Outline of This Course

- RL1: Introduction to Reinforcement Learning
- RL2: Reinforcement Learning for Lightweight Model
  - Applications
  - Fundamentals of RL
- RL3: Value Based Reinforcement Learning
  - Fundamentals of Value Based RL
  - Algorithms
- RL4: Policy-based Reinforcement Learning
  - Fundamentals of Policy Based RL
  - Algorithms

Introduction to Reinforcement Learning

I-Chen Wu

- Sutton, R.S. and Barto, A.G., Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, 1998.
  - http://webdocs.cs.ualberta.ca/~sutton/book/ebook/the-book.html
  - Bible in this area.
- David Silver, Online Course for Deep Reinforcement Learning.
  - http://www.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html

"Introduce deep reinforcement learning"

Chat-GPT:

"Deep reinforcement learning is a type of machine learning algorithm that combines the principles of reinforcement learning with the power of deep learning neural networks. In reinforcement learning, an agent learns by interacting with its environment and receiving rewards or punishments for certain actions. By incorporating deep learning, the agent is able to process and analyze large amounts of data to make more informed decisions and improve its performance over time. This allows the agent to tackle complex and dynamic environments, such as video games or robotic control, and make optimal decisions based on the current state and available actions. Deep reinforcement learning has been used to achieve state-of-the-art results in a variety of fields, including natural language processing and autonomous driving."

"Talk about deep reinforcement learning"

Chat-GPT:

"Deep reinforcement learning is a subfield of machine learning that involves training agents to make decisions in complex, uncertain environments. The goal of deep reinforcement learning is to build algorithms that can automatically learn to make good decisions based on sensory input, such as video or audio data.

In deep reinforcement learning, an agent interacts with its environment by taking actions and receiving rewards or penalties in return. The agent's goal is to maximize the cumulative reward it receives over time. To do this, the agent must learn to map its sensory input to appropriate actions.

One of the key challenges in deep reinforcement learning is that the agent must learn to make decisions based on raw sensory data, which can be noisy and unstructured. To overcome this challenge, deep reinforcement learning algorithms typically use deep neural networks to process and interpret sensory data.

Deep reinforcement learning has been used to solve a wide range of problems, including playing video games, controlling robots, and optimizing supply chain logistics. It has also been applied to more complex tasks, such as natural language processing and drug discovery.

Overall, deep reinforcement learning is a powerful tool for building intelligent agents that can learn to make good decisions in complex environments.

"

David Silver:

(the leader of the AlphaGo team)

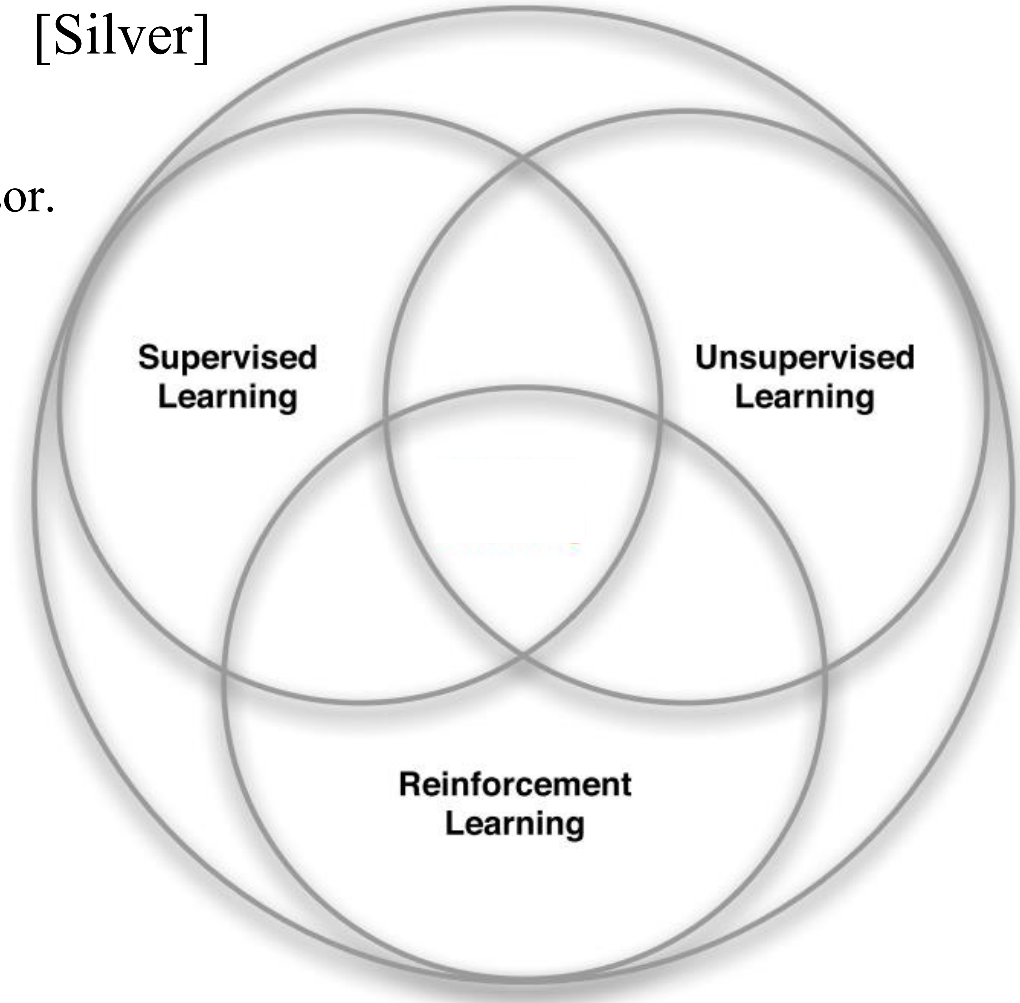
“DL+RL = AI”

