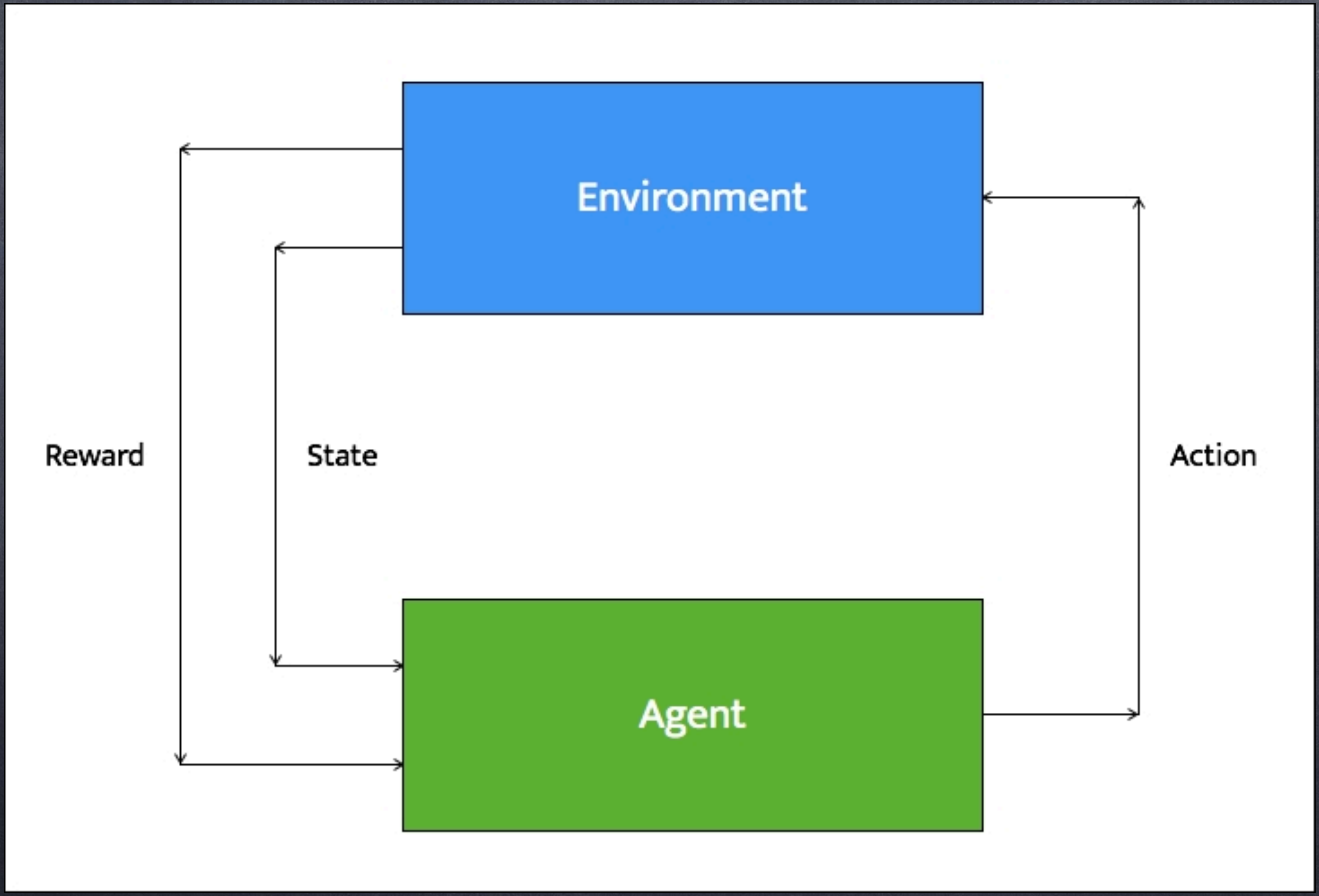
Reinforcement Learning 2

Typical RL Environment



Typical RL Environment

- There is a set of states related to the agent and the environment. At a given point of time, the agent observes an input state to sense the environment.
- There are policies that govern what action needs to be taken. These policies act as decision making functions. The action is determined based on the input state using these policies.
- The agent takes the action based on the previous step.
- The environment reacts in a particular way in response to that action. The agent receives reinforcement, also known as reward, from the environment.
- The agent records the information about this reward. It's important to note that this reward is for this particular pair of state and action.

Steps in RL

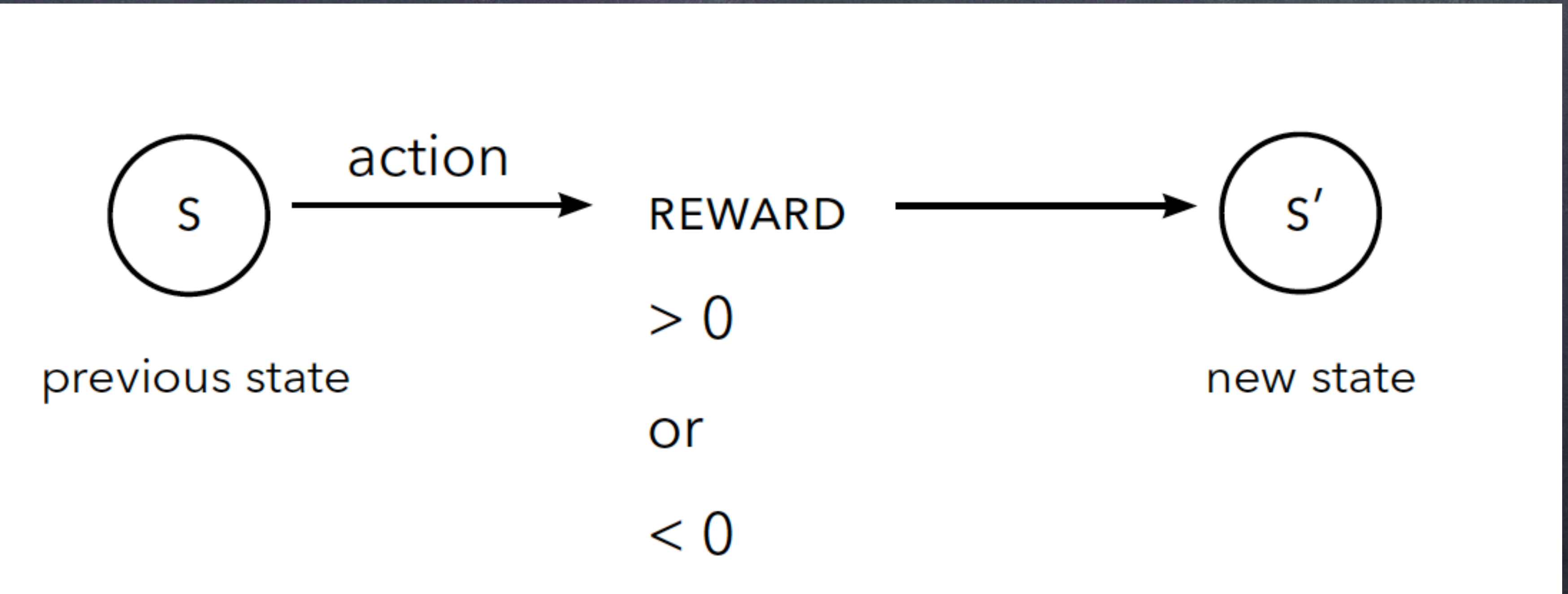
- Creating or modeling an environment
- Defining agent, states, and action
- Defining learning algorithm (e.g. Q Learning)

Agent

- The agent performs the reinforcement learning task
  - It has explicit goals (problem for music...)
  - It can sense aspect of the environment (environment described in terms of states)
  - It performs actions to influence the environment

Reward Function

- It defines the goal in a reinforcement learning problem
- It gives the agent a sense of what is good in an immediate sense (pleasure / pain)



