

p gives the dynamics of the environment



$$\underbrace{S_0,\ A_0,\ R_1,\ S_1,\ A_1,\ R_2,\ S_2,\ A_2 \ldots\ldots S_t,\ R_t}_{\text{Trajectory}}$$

Markov Decision Process

A Markov Decision Process is a tuple $\langle S, A, P, R, \gamma \rangle$

- S is the set of states (finite)
- A is a finite set of actions
- P is the state transition matrix, $P^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$
- R is the reward function
- γ is the discount factor

One matrix for each action in the action space

Tasks

- Episodic Task
- Continuing Task

Episodic Tasks

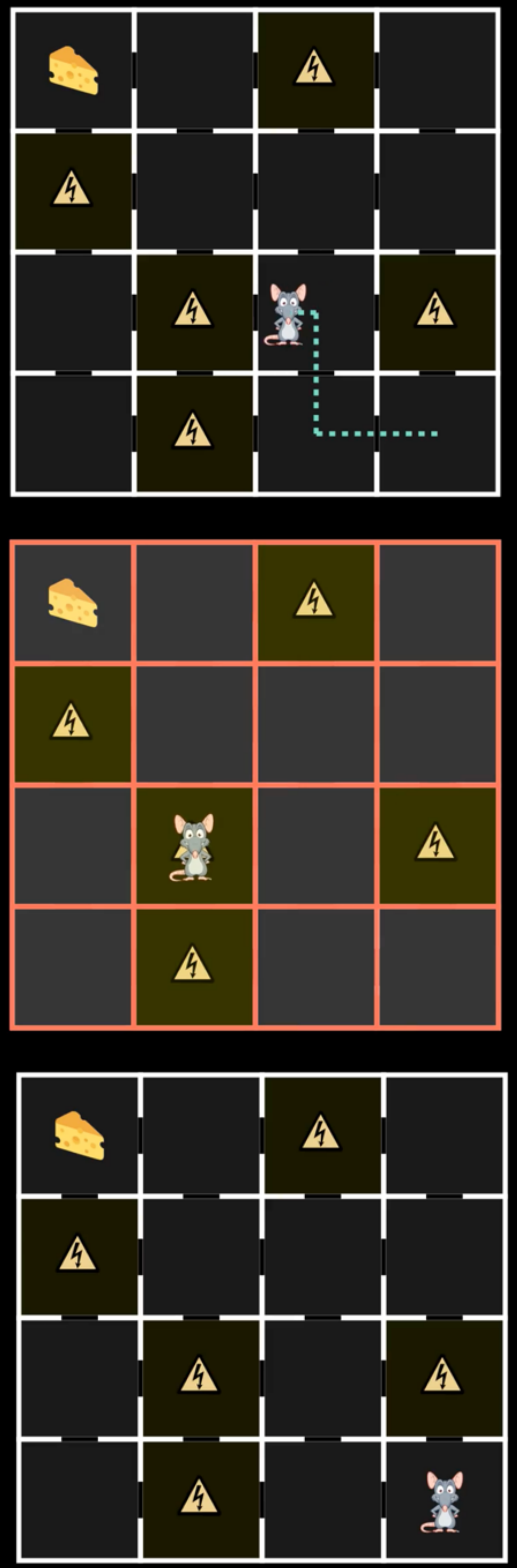
A task that consists of "episodes" which end naturally in a special stated called terminal state S^t.

After the terminal state, we reset back to the start state.

The termination time T is a random variable and varies from one episode to another.

The return in this task type is a simple summation of the rewards.

$$G_t = \sum_{i=t+1}^{T} R_i$$



Continuing Tasks

Task where there is no natural break into identifiable episodes.

It goes on continually without a definitive limit i.e. the final step $T = \infty$.

Hence, a discounting factor γ is introduced when computing the return.

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \text{ where } 0 \leq \gamma \leq 1.$$

γ determines the present value of future rewards: A reward received k steps from now is worth only γ^{k-1} times the immediate reward.

If $\gamma < 1$, the sum is finite. If $\gamma = 0$, we maximise only the immediate reward.

Episodic and Continuing Tasks

To combine episodic and continuing task we introduce an absorbing state.

On termination of an episode, we go to this state that transitions to itself and generates only rewards of zero.

The return for T rewards is the same as the return for ∞ rewards.

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \text{ where } 0 \leq \gamma \leq 1$$

REWARD: 0

R_t

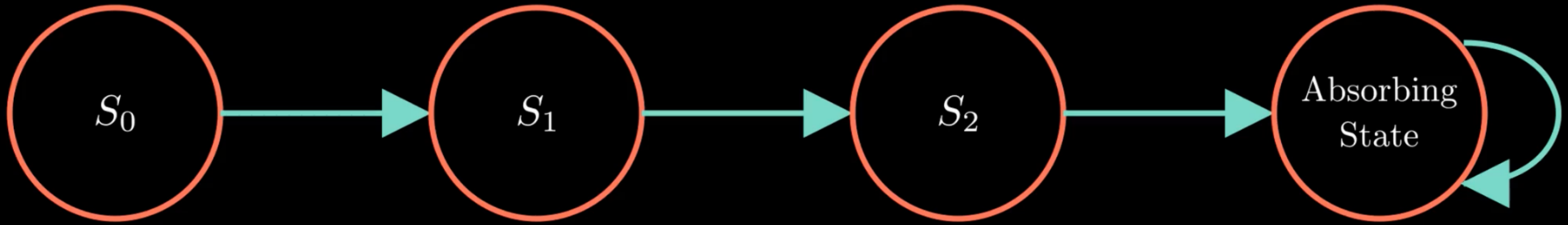
TOTAL RETURN: 0

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

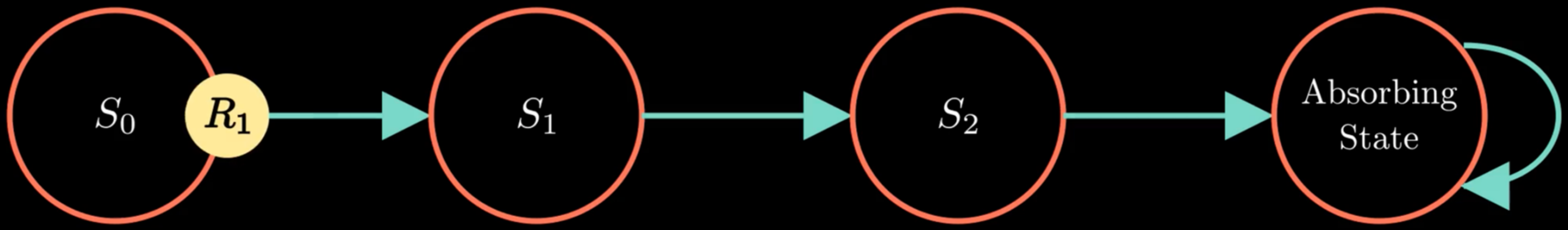
TOTAL RETURN: 0

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

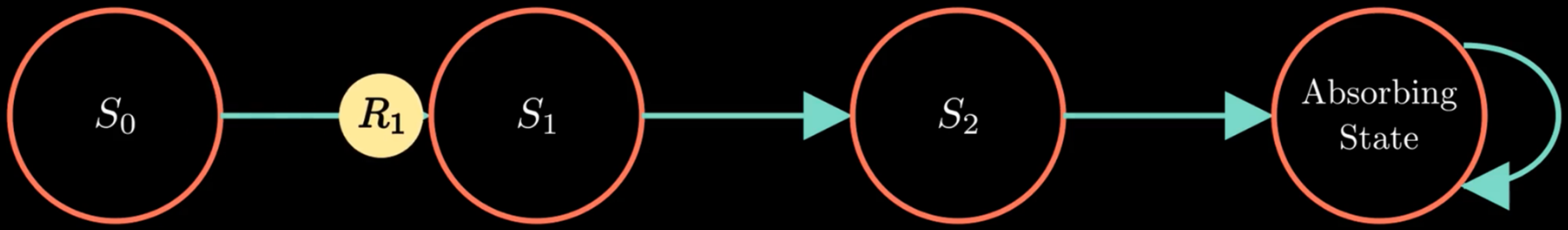
TOTAL RETURN: 0

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

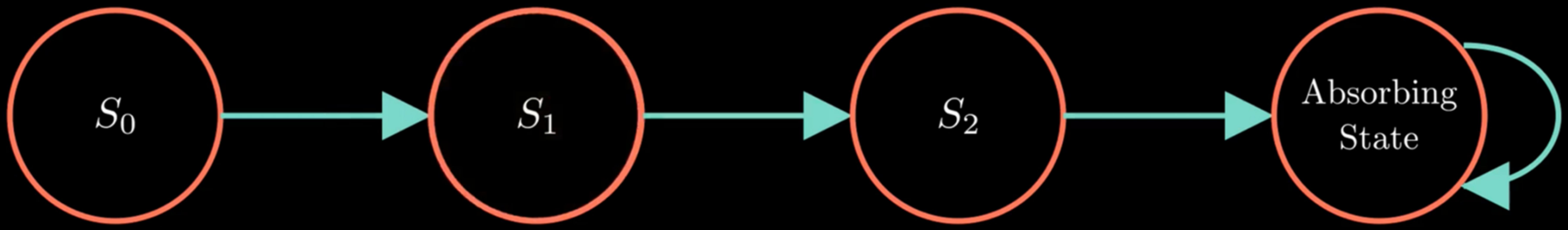
TOTAL RETURN: 10

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 9.0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

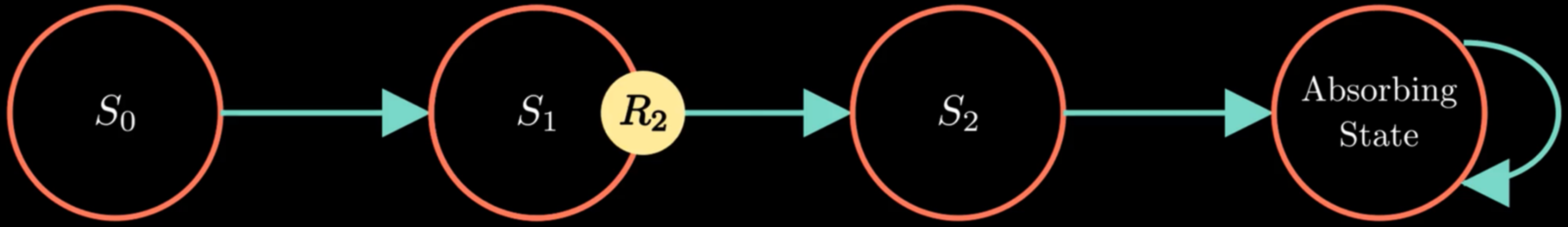
TOTAL RETURN: 10

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 9.0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