Many Faces of Reinforcement Learning

- Computer Science
  - Machine Learning
- Engineering
  - Optimal Control
- Mathematics
  - Operations Research
- Economics
  - Bounded Rationality
- Psychology
  - Classical/Operant Conditioning
- Neuroscience
  - Reward System

Branches of Machine Learning

- Supervised Learning (SL) [Silver]
  - learning from a training set of labeled examples provided by a knowledgeable external supervisor.
- Unsupervised Learning (UL)
  - typically about finding structure hidden in collections of unlabeled data.
- Reinforcement Learning (RL)
  - learning from interaction



What are different from others?

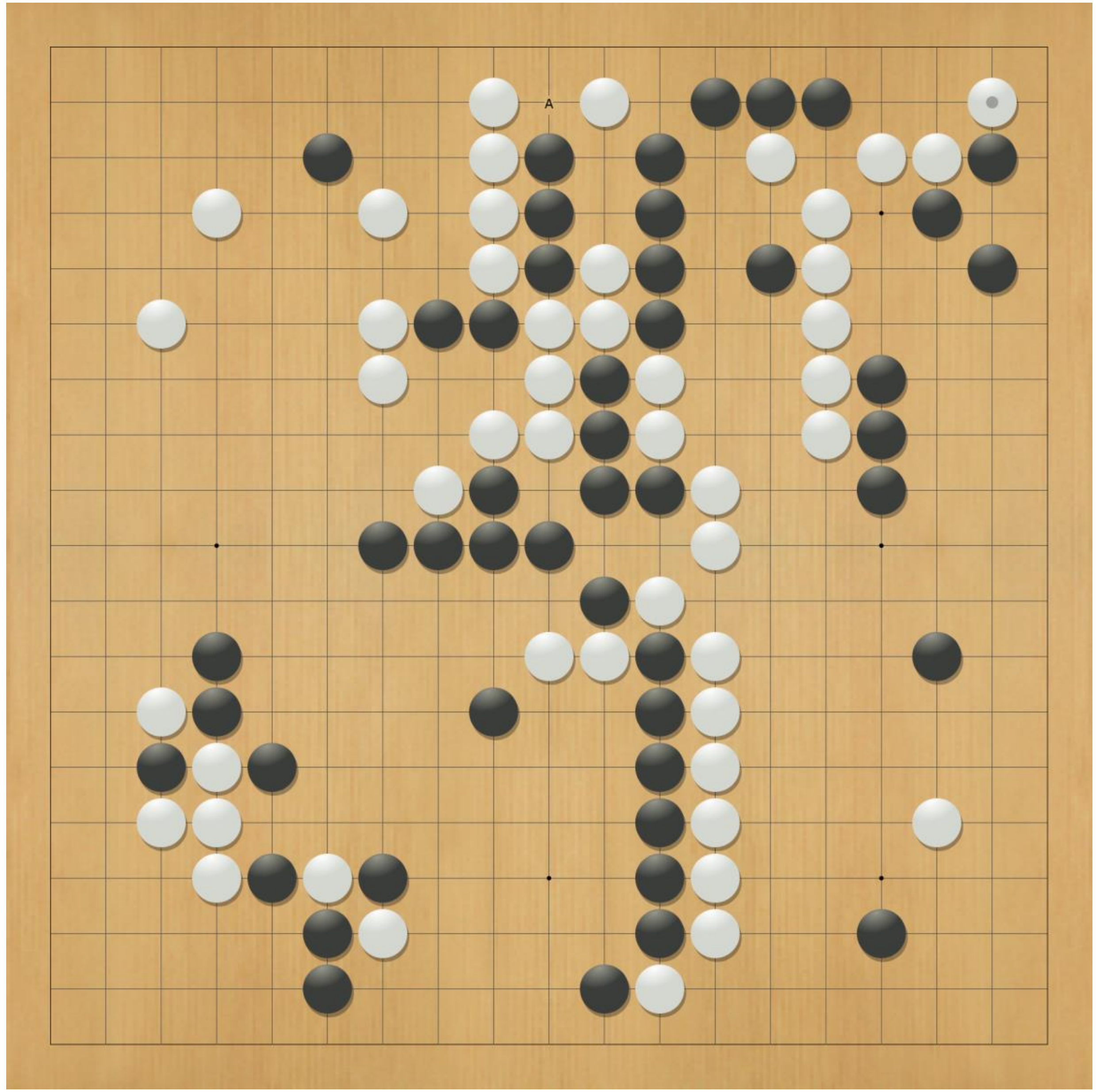
- Characteristics:
  - No supervisor, only a reward signal
  - Feedback is delayed, not instantaneous
  - Time really matters
  - Agent's actions affect the subsequent data and actions
- UL vs. RL:
  - RL is learning from interaction.
  - RL does not rely on examples of correct behavior.
  - RL is trying to maximize a reward signal,
    instead of trying to find hidden structure.