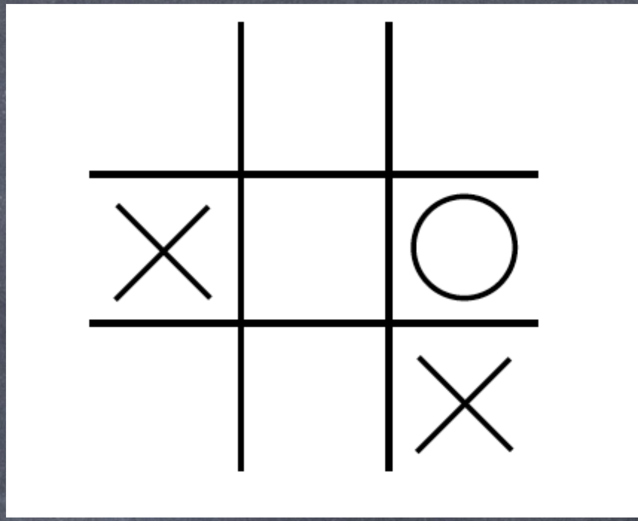
Value Function

- It gives the agent a sense of what is good in the long run
- The value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state
- It is either:
  - A function of the environment's states (state value function)
  - A function of the environment's states and of the agent's actions (action value function)

Model of Environment

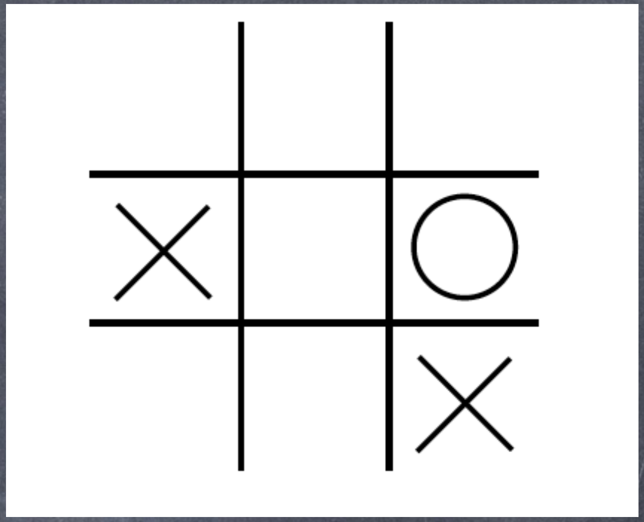
- It is used to predict the states the environment will be in after the agent performs its actions
- In reinforcement learning, the agent often uses the model to compute series of potential state-action sequences: it projects himself in the future to decide which action to perform in the present

Example: Tic-Tac-Toe Game



- Approach?
- Reward?
- value function?

Example: Tic-Tac-Toe Game

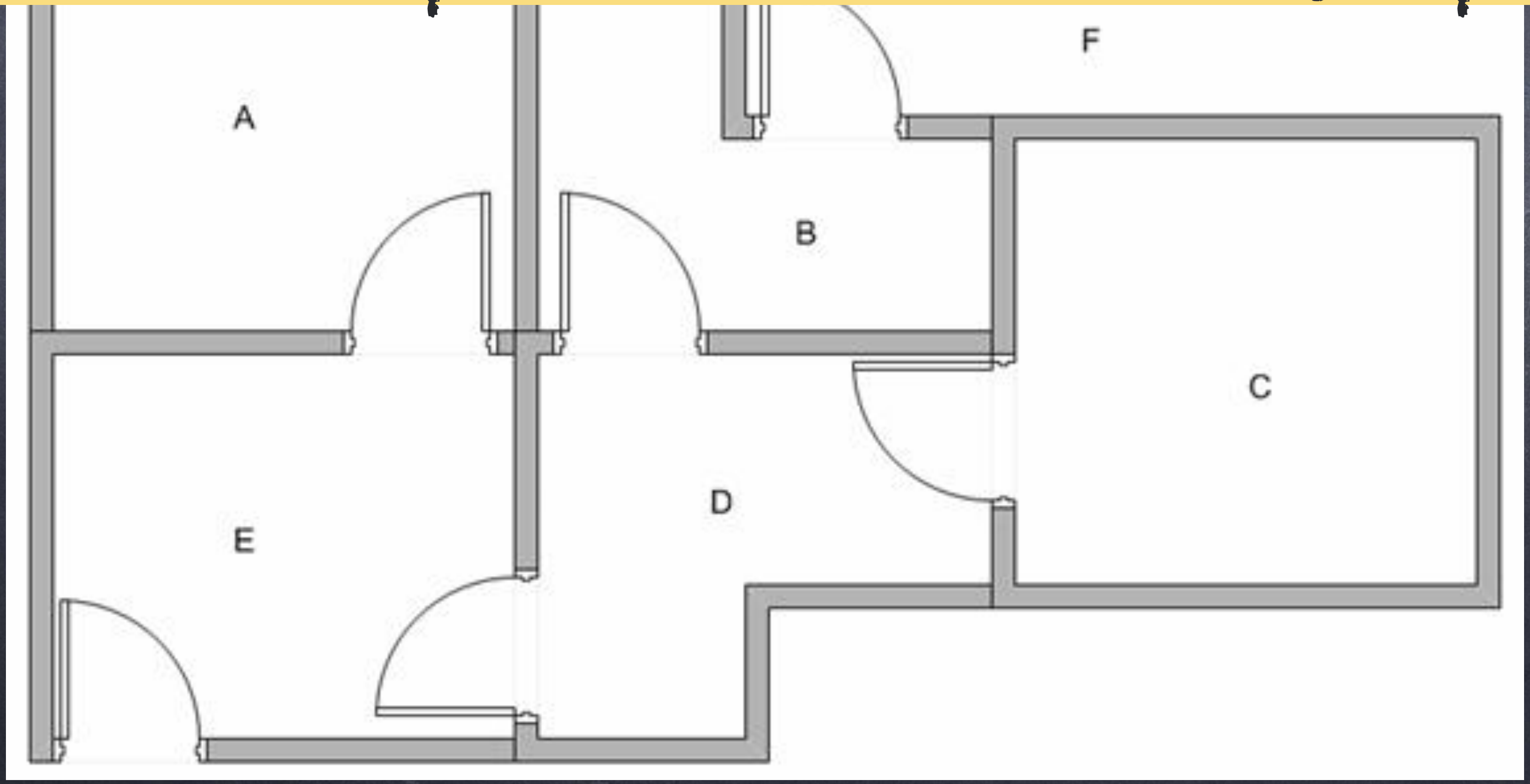


- Approach?
- Reward?
- value function?
- Approach: Reinforcement learning with approximate value functions.
- Reward: +1 for winning the game.
- Value Function: A table storing the last estimated probability of our winning from each state of the game (init at 0.5).