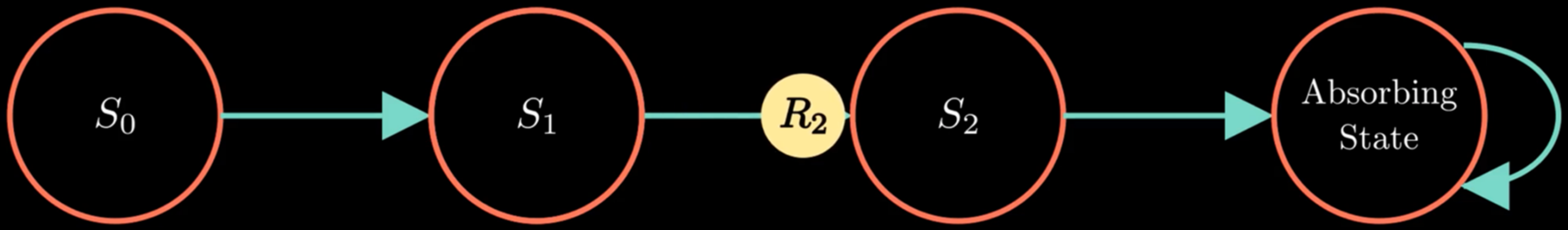
TOTAL RETURN: 10

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 9.0

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

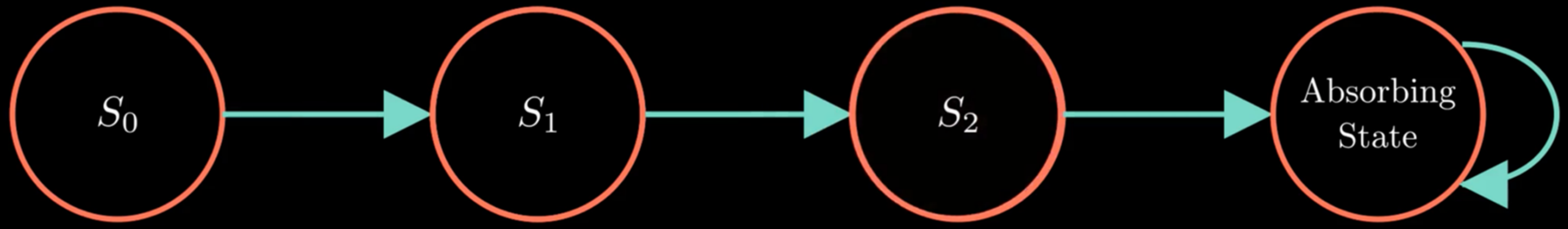
TOTAL RETURN: 20

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 17.1

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

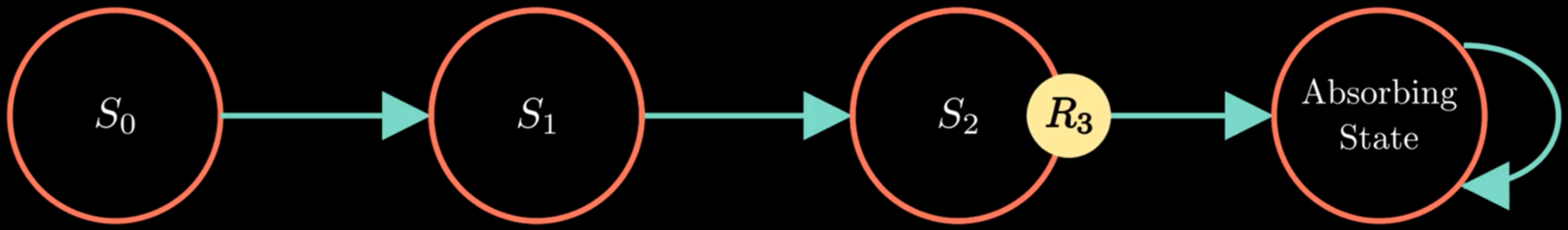
TOTAL RETURN: 20

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 17.1

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

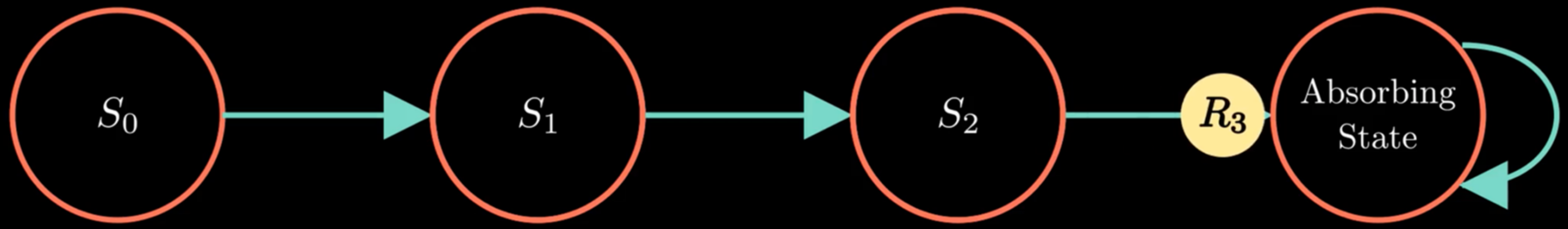
TOTAL RETURN: 20

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 17.1

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 10

R_t

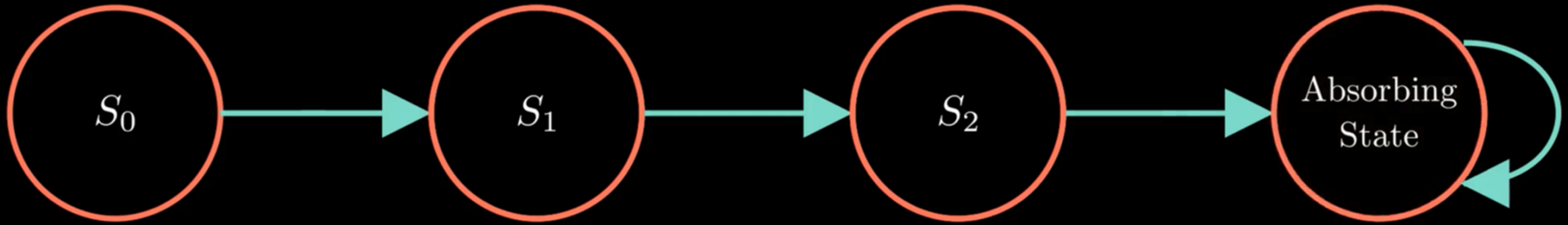
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

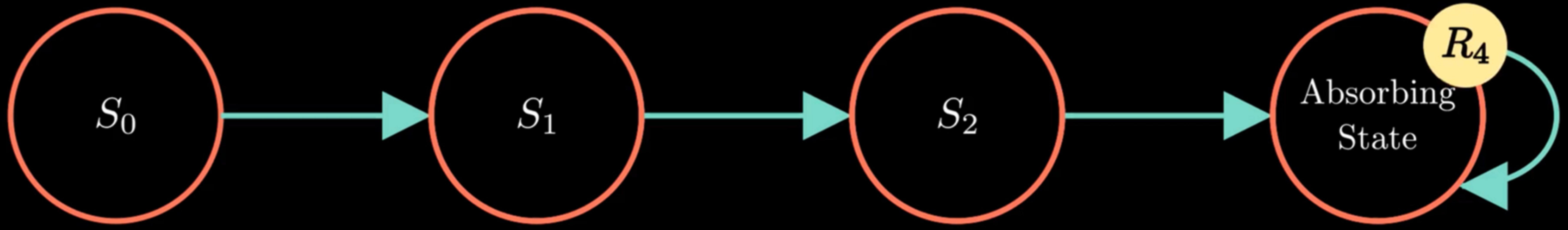
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

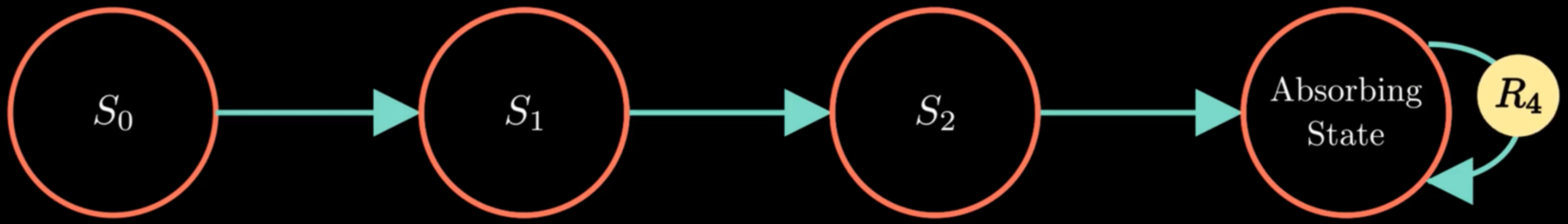
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

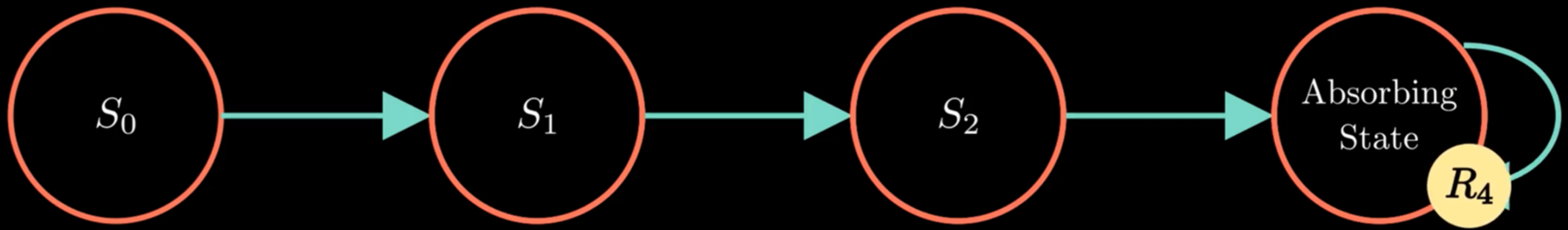
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

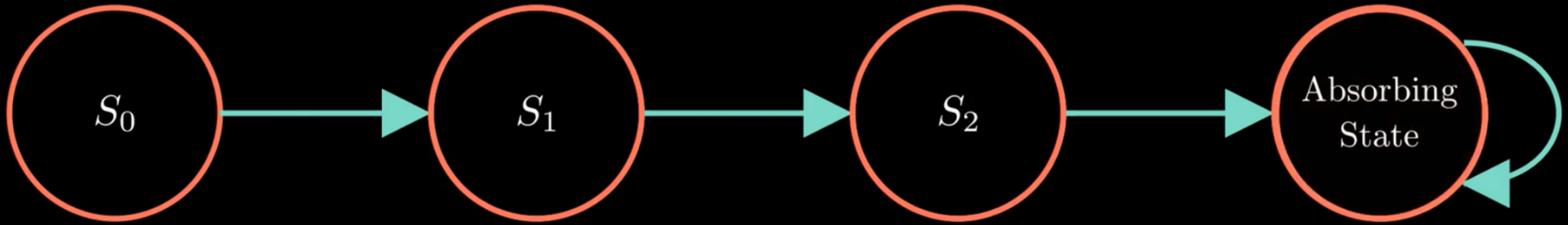
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