Successful Examples

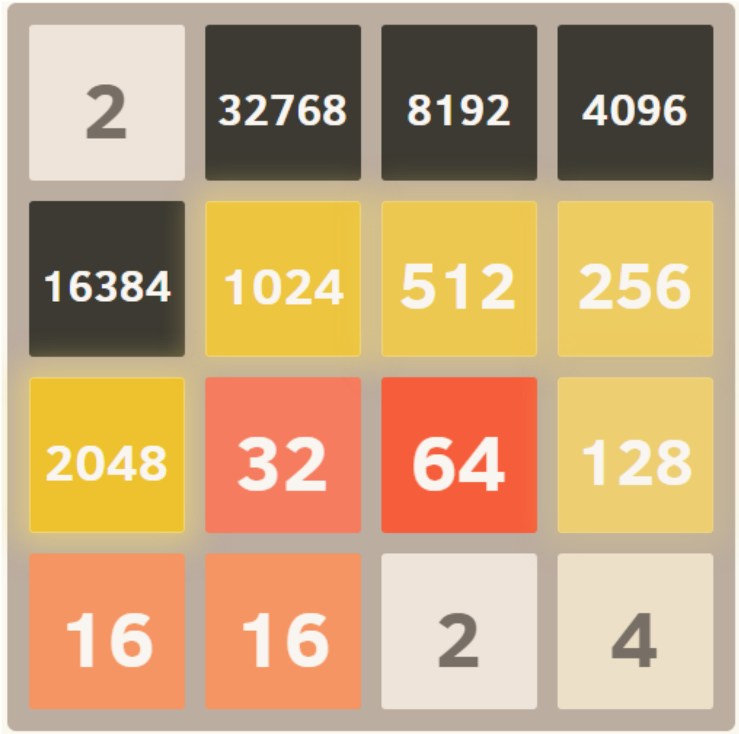
- Games: Super-human levels
  - Backgammon (Tesauro, 1994).
  - Connect6/2048/Threes! (CGI, 2022). Reach the top levels.
  - AlphaGo/AlphaZero/Muzero, using deep reinforcement learning (2016)
  - Open AI Five for Dota 2, 2019
  - AlphaStar for StarCraft by DeepMind (in nature), 2019
- Robotics: robot-controlled helicopters and humanoid robot walk (Abbeel et al.).
- Autonomous driving/racing: AWS DeepRacer (Amazon, CGI, 2019-)
- Chat bot: Chat-GPT (OpenAI, 2022)
- Optimizing matrix multiplication: AlphaTensor (2022)
- For manufacturing scheduling, a faster and smaller gap-to-optimal (by CGI, 2022).
- In chip design, a fast graph placement by Google Brain (Nature, 2021)
- …(More recent successful examples for deep reinforcement learning)