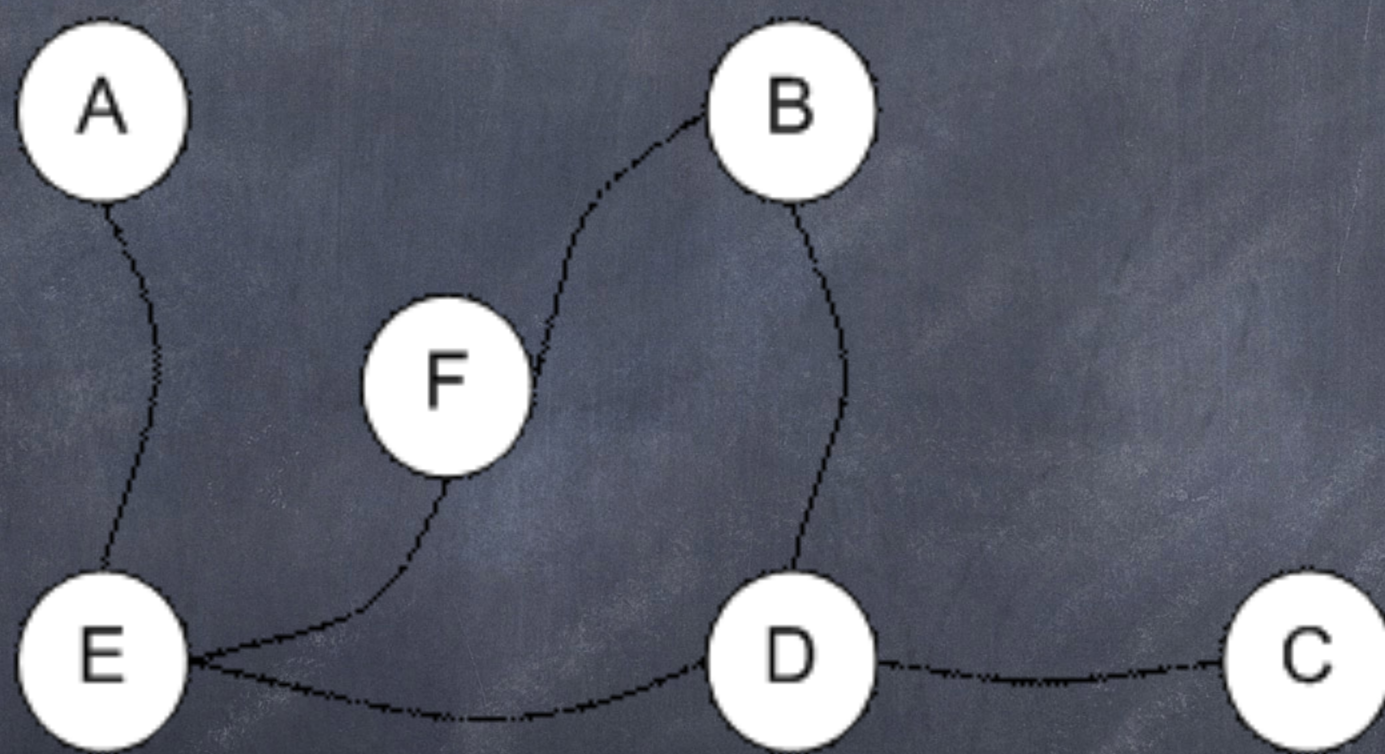
Modeling the Environment

- 5 room example

Can we represent rooms as graph?



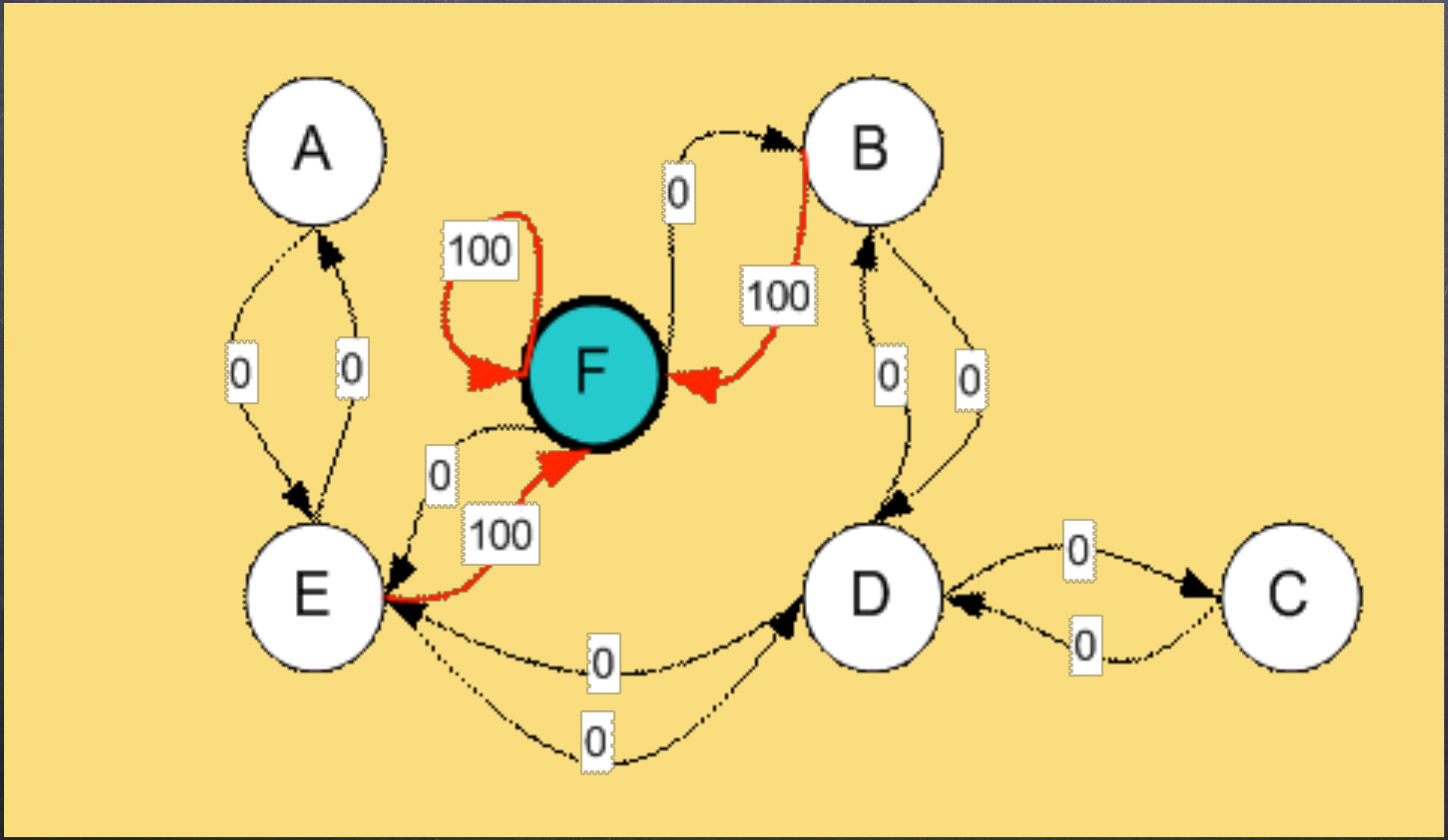
Can we represent rooms as graph?