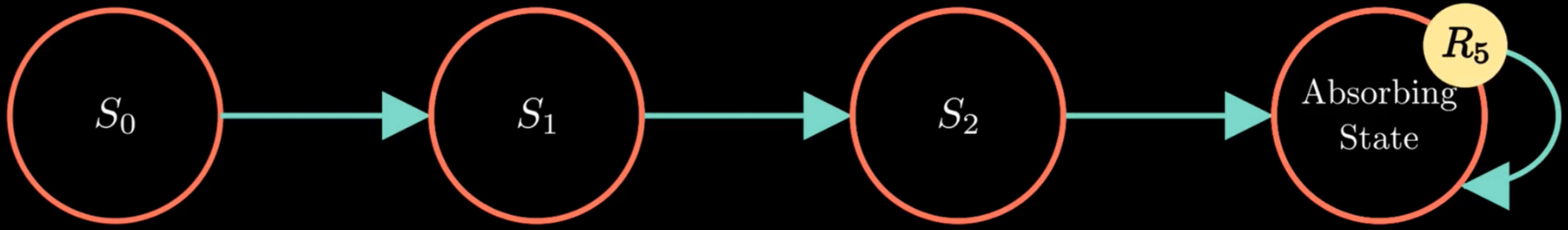
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

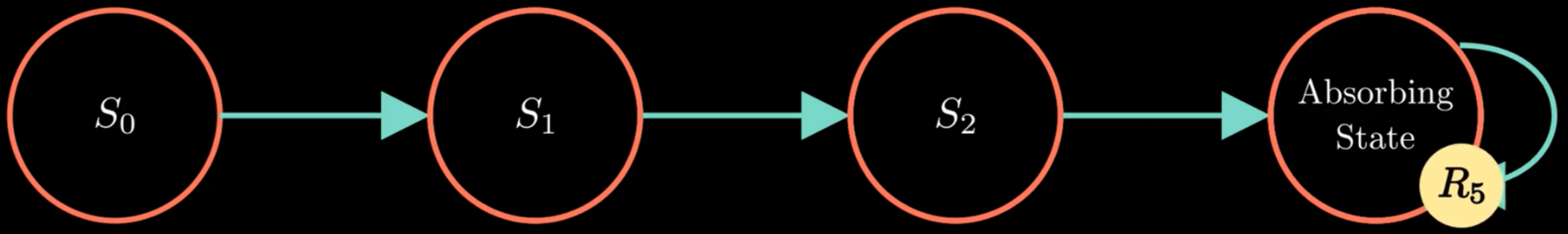
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

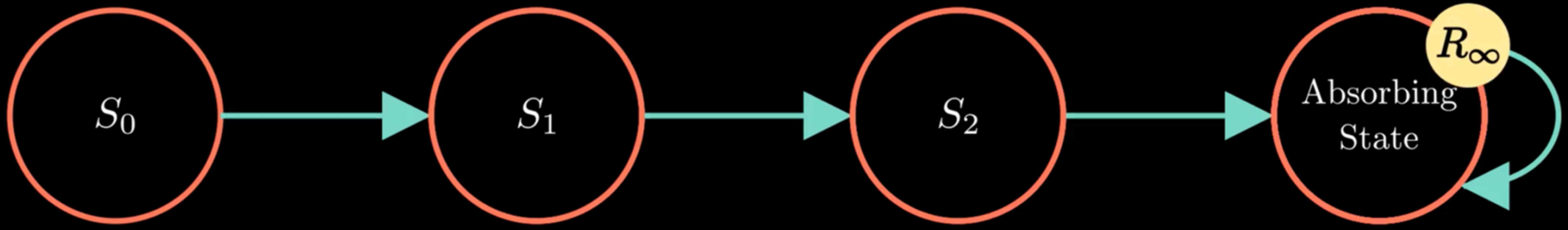
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

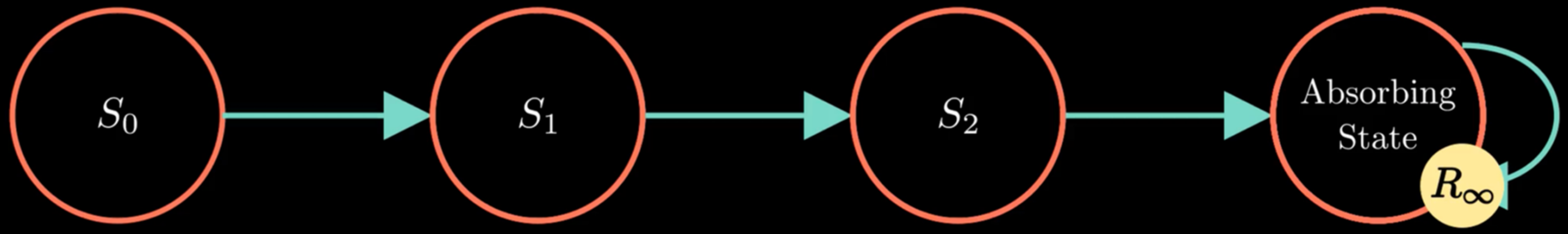
TOTAL RETURN: 30

$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$



Rewards R_1, R_2, R_3 are equal to 10

All rewards after this are equal to 0

REWARD: 0

R_t

TOTAL RETURN: 30

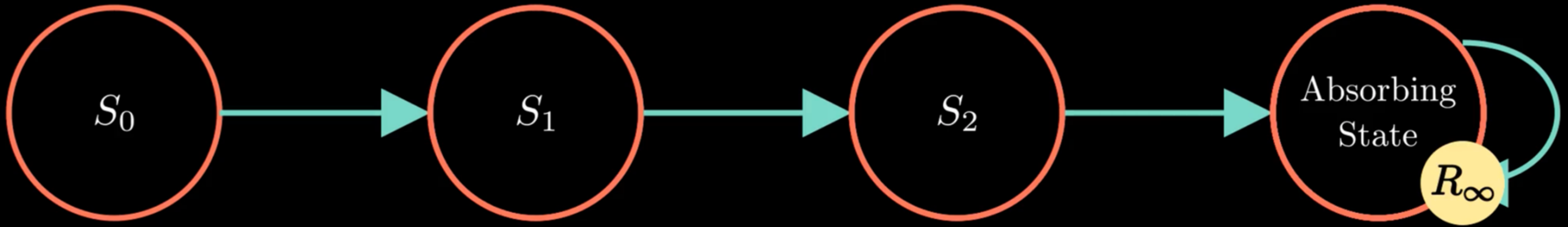
$\sum_{t=0}^{T} R_t$

DISCOUNTED RETURN: 24.39

$\gamma = 0.9$

$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$

Here T can be ∞ and $\gamma = 1$



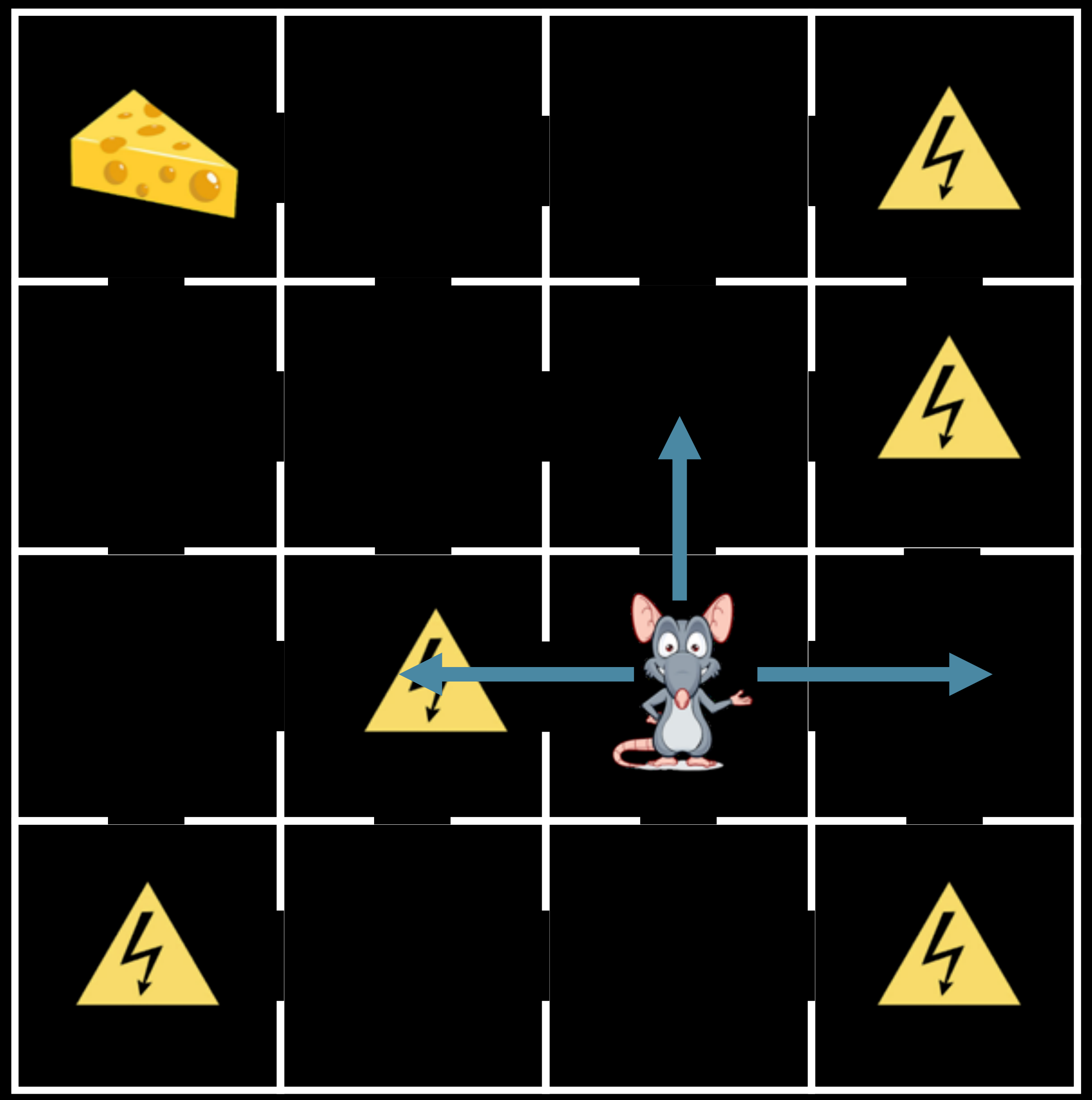
The inclusion of the absorbing state allows episodic and continuous tasks to be formulated using one equation