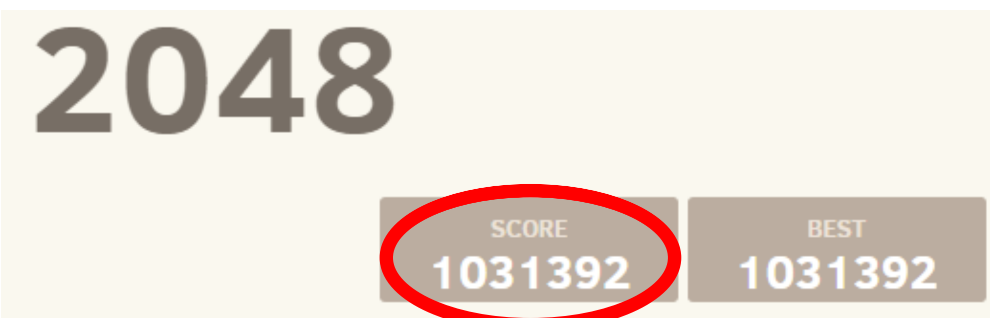
Board Game: Go

- Game 1: AlphaGo vs. 李世石



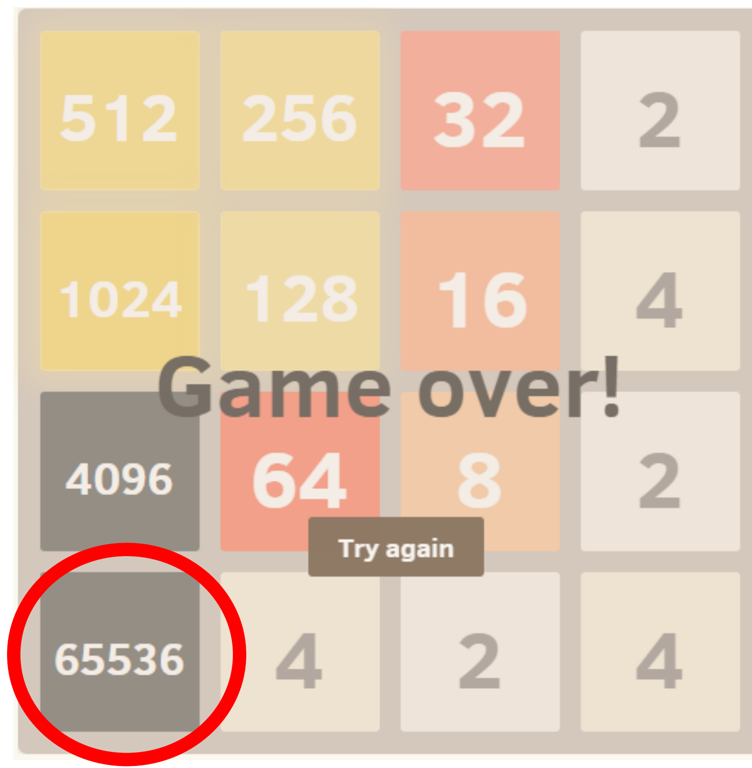
Stochastic Game: 2048 (by our lab)





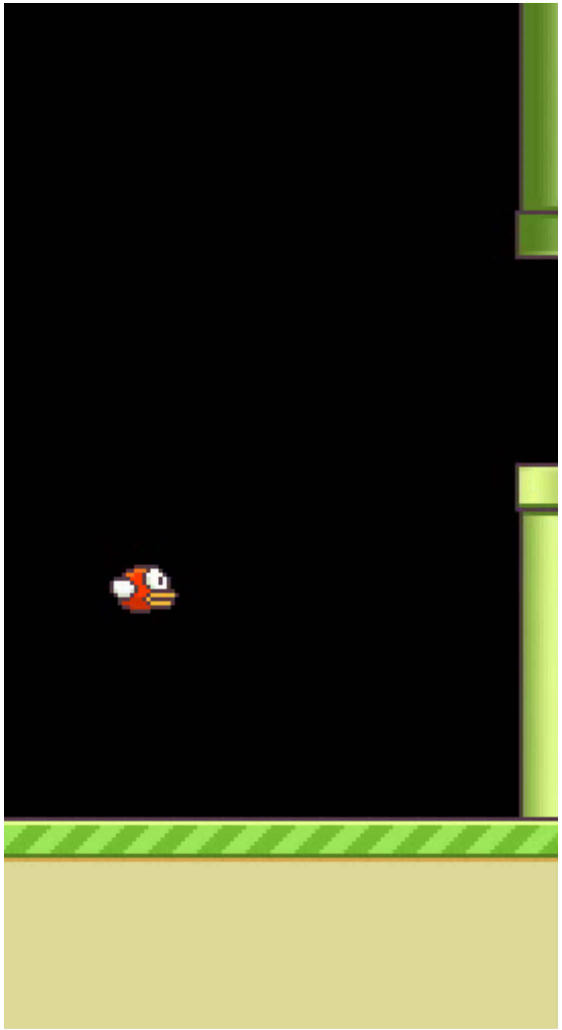
The First Game Reaching 65536 in the World (in 10,000 Trials)

http://2048.aigames.nctu.edu.tw/replay.php

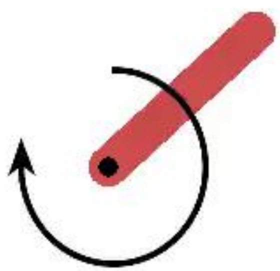


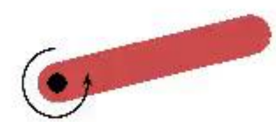
I-Chen Wu

Video Games: Flappy Bird (lab)