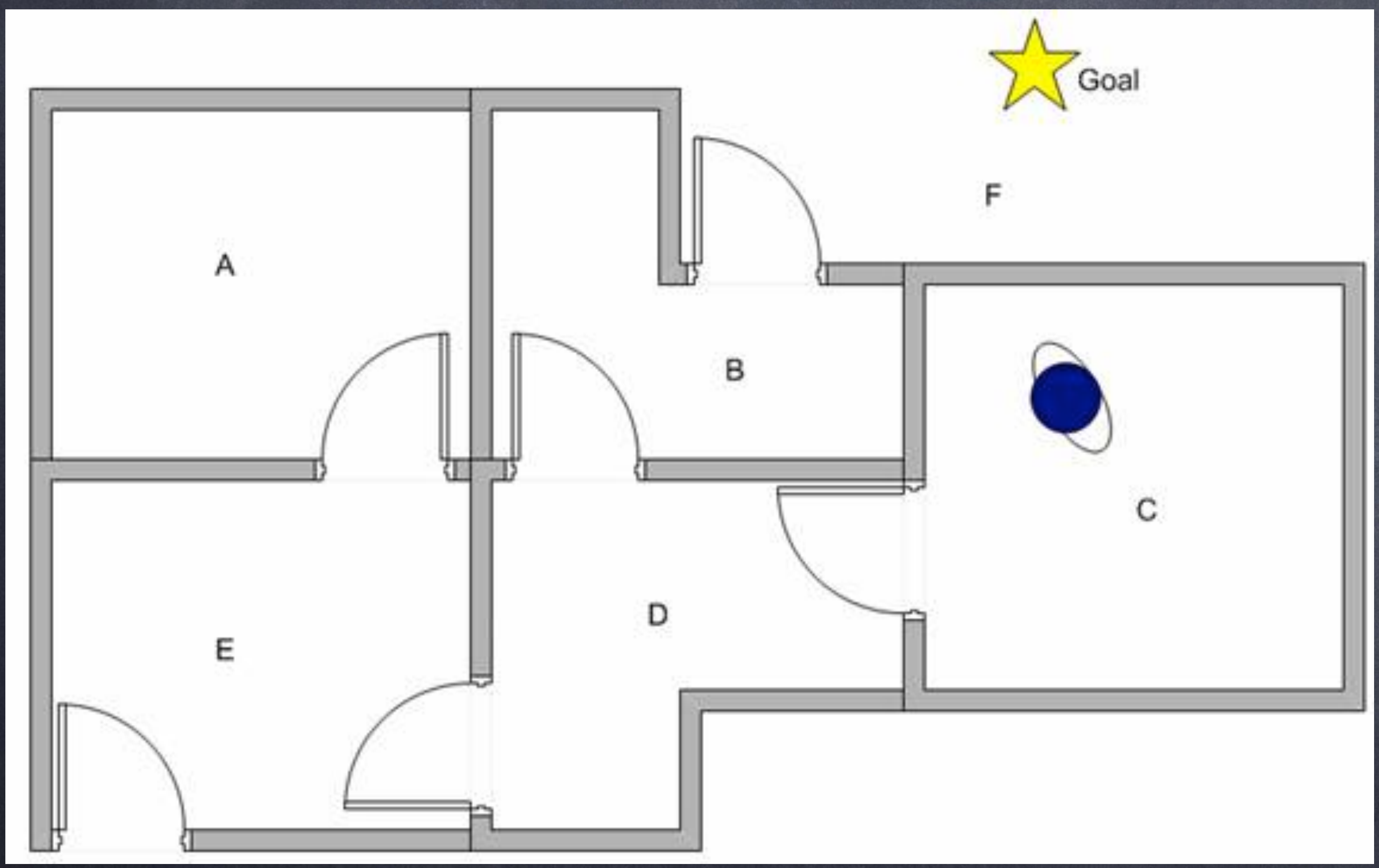
Target and Reward

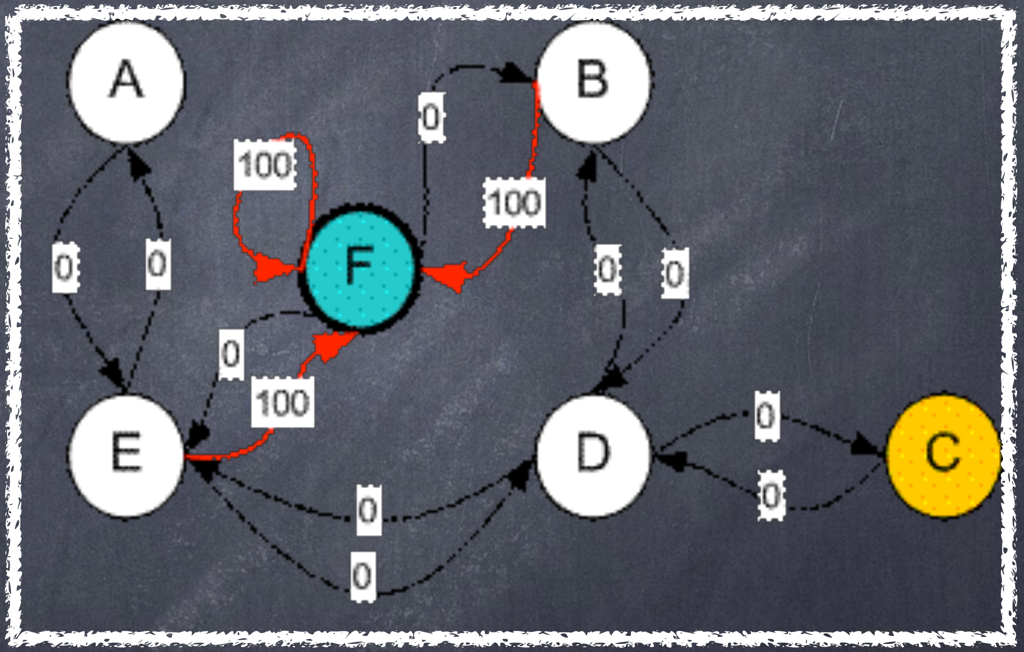
- We want to go out of the building
- This means, F is our target node
- Give "reward value" to each door
- Nodes that are "connected" to "F" will have reward value of 100 and others have reward value as 0 (zero)