$$\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$$

Reward in the absorption state is always **0**

Policy

Consider the following scenario

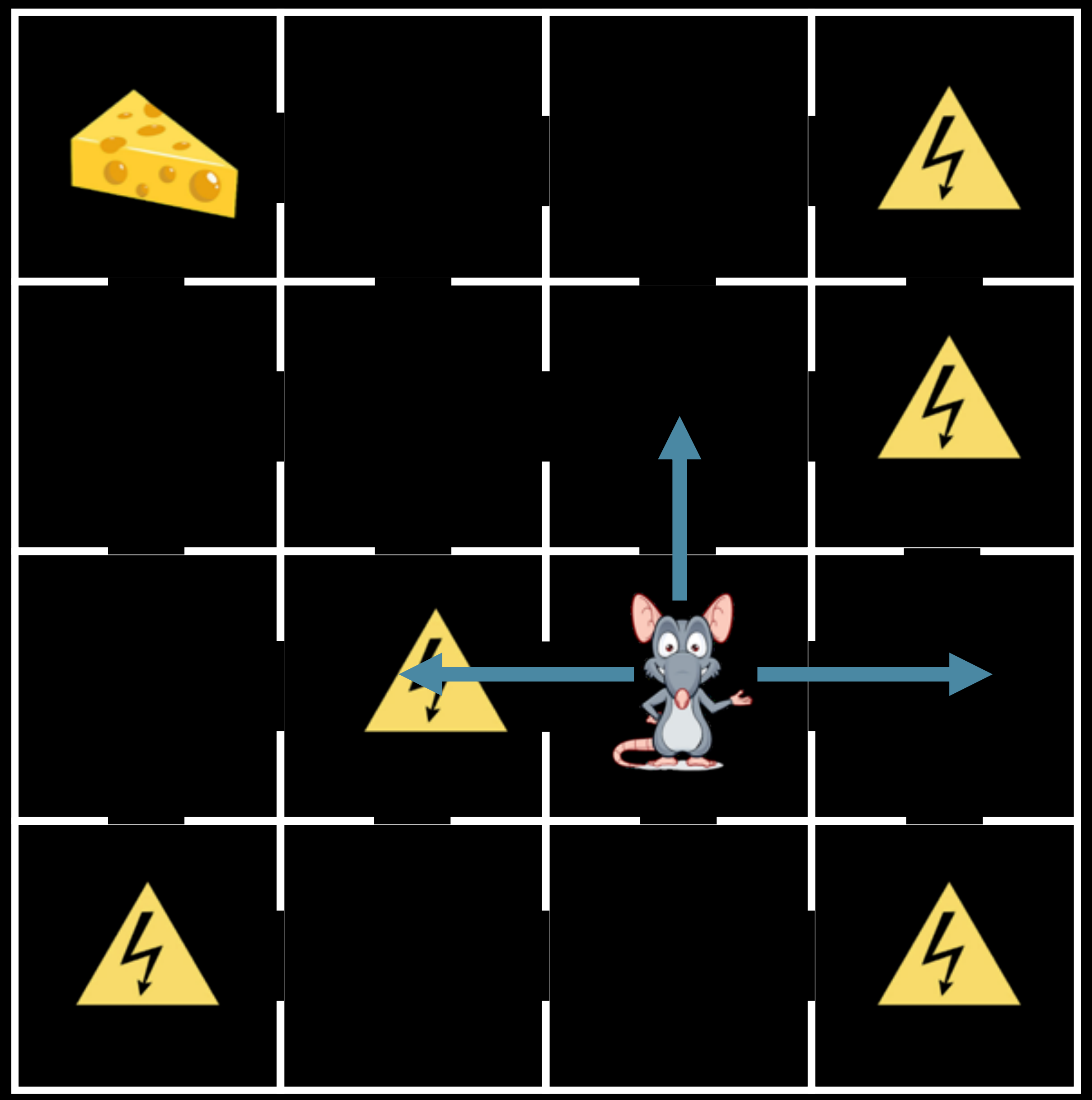


We have a mouse in a state. Let's call this state S_6.

The mouse can take 3 possible actions: Go up, left or right. No backward move in this policy.

Policy

Consider the following scenario

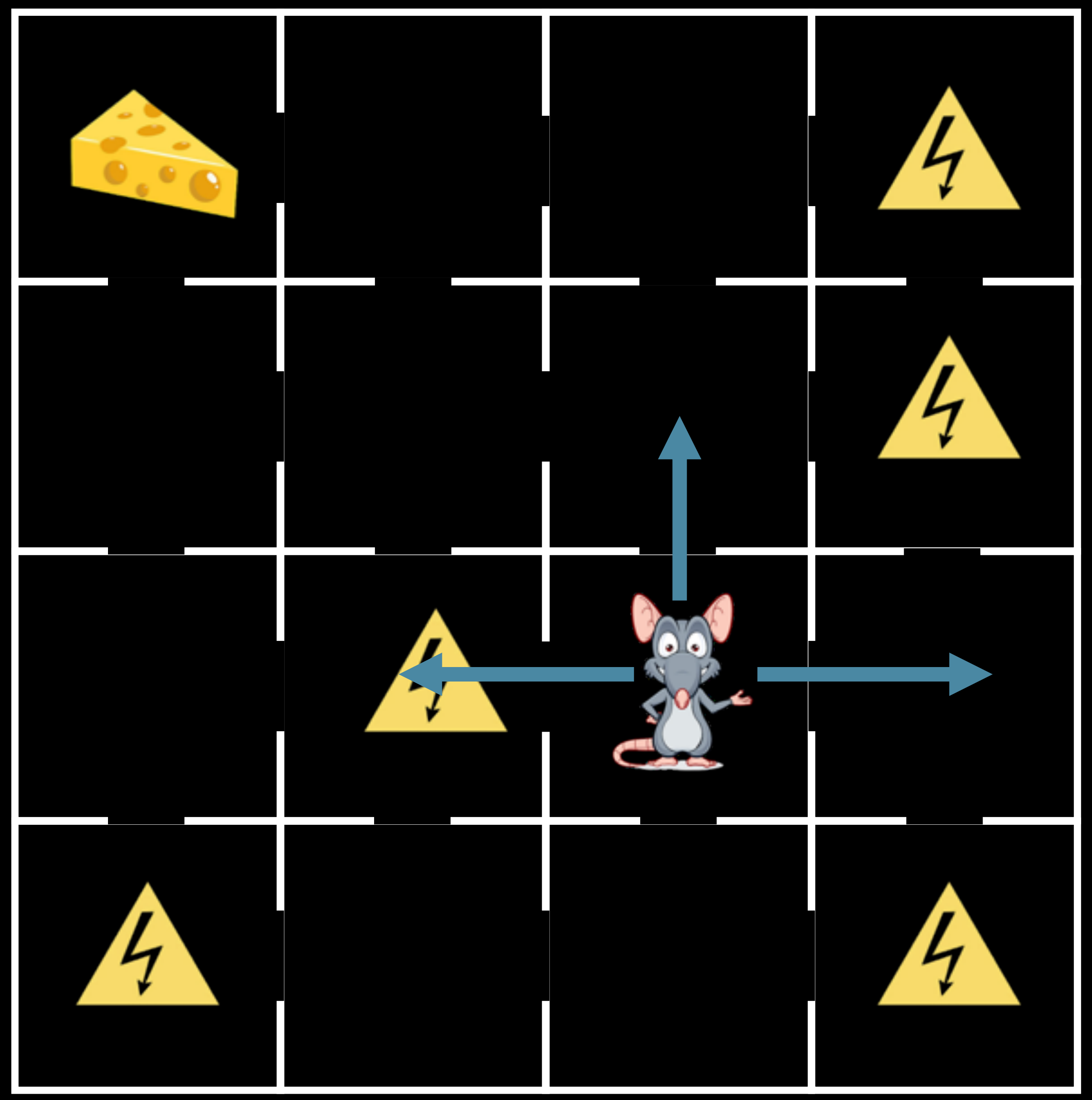


We have a mouse in a state. Let’s call this state S_6.

The mouse can take 3 possible actions: Go up, left or right. No backward move in this policy.

If the mouse was not very smart, then the probability of it taking any one of those actions is $\frac{1}{3}$.

Policy



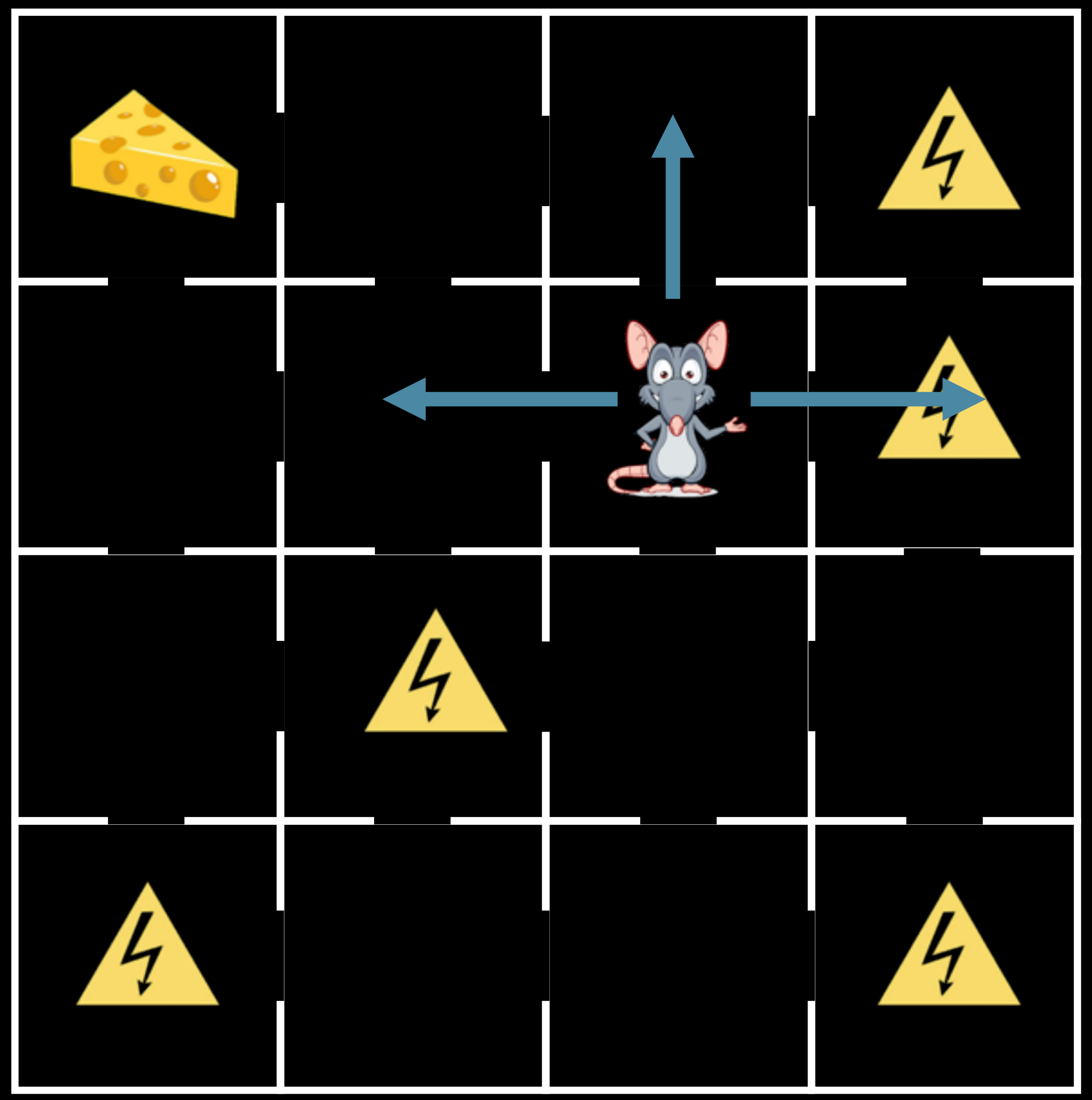
However, a smarter mouse would realize that going left would give it a slight electric shock, hence it drastically reduces the probability of taking that action.

Further, the mouse can smell the cheese from somewhere above it, hence it likely for it to want to try going up.