Open AI: Pendulum

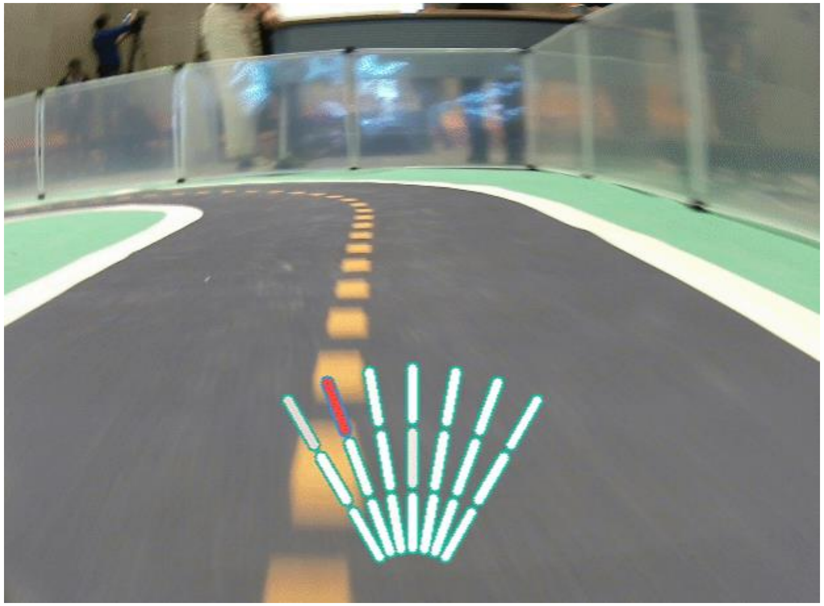




RL Demo (DDPG)