Reward



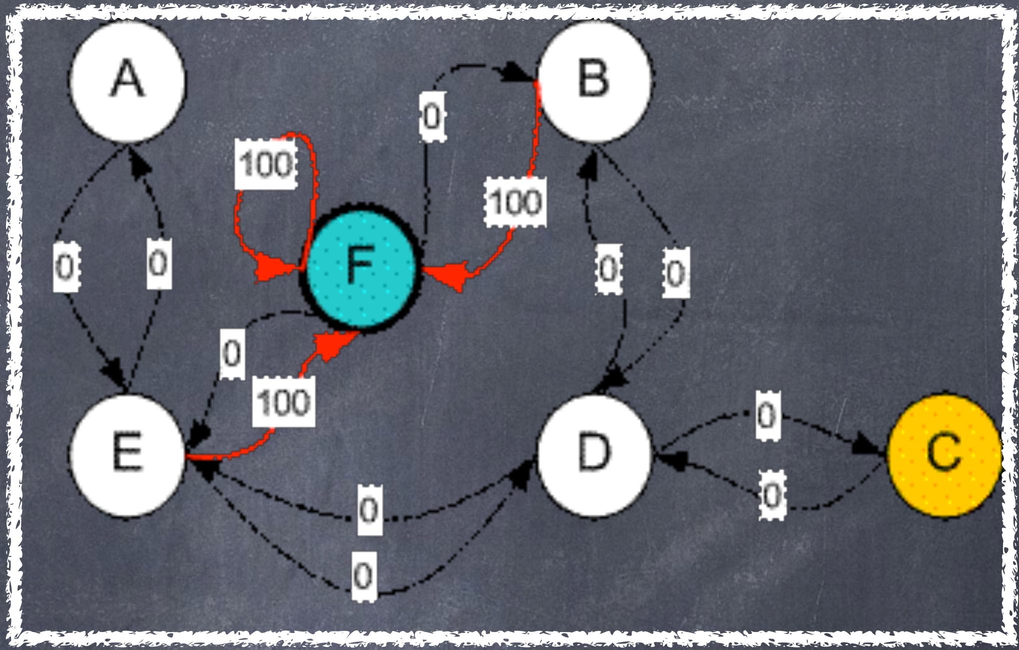
Agent





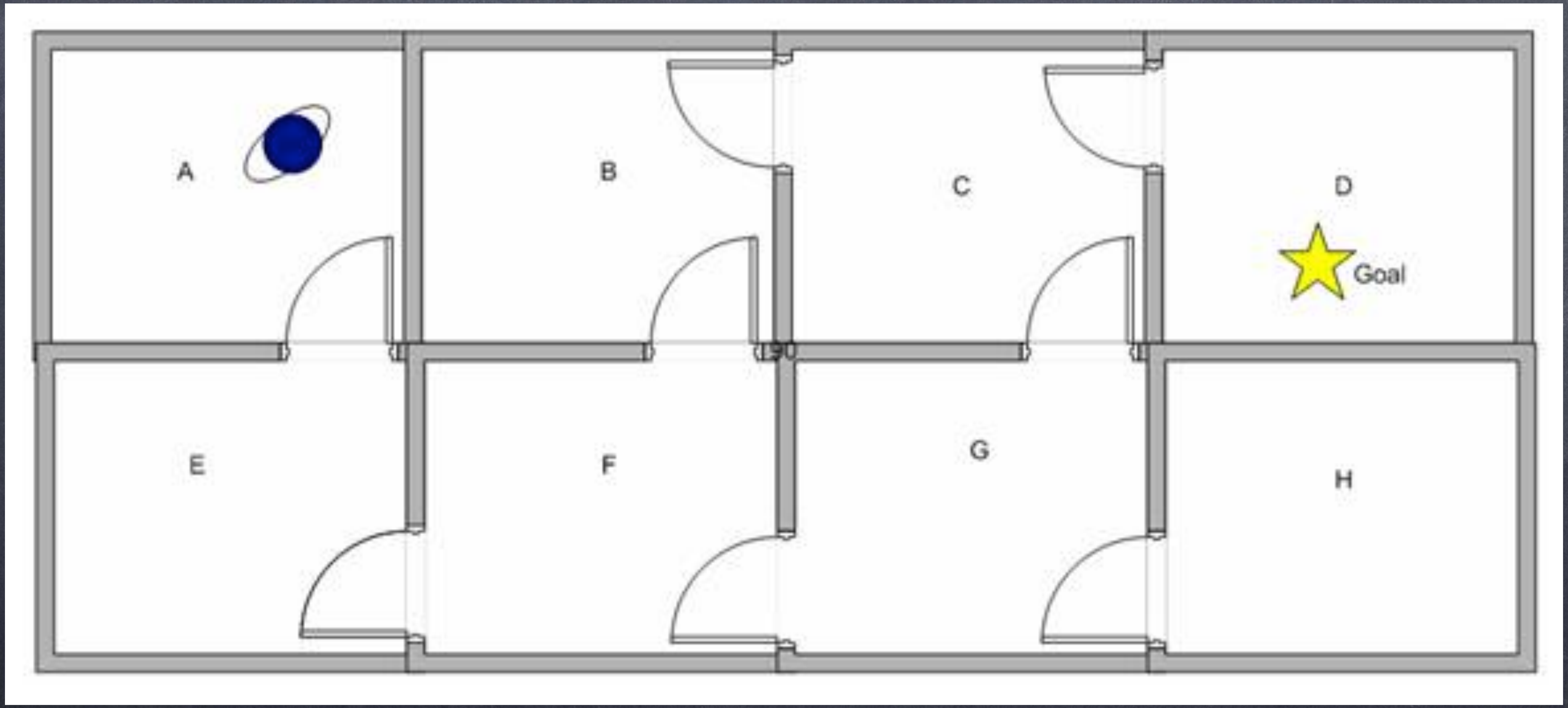
Reward Table

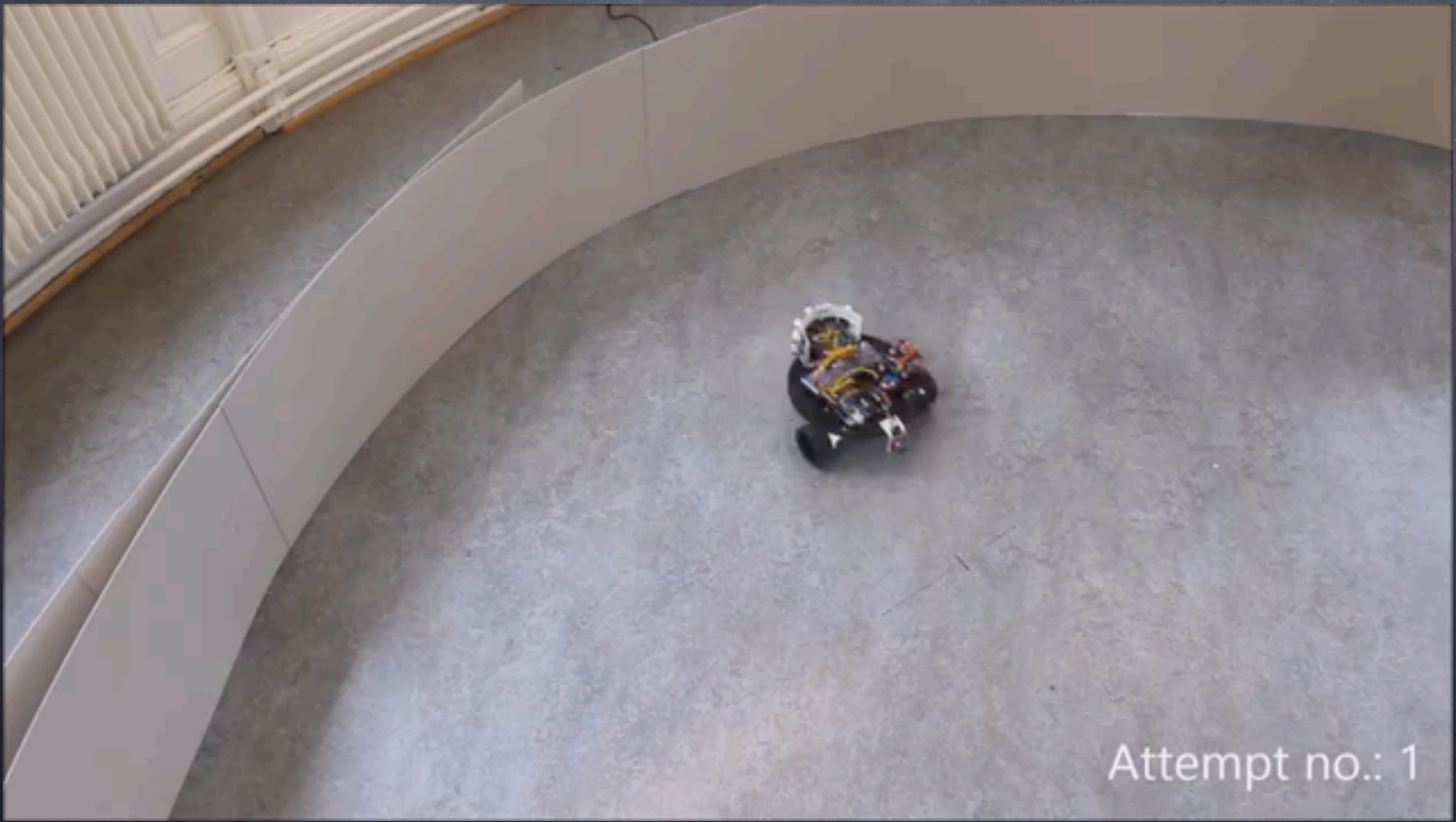
- Using the state diagram, we can create reward table



	Action to go to state					
Agent now in state	A	B	C	D	E	F
A	-	-	-	-	0	-
B	-	-	-	0	-	100
C	-	-	-	0	-	-
D	-	0	0	-	0	-
E	0	-	-	0	-	100
F	-	0	-	-	0	100

Let us try one more problem




Attempt no.: 1

What the last video teaches us about RL?

- Exploration and Exploitation
- Reward (positive and negative) - hitting the wall/obstacle
- Learning how to move (end of the RL training, we have the state space in which the bot know how to avoid obstacle)

Q Learning Simplified

- Given: state diagram (R matrix)
- Set parameters and environment
- Initialize Q matrix as zeros
- For each iteration
  - Select random initial state
  - Do while not reach goal state
    - Select one among all possible states
    - Using possible action, go to next state
    - Get maximum Q value based on R value and update Q-table
    - Set the next state as current state