Thus, the new probability of going

$\text{Up} = \frac{1}{2}, \quad \text{Left} = \frac{1}{6}, \quad \text{Right} = \frac{2}{6}$

Policy

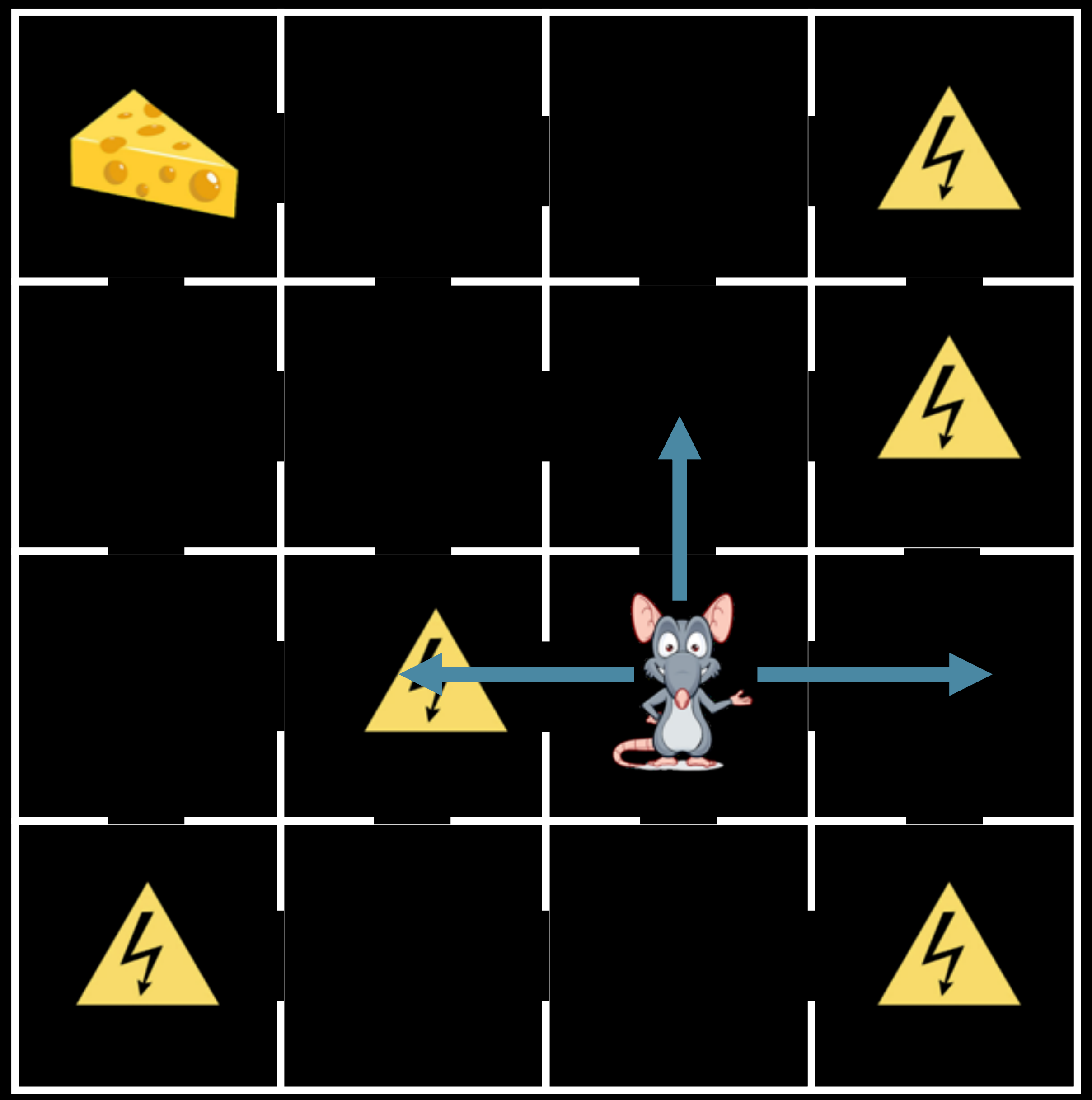


Assume, the mouse takes an action and goes up. It is now in state S_{10}.

Again, we have the same set of possible actions. But the mouse now knows that it will get an electric shock when it goes left and not right like the previous case.

Thus, the probability of taking any action in this state changes.

Policy



This probability, that defines the action taken by an agent in a given state is what is called a Policy.

The probability of an action changes for each state the agent is present in.

Policy

Policy π, provides a probability mapping given a state.

If an agent is said to follow a policy π at time t, then $\pi(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

$$\pi(a|s) = Pr\{A_t = a|S_t = s\}$$

At a time *t*, under policy π that probability of taking action *a* in state *s* is $\pi(a|s)$.

For each state $s \epsilon S$, π is a probability distribution over $a \epsilon A(s)$ i.e. probability distribution for all actions permissible in that state.

Value Function

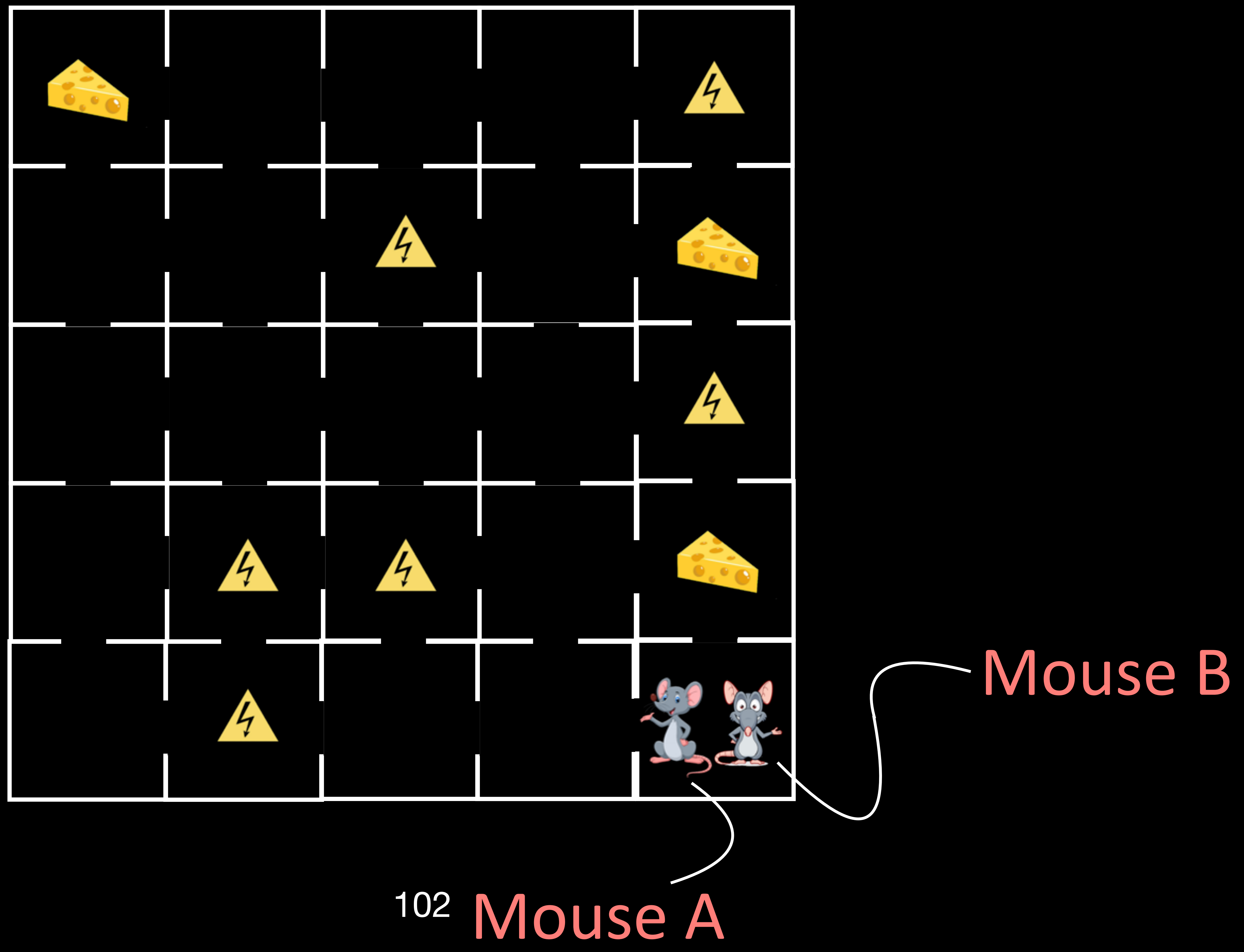
State value function

Action value function

Let’s now think of a scenario where we have 2 mice starting from an intial state, S_0.

Each mouse can take a total of 3 cheese slices. But remember, it cannot come back. So, once it misses a slice it can never eat it.

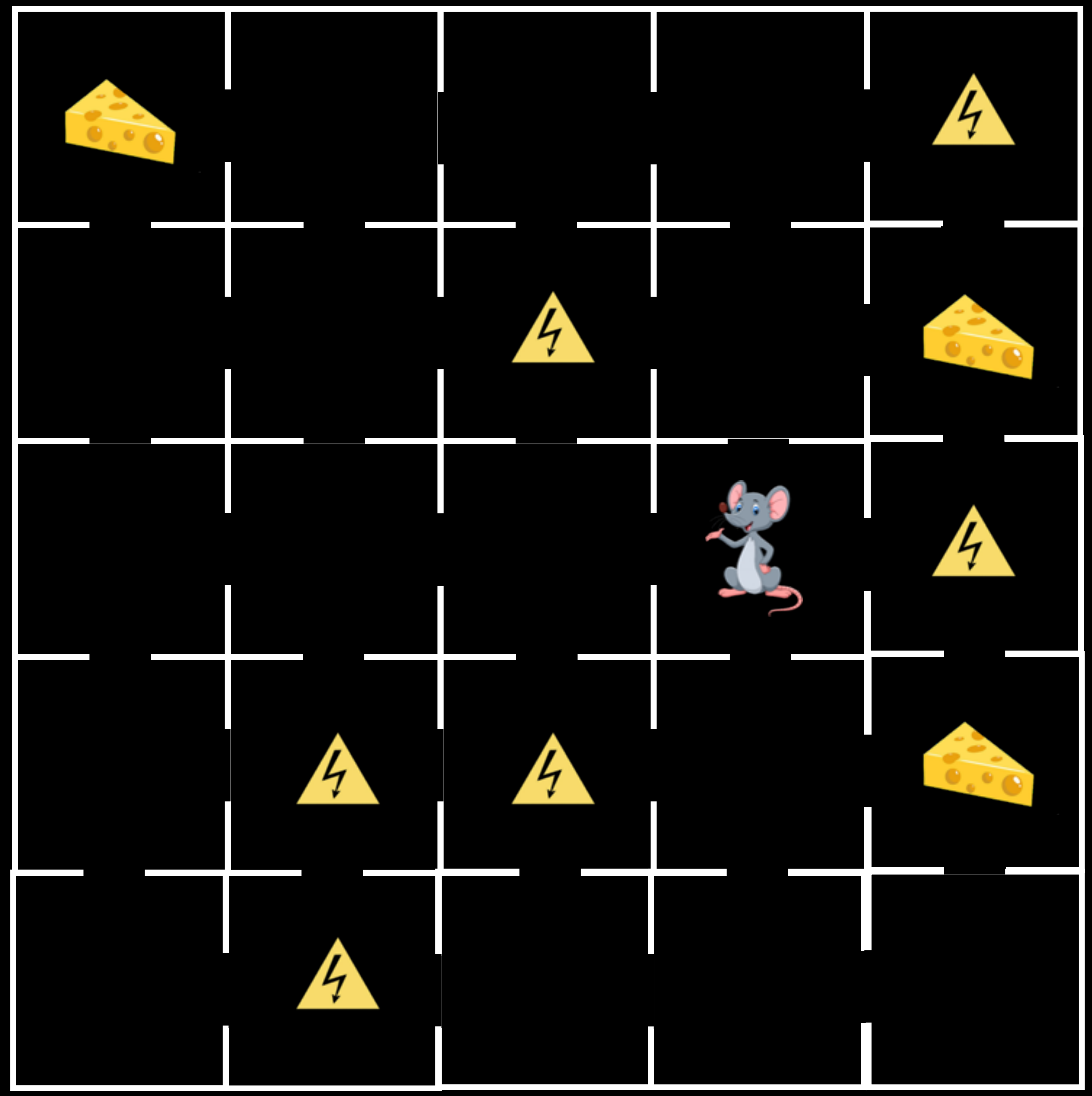
At this point, both the mice can get all the 3 cheese slices.