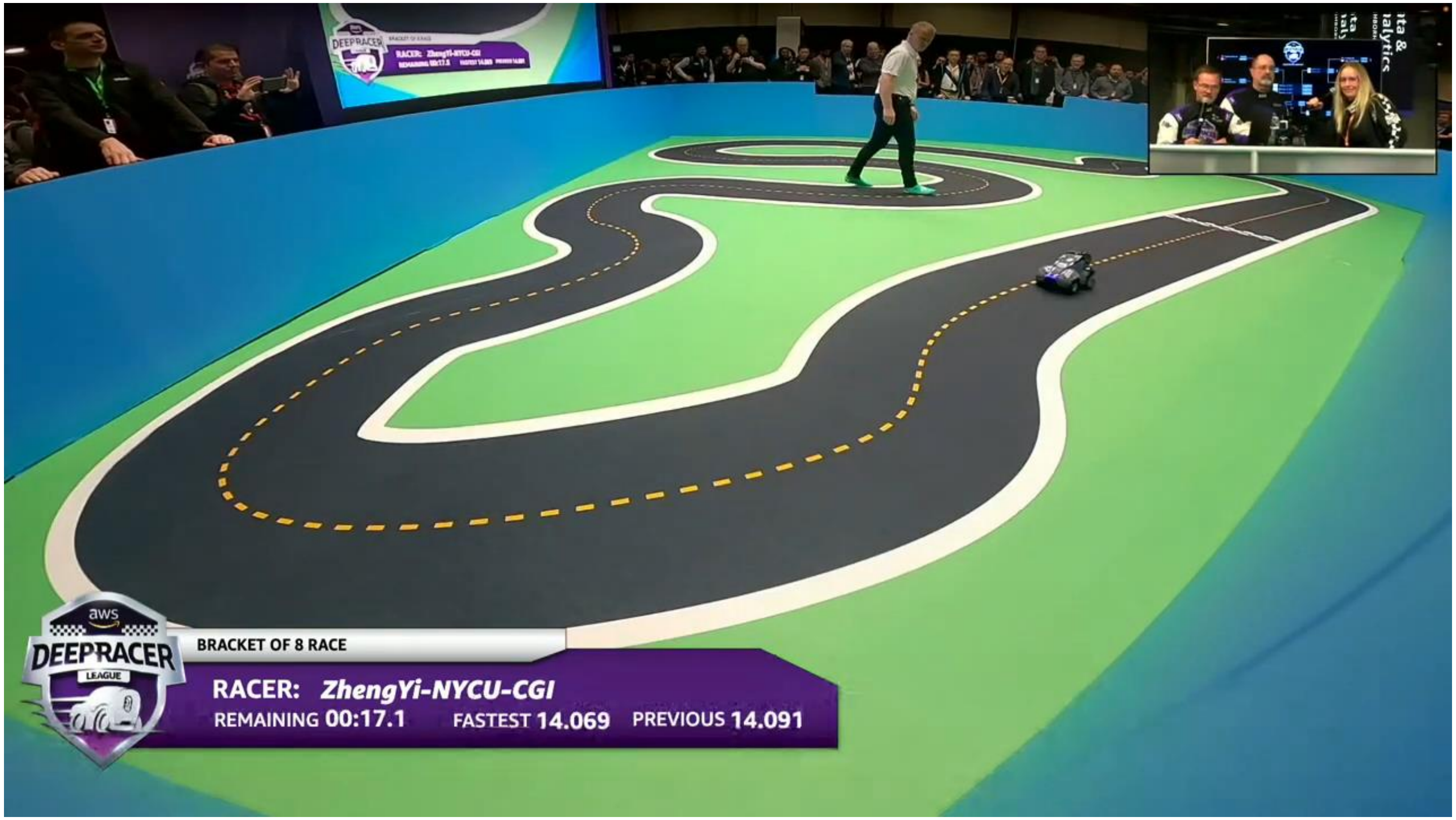
DeepRacer

- Based on RL: (by our lab CGI)
  - Won Taipei summit circuit: 1st + 3rd place
  - October AWS virtual circuit: 1st + 2nd place
  - 2019 AWS DeepRacer World Championship Cup: 3rd place
  - 2020 AWS DeepRacer World Championship Cup: 1st + 3rd places
  - 2022 AWS DeepRacer World Championship Cup: 1st + 2nd + 3rd places