Q Learning

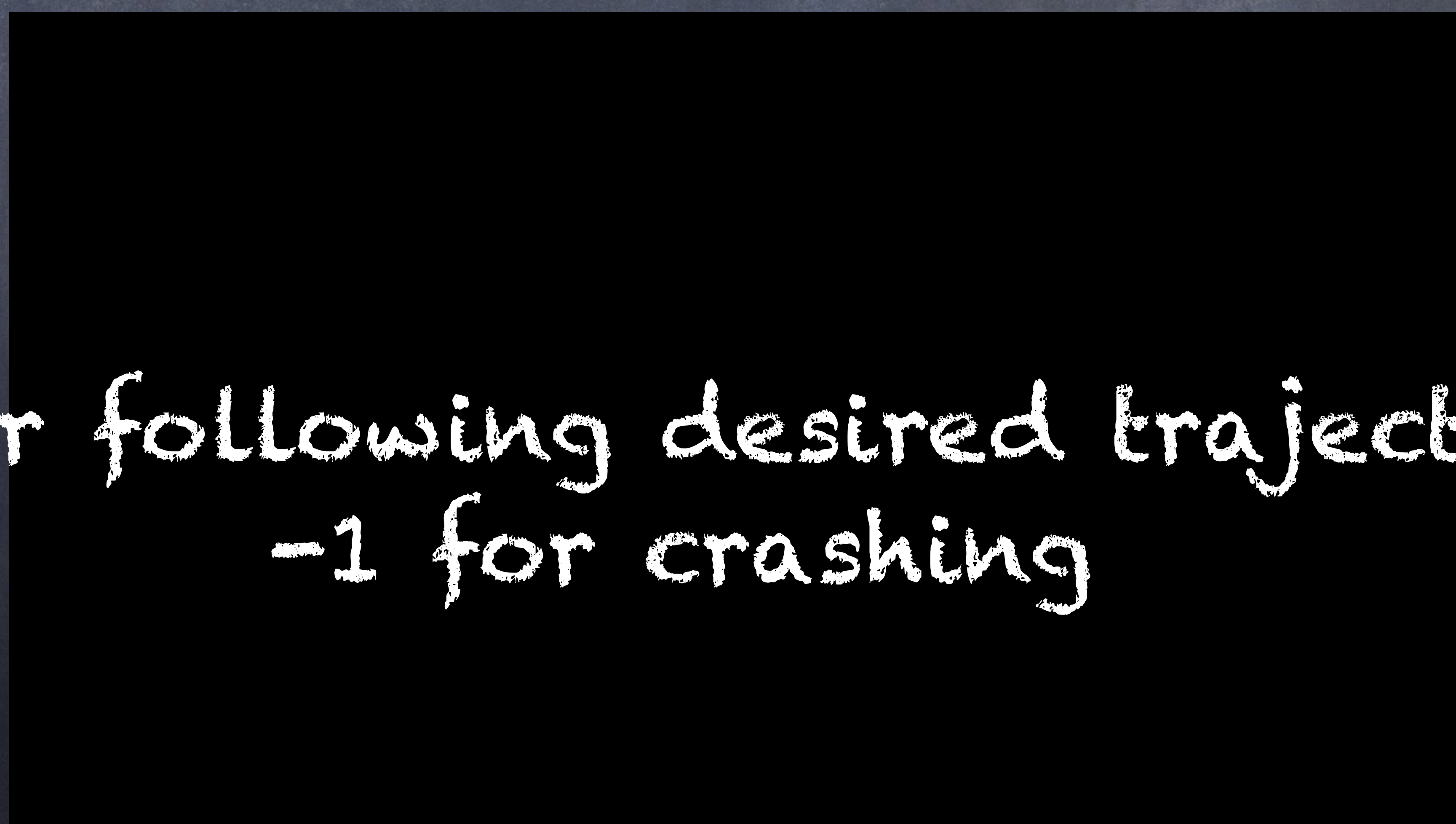
- Do
  - Select an action a and execute it
  - Receive immediate reward r
  - Observe new state s.
  - Update each table entry for ^Q(s, a) as follows

$$\hat{Q}(s,a) \leftarrow r + \gamma max_{a'} \hat{Q}(s', a')$$

  - Change state: $s \leftarrow s'$

Recall Reward

- Reward R is a scalar feedback signal
- How well the agent is doing at step t
- What would be reward for the given application (video)?



+1 for following desired trajectory

-1 for crashing

History and State

- History is what the agent has seen so far (action, observation, reward - observable variables)
- Ht=A1,O1,R1, ... At,Ot,Rt

History and State

- State is the information used to determine what happens next
- What happens next is dependent on the history
  - Agent selects actions
  - Environment selects observation/rewards

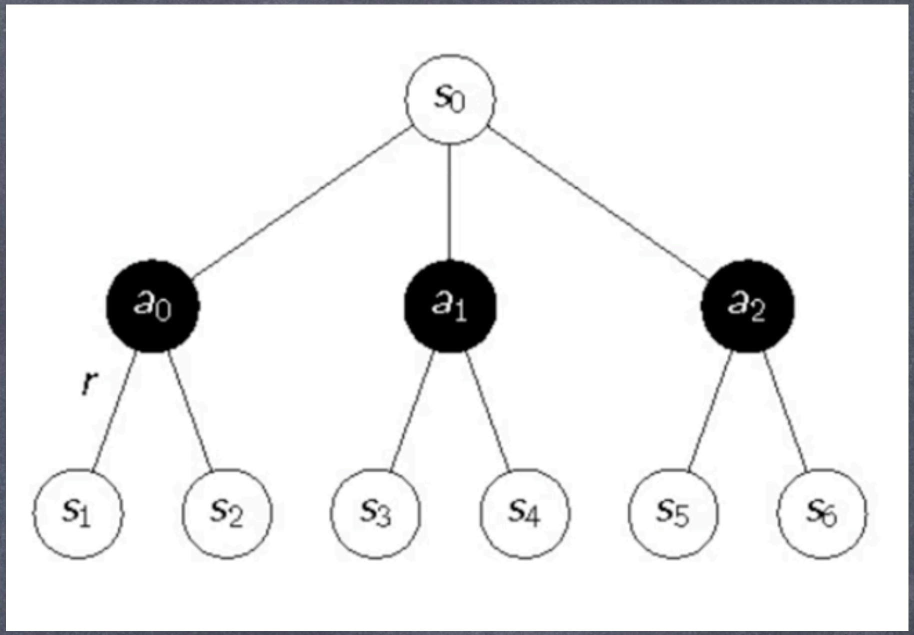
State

- St=f(Ht)
- In atari, look at last 4 "history"
- Environment state
- Agent state

Information State

- Or Markov state
  - St is Markov if and only if
  - P[S(t+1)|S(t)]=P[S(t+1)|S(1),...,S(t)]
  - Future is independent of the past given the present
  - H(1:t)->S(t)->H(t+1:\infinite)

Another Nomenclature



- state
- action
- reward
- Policy : agent's behavior function
- Value function : how good is each state and/or action
- Model : agent's representation of the environment

Policy

- A policy is the agent's behaviour
- It is a map from state to action:
- Deterministic policy: $a = \pi(s)$
- Stochastic policy: $\pi(a|s) = \mathbb{P}[a|s]$

Stochastic vs Deterministic Policy

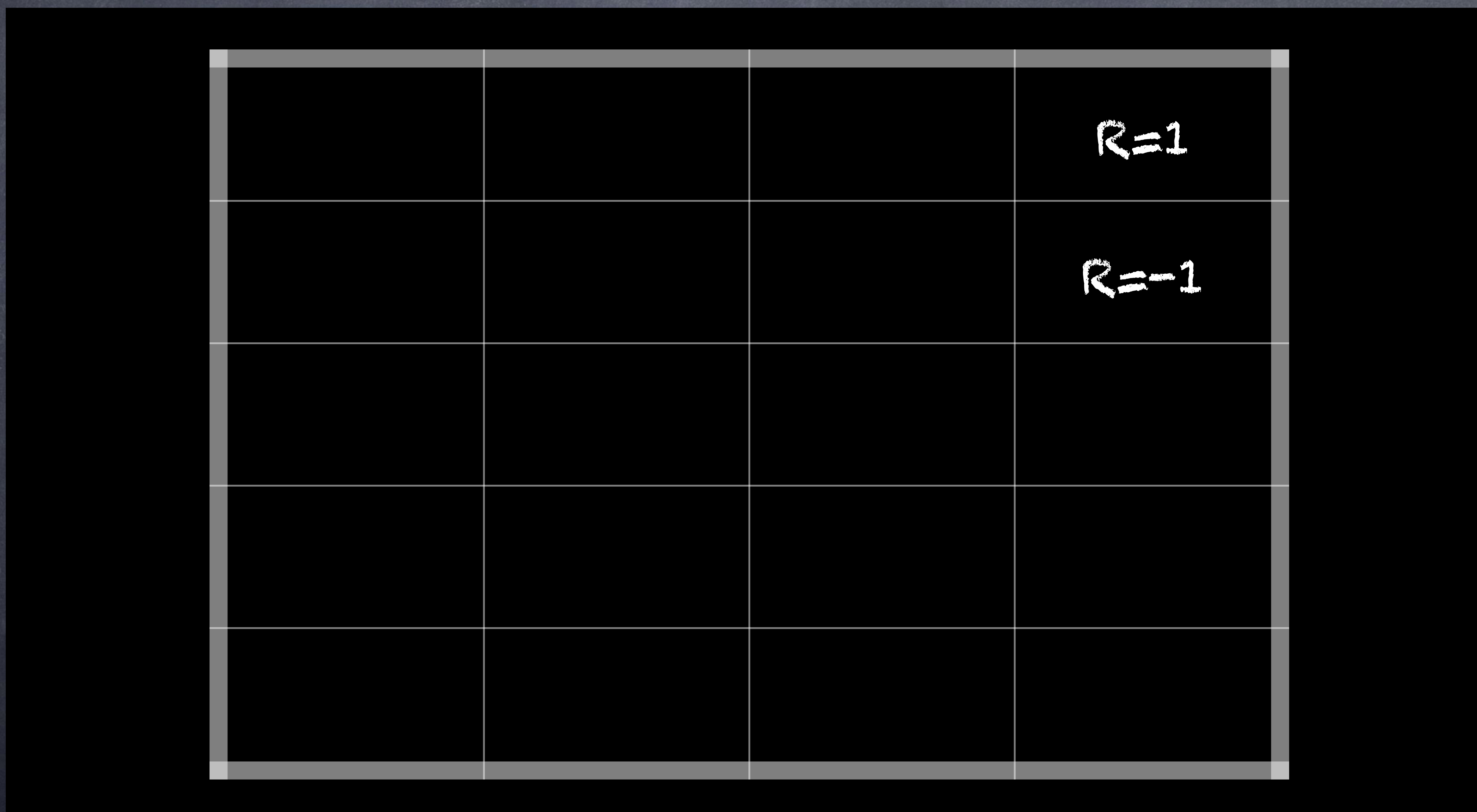
- stochastic policy models a distribution over actions, and draws action according to this distribution.
- A deterministic policy always returns the same action with the highest expected Q value.