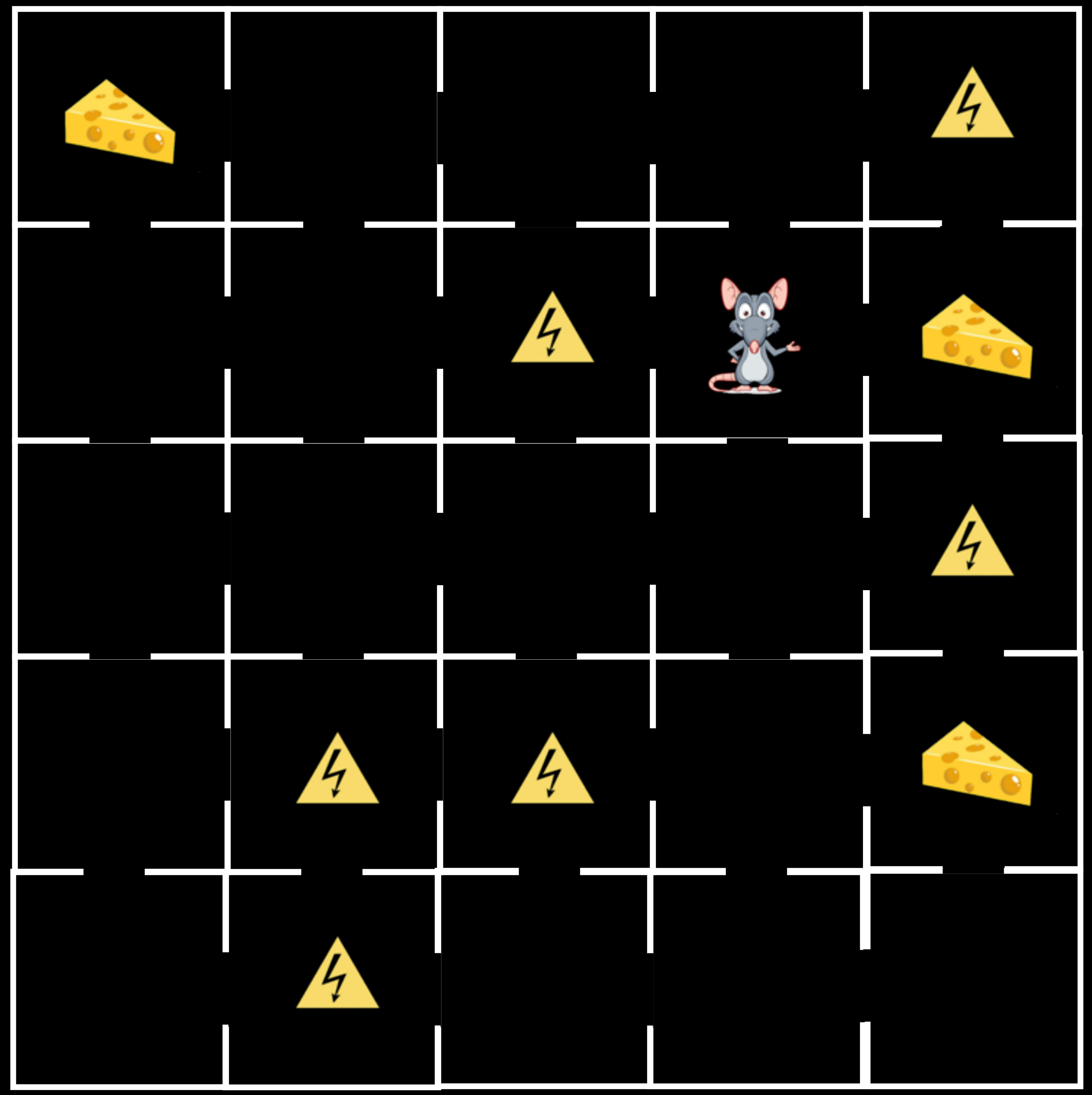
After, a fixed number of actions, each of the mouse ends up in a different state and both of them missed the bottom-most slice.

Both, of them can now have almost 2 slices of cheese.

Mouse A however, has many more paths to reach the top-most cheese



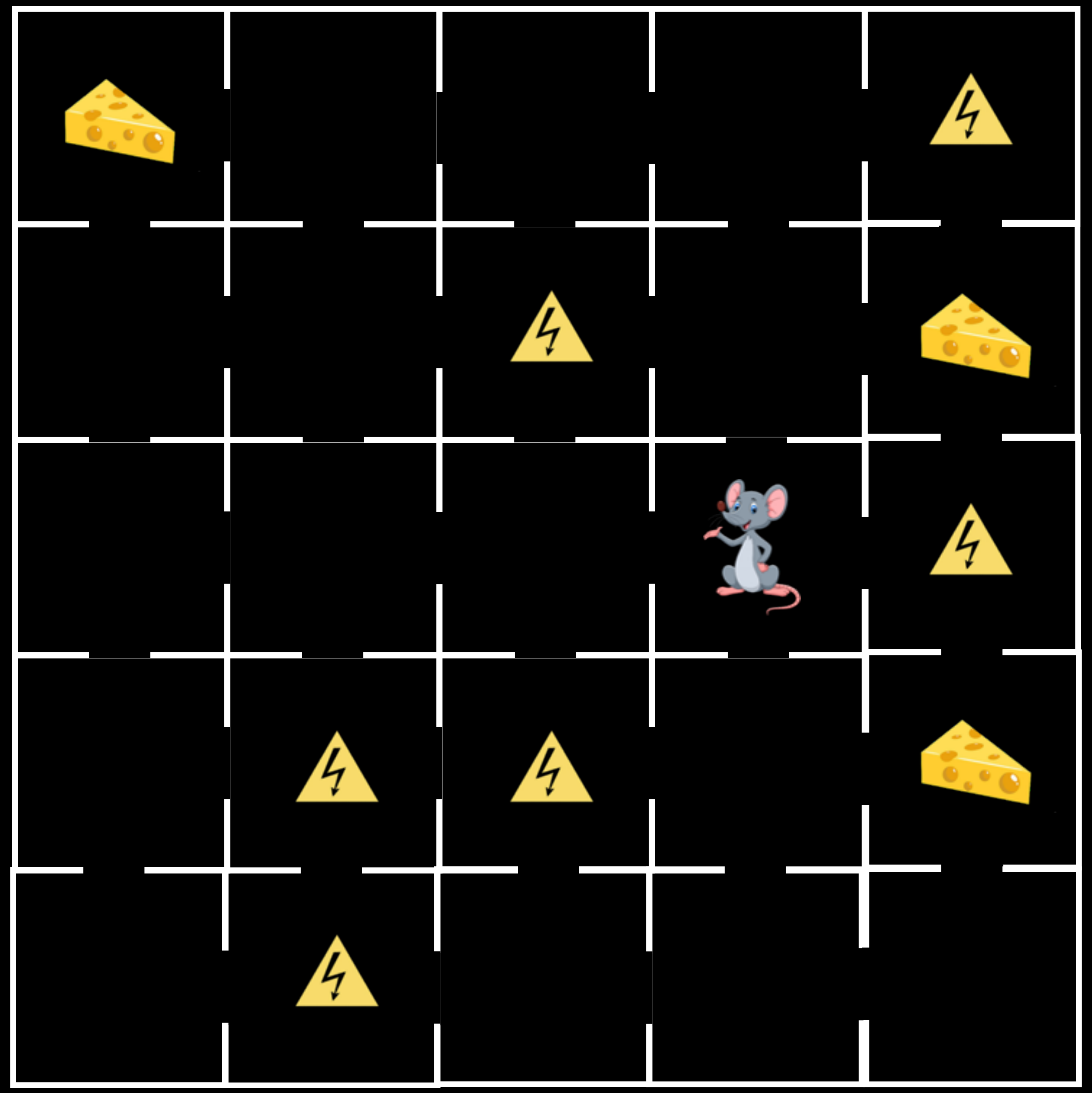
Mouse B however, has fewer possible paths to reach the top-most cheese



After, a fixed number of actions, each of the mouse ends up in a different state and both of them missed the bottom-most slice.

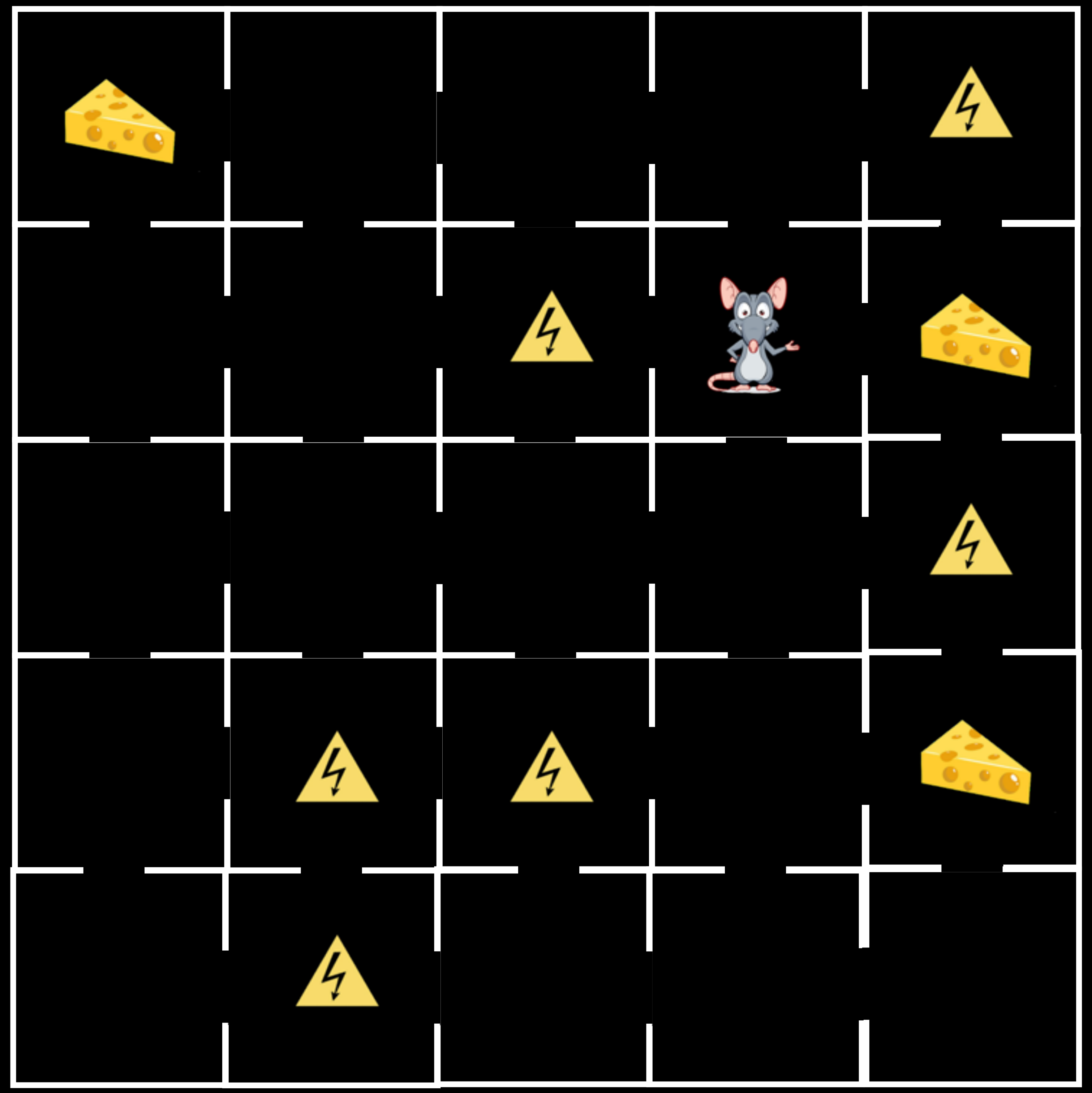
Both, of them can now have almost 2 slices of cheese.