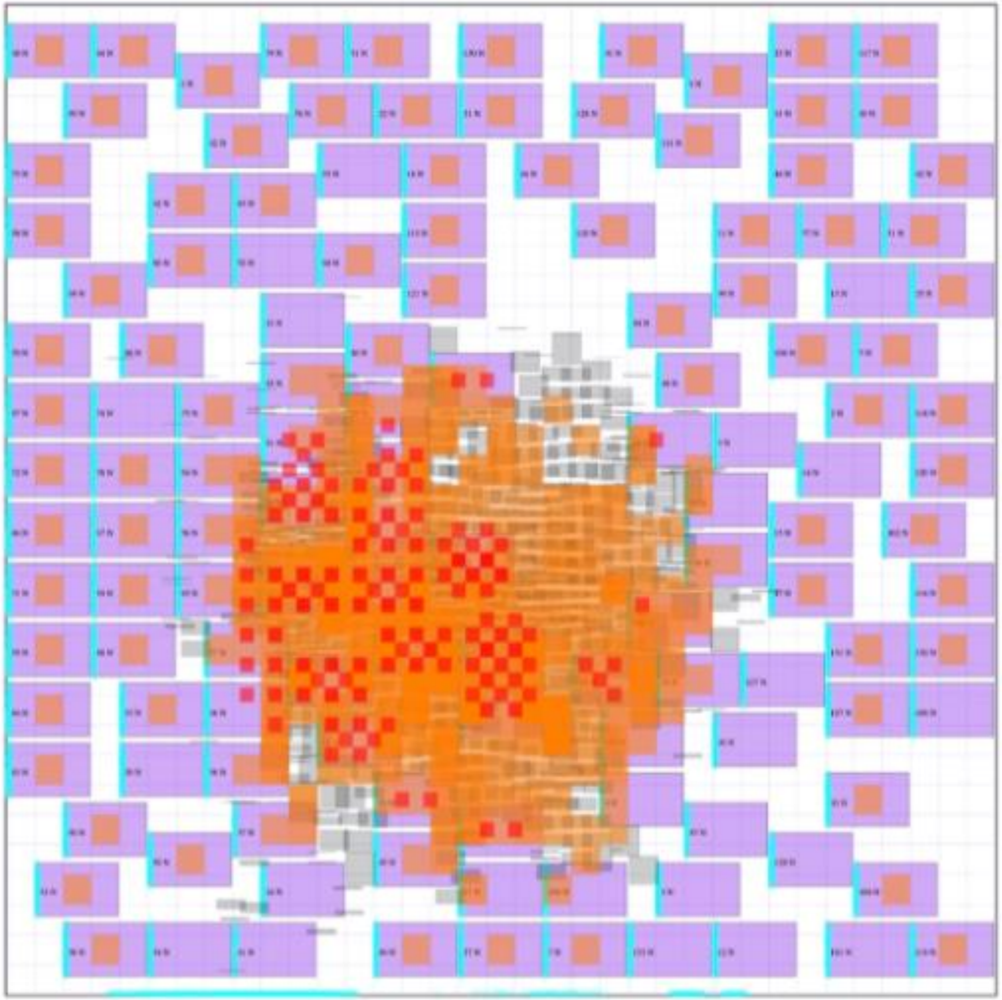
DeepRacer

- The record of AWS DeepRacer 2022: 13.756 seconds!



Better and Faster Chip Design

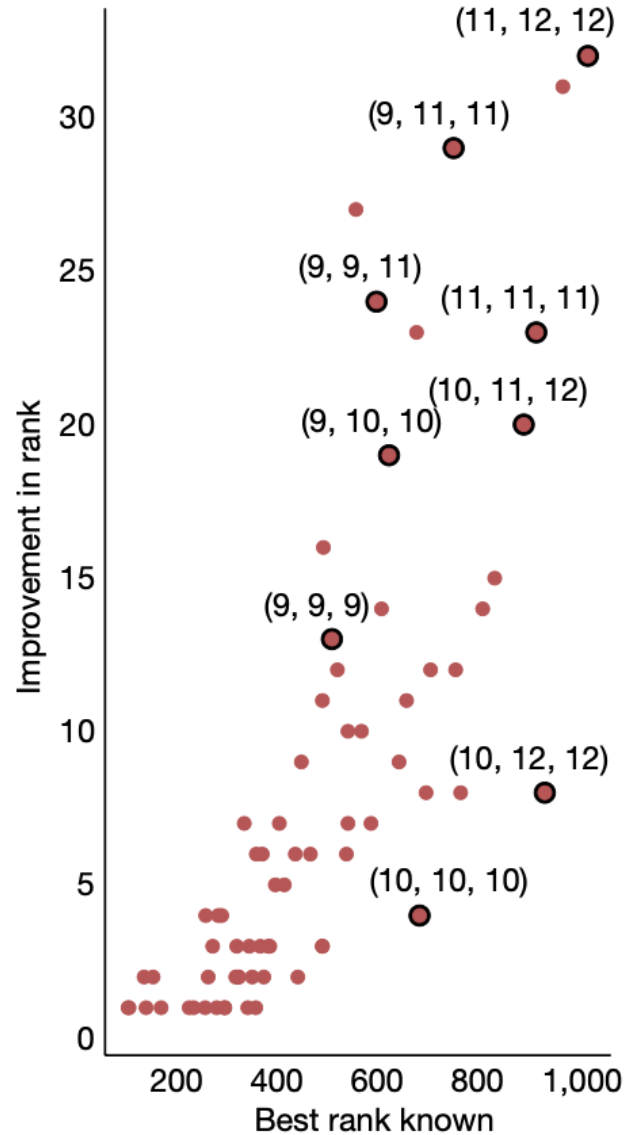
- Better and faster for chip design than any human designer.
  - Generate chip floorplans that are comparable or superior to human experts in under six hours,
  - whereas humans take months to produce acceptable floorplans for modern accelerators.



[1] A. Mirhoseini, et al. (by Google brain), A graph placement methodology for fast chip design, Nature, 2021

AlphaTensor

Size (n, m, p)	Best method known	Best rank known	AlphaTensor rank Modular	AlphaTensor rank Standard
(2, 2, 2)	(Strassen, 1969)[2]	7	7	7
(3, 3, 3)	(Laderman, 1976)[15]	23	23	23
(4, 4, 4)	(Strassen, 1969)[2] (2, 2, 2) ⊗ (2, 2, 2)	49	47	49
(5, 5, 5)	(3, 5, 5) + (2, 5, 5)	98	96	98
(2, 2, 3)	(2, 2, 2) + (2, 2, 1)	11	11	11
(2, 2, 4)	(2, 2, 2) + (2, 2, 2)	14	14	14
(2, 2, 5)	(2, 2, 2) + (2, 2, 3)	18	18	18
(2, 3, 3)	(Hopcroft and Kerr, 1971)[16]	15	15	15
(2, 3, 4)	(Hopcroft and Kerr, 1971)[16]	20	20	20
(2, 3, 5)	(Hopcroft and Kerr, 1971)[16]	25	25	25
(2, 4, 4)	(Hopcroft and Kerr, 1971)[16]	26	26	26
(2, 4, 5)	(Hopcroft and Kerr, 1971)[16]	33	33	33
(2, 5, 5)	(Hopcroft and Kerr, 1971)[16]	40	40	40
(3, 3, 4)	(Smirnov, 2013)[18]	29	29	29
(3, 3, 5)	(Smirnov, 2013)[18]	36	36	36
(3, 4, 4)	(Smirnov, 2013)[18]	38	38	38
(3, 4, 5)	(Smirnov, 2013)[18]	48	47	47
(3, 5, 5)	(Sedoglavic and Smirnov, 2021)[19]	58	58	58
(4, 4, 5)	(4, 4, 2) + (4, 4, 3)	64	63	63
(4, 5, 5)	(2, 5, 5) ⊗ (2, 1, 1)	80	76	76