Value Function

- A value function is a prediction of future reward
  - "How much reward will I get from action a in state s ?"
- Q-value function gives expected total reward from state s and action a under policy \pi with discount factor

$$Q^\pi(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots \mid s, a\right]$$

Bellman Equation



		V=1	1
			-1

V=1	V=1	V=1	1
V=1			-1
V=1			

Agent	V=1	V=1	1
V=1			-1
V=1			

Bellman Equation

- V(s) = max{R(s,a)+\gamma*V(s')}

Bellman Equation

- V(s) = max{R(s,a)+\gamma*V(s')}

Bellman Equation

- V(s) = max{R(s,a)+\gamma*V(s')}

Bellman Equation

$\gamma=0.9$

- $V(s) = \max\{R(s,a)+\gamma * V(s')\}$

Bellman Equation

$\gamma=0.9$

- $V(s) = \max\{R(s,a)+\gamma*V(s')\}$

Bellman Equation

\gamma=0.9

V=0.81	V=0.9	V=1	R=1
0.73			R=-1

- $V(s) = \max\{R(s,a)+\gamma * V(s')\}$

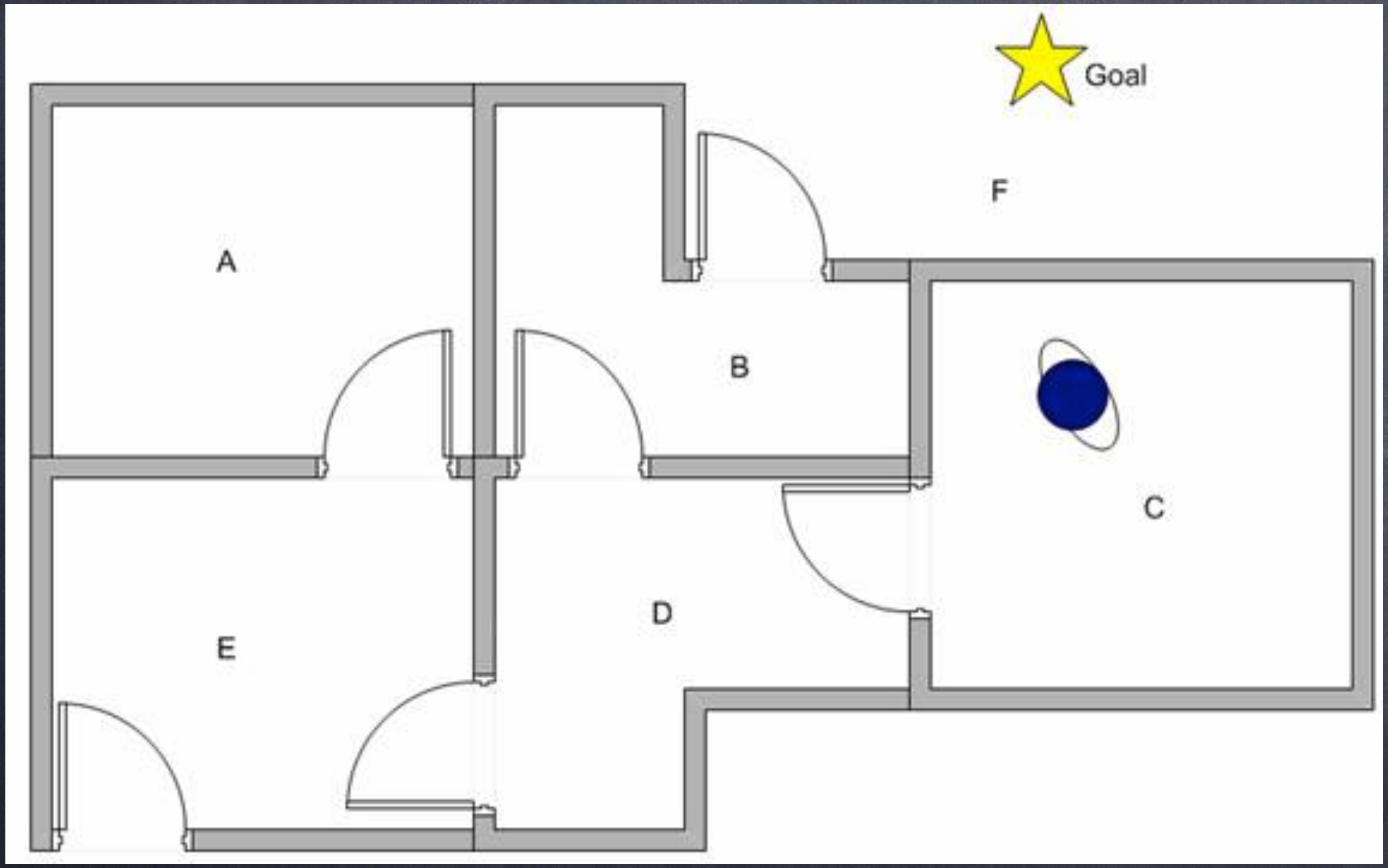
Bellman Equation

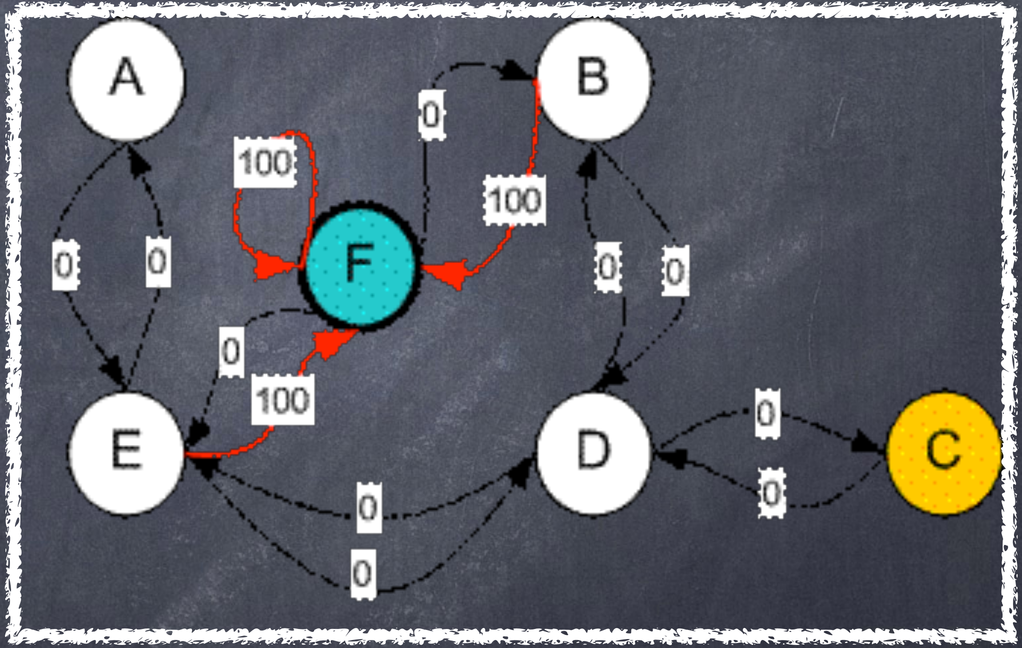
\gamma=0.9

V=0.81	V=0.9	V=1	R=1
0.73		?	R=-1
		?	