Mouse A however, has many more paths to reach the top-most cheese



If, more movement implies losing energy getting to the cheese, Mouse A is essentially getting lesser reward that Mouse B.

Mouse B however, has fewer possible paths to reach the top-most cheese



Thus, for the same environment, and the same set of possible of rewards, the actual reward varies for different states.

This actual reward got when the agent is in a particular state is called the State Value Function or Value Function of a state.

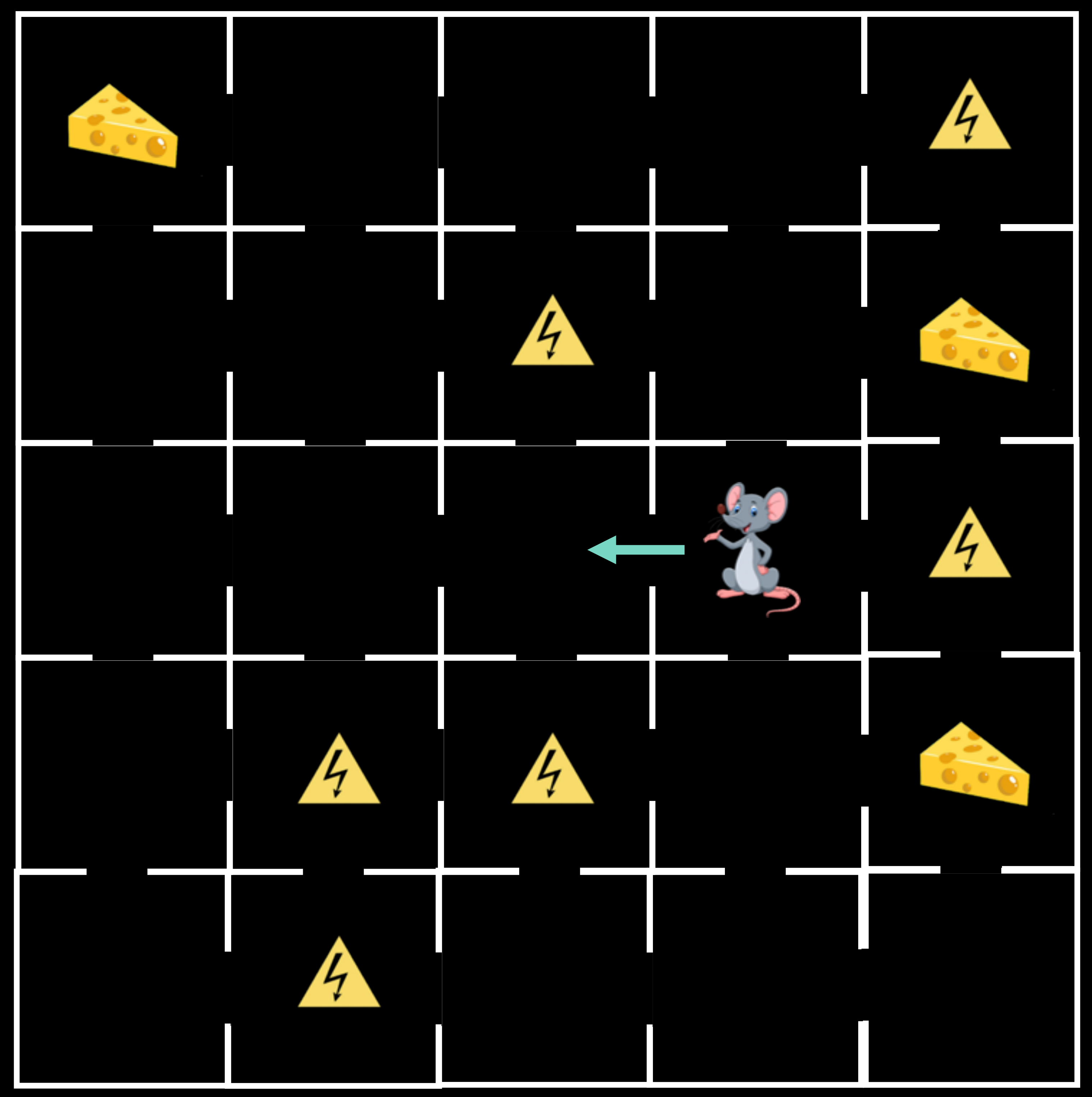
Value function $v_\pi(s)$ of a state s under a policy π is the expected return when starting at s and then thereafter following policy π.

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s]$$

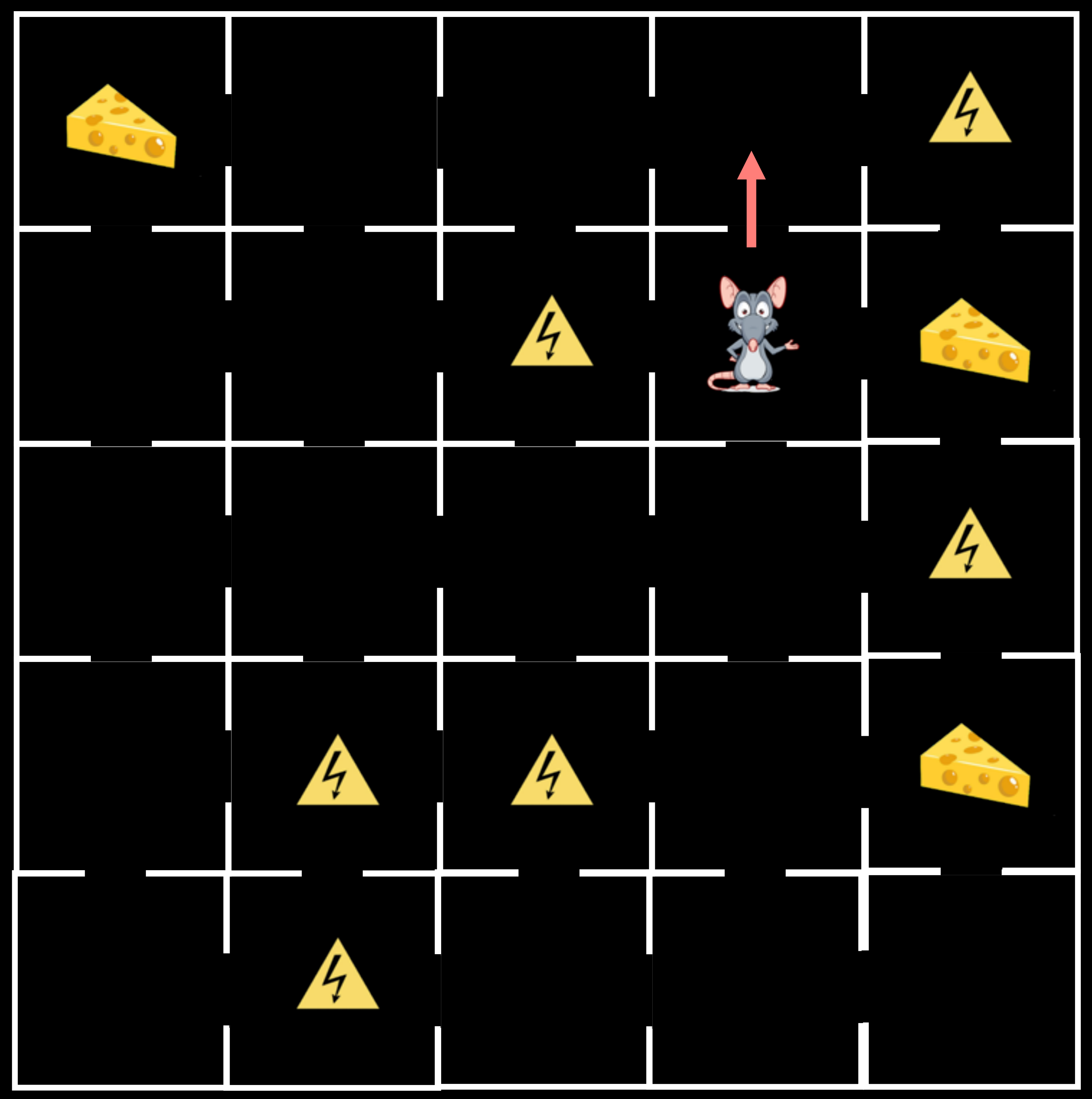
$\mathbb{E}$ is the expected value of random variable given that the agent follows policy π and t is any time step.

Time for some action!