ChatBot

Chat-GPT
(you know it!)

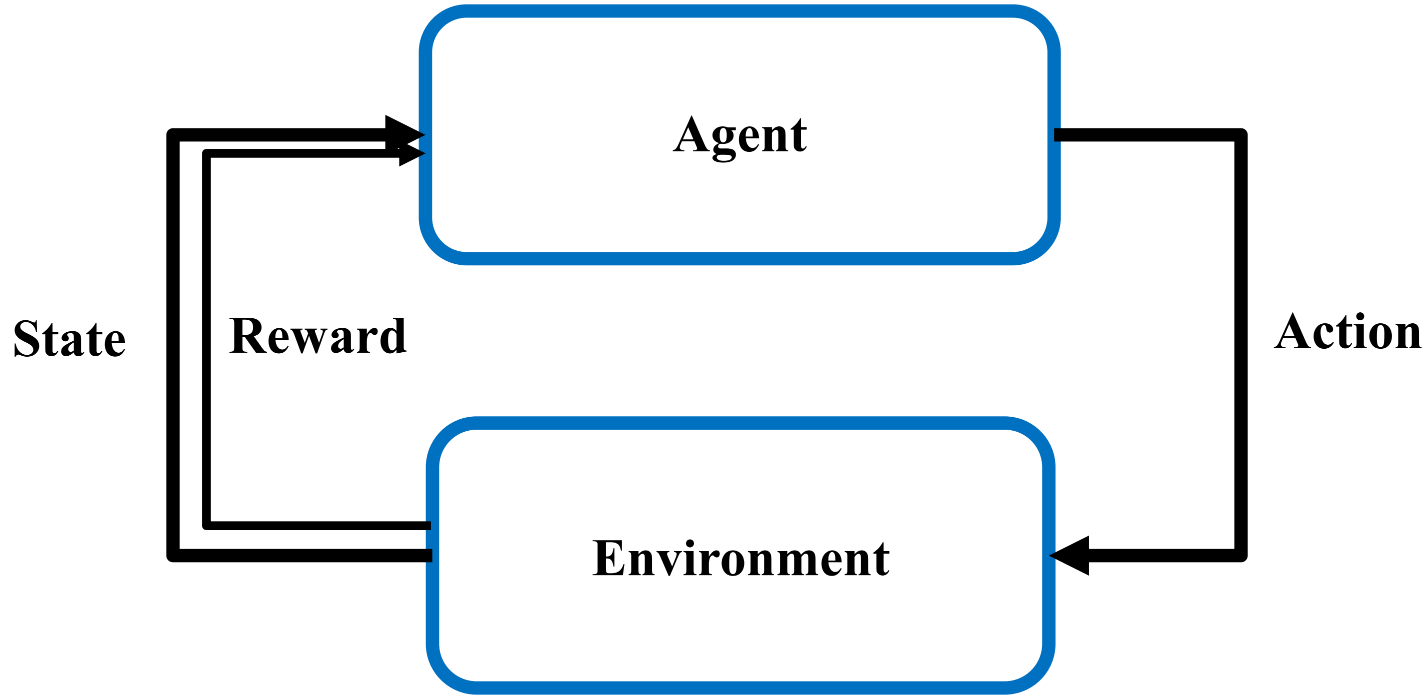
Reinforcement Learning from Human Feedback

Reinforcement Learning

- A computational approach to learning from interaction
  - Explore designs for machines that are effective in
    - solving learning problems of scientific or economic interest,
    - evaluating the designs through mathematical analysis or computational experiments.
  - Focus on goal-directed learning from interaction, when compared with other approaches to machine learning.
  - The learner must discover which actions yield the most reward by trying them.
    - Two characteristics: most important distinguishing features of reinforcement learning.
      - trial-and-error search
      - delayed reward