- $V(s) = \max\{R(s,a)+\gamma * V(s')\}$

Agent





RL Model

- Value Function: $V^{\pi}(s) = E[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi]$

- Bellman form:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s') V^{\pi}(s')$$

$$Q^{\pi}(s, a) = R(s) + \gamma \sum_{s' \in S} T(s') max_a Q^{\pi}(s', a')$$

- T-> the probability of transition from s to s' given action a

Bellman Equation

$$Q^{\pi}(s,a) = \mathbb{E}\left[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots \mid s, a\right]$$

$$\downarrow$$

$$Q^{\pi}(s,a) = \mathbb{E}_{s',a'}\left[r + \gamma Q^{\pi}(s',a') \mid s, a\right]$$

Bellman Equation

- An optimal value function is the maximum achievable value

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

- Once we have Q^* we can act optimally,

$$\pi^*(s) = \operatorname*{argmax}_a Q^*(s,a)$$

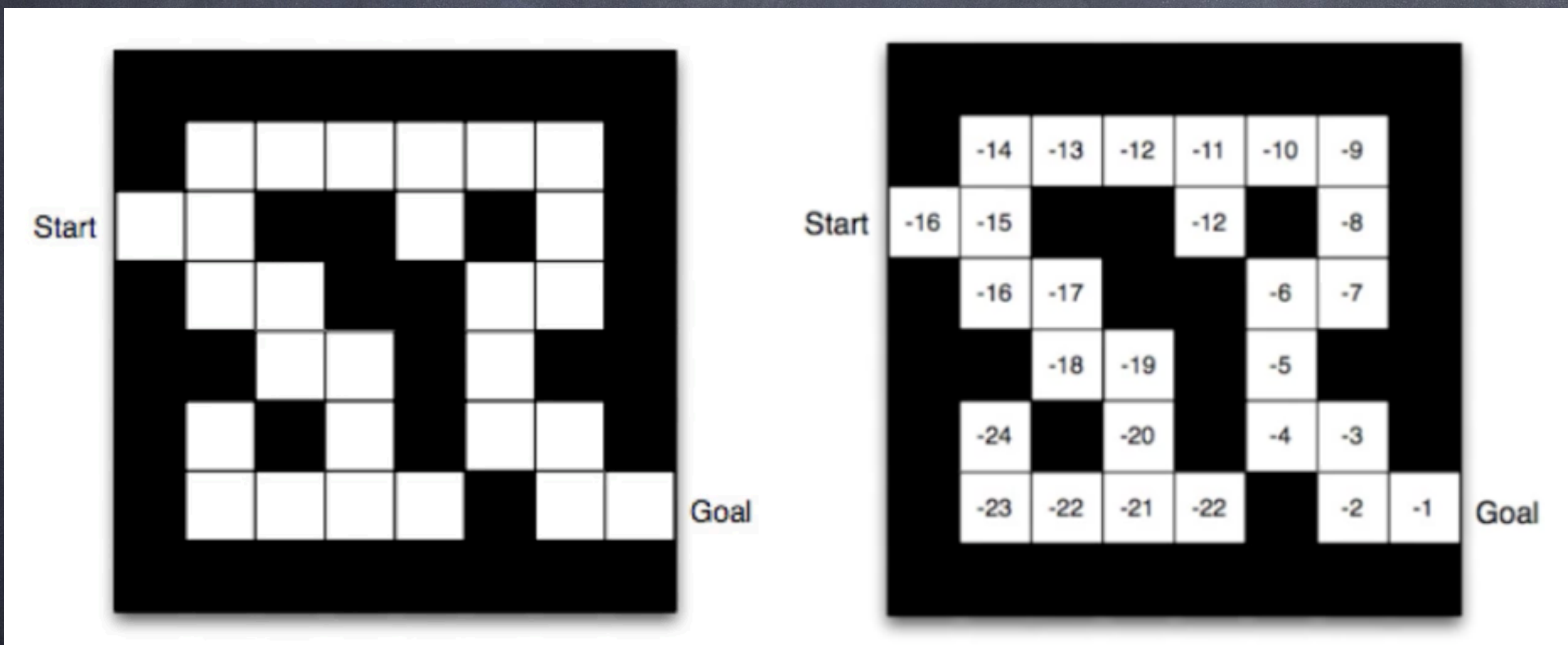
- Optimal value maximises over all decisions. Informally:

$$\begin{aligned} Q^*(s,a) &= r_{t+1} + \gamma \max_{a_{t+1}} r_{t+2} + \gamma^2 \max_{a_{t+2}} r_{t+3} + \dots \\ &= r_{t+1} + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \end{aligned}$$

- Formally, optimal values decompose into a Bellman equation

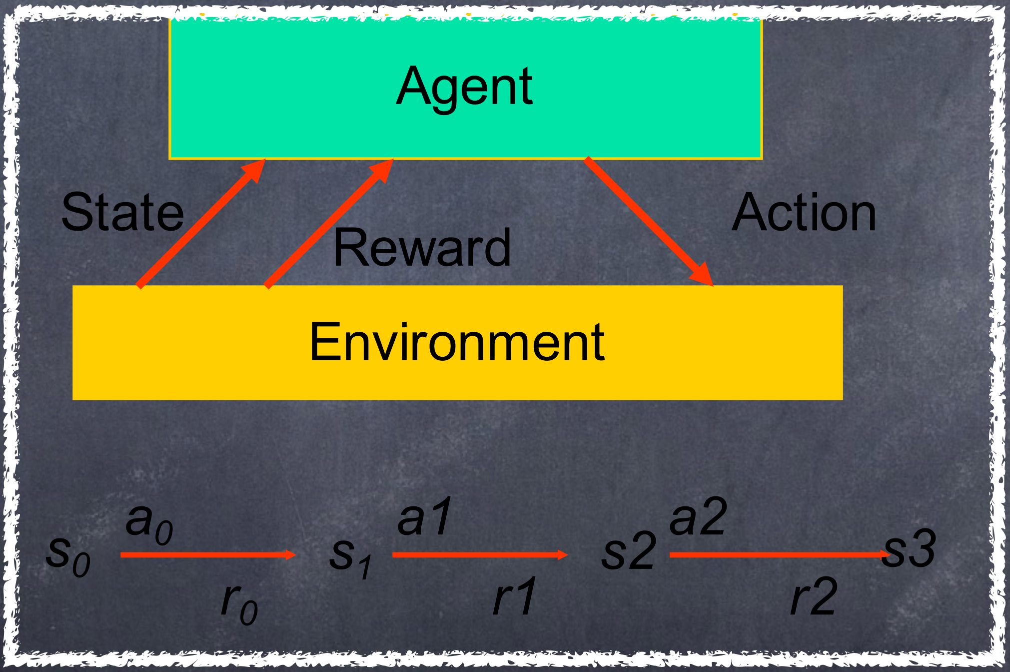
$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a \right]$$

Quick Example



Revisiting Q-Learning

- MDP (S,T,A,R)



- S-> states, A-> actions, R-> the expected reward for taking action a in state s
- T-> the probability of transition from s to s' given action a

Q Learning

- Do
  - Select an action a and execute it
  - Receive immediate reward r
  - Observe new state s.
  - Update each table entry for ^Q(s, a) as follows

$$\hat{Q}(s,a) \leftarrow r + \gamma max_{a'} \hat{Q}(s',a')$$

  - Change state: $s \leftarrow s'$

Q Learning Simplified

- Give: state diagram (R matrix)
- Set parameters and environment
- Initialize Q matrix as zeros
- For each iteration
  - Select random initial state
  - Do while not reach goal state
    - Select one among all possible states
    - Using possible action, go to next state
    - Get maximum Q value based on R value and update Q-table
    - Set the next state as current state