Mouse A goes left

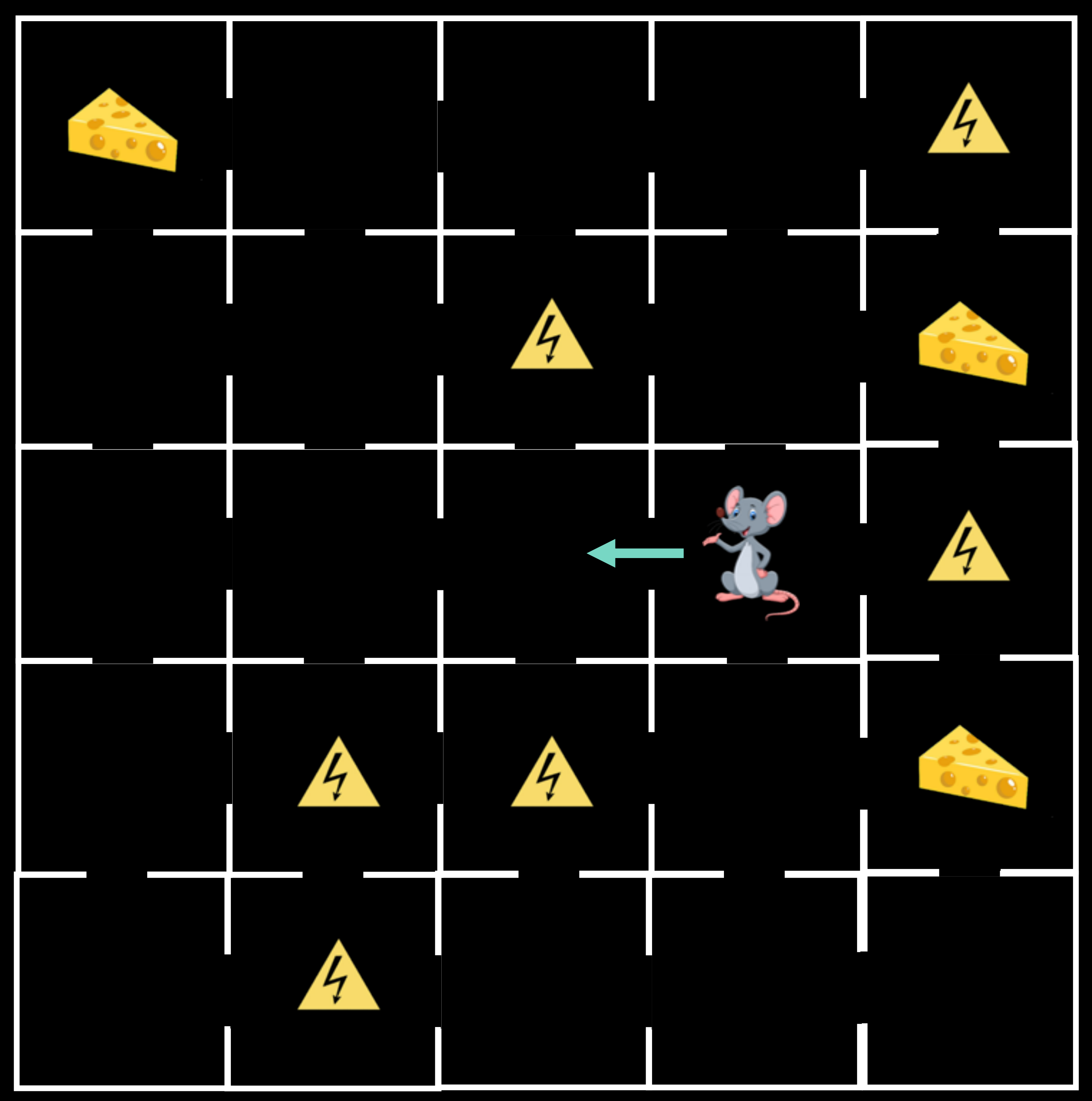


Mouse B goes up

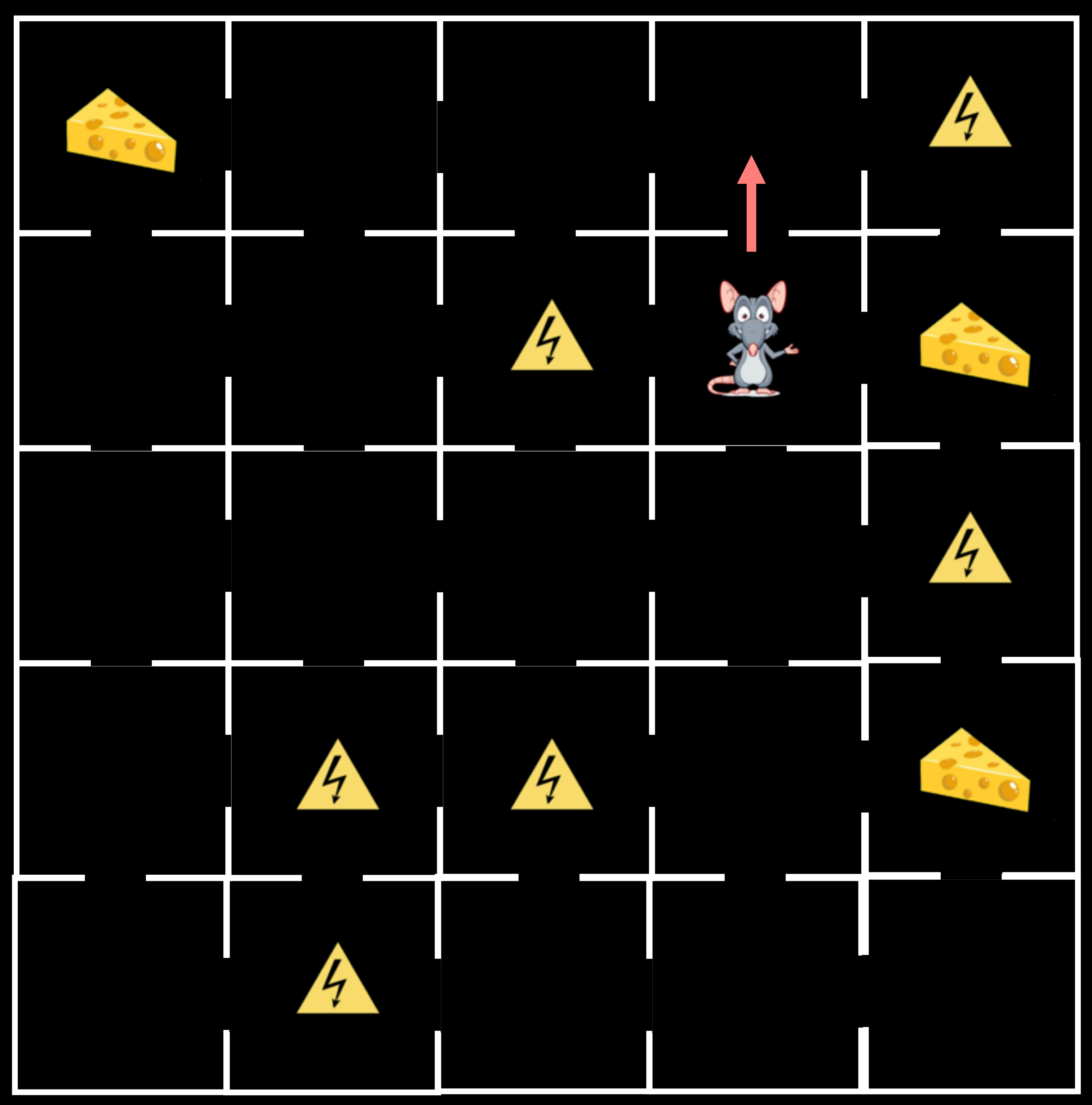


Based on the action, taken Mouse A still has possibility of taking 2 cheese slices, whereas Mouse B can only take one.

Mouse A goes left



Mouse B goes up



Based on the action, taken Mouse A still has possibility of taking 2 cheese slices, whereas Mouse B can only take one.

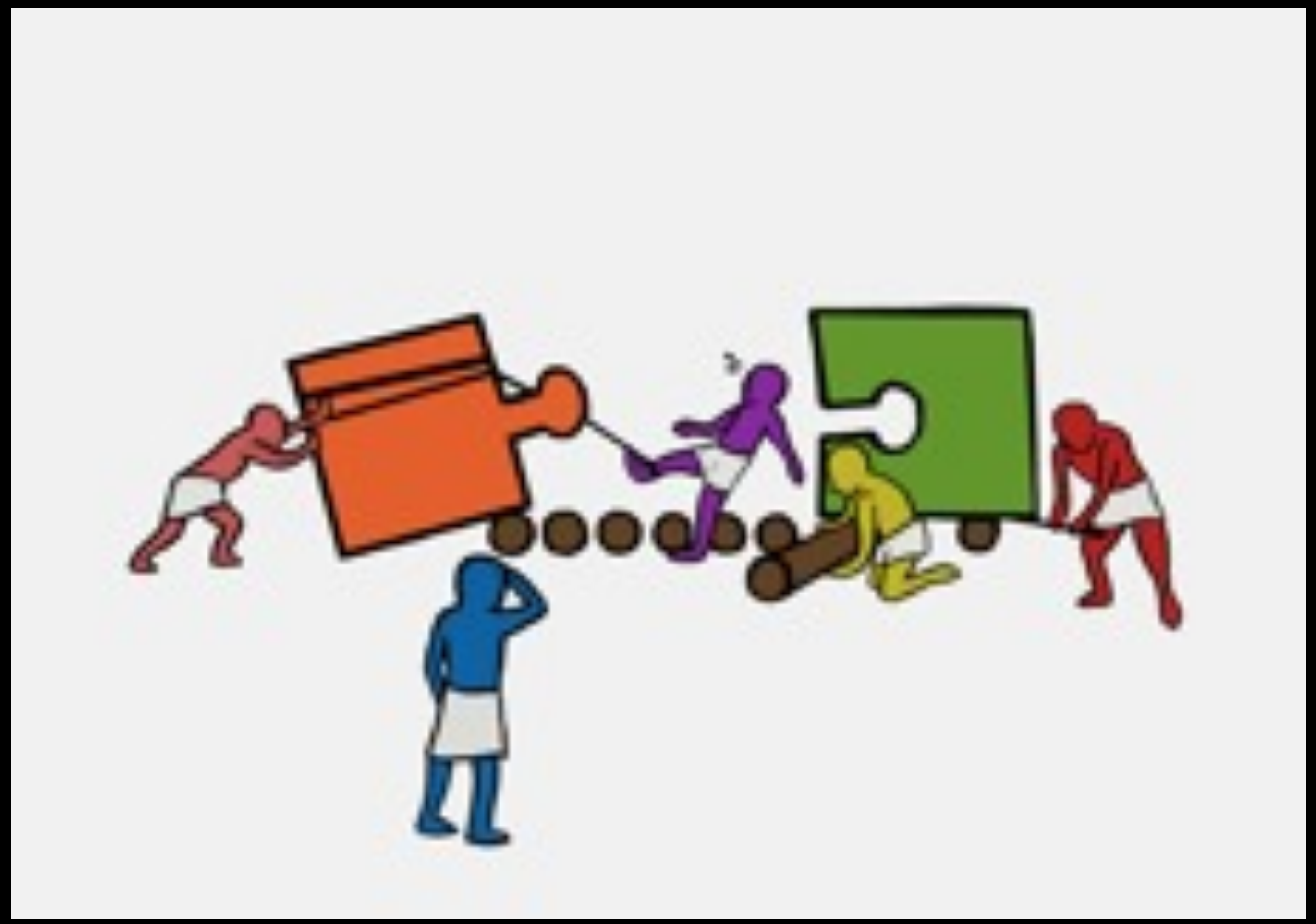
Thus, for the same state, on selecting a different action, the reward an agent gets changes. This is called Action Value function.

Value of taking action *a* state *s* under a policy π is given by $q_\pi(s, a)$. It is the expected return starting from *s*, taking the action *a*, thereafter following policy π.

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$$

q_π is called the Q-function.

Exercise: Setting up a Custom Environment



The aim of this exercise is to learn how to set up a custom environment using OpenAI Gym. For setting up any custom environment, we will have to define, the state, possible actions and the reward obtained for a particular action in a given state.

For our custom environment, we will implement the mouse grid present in the slides. The possible rewards and state given the current state are in the helper file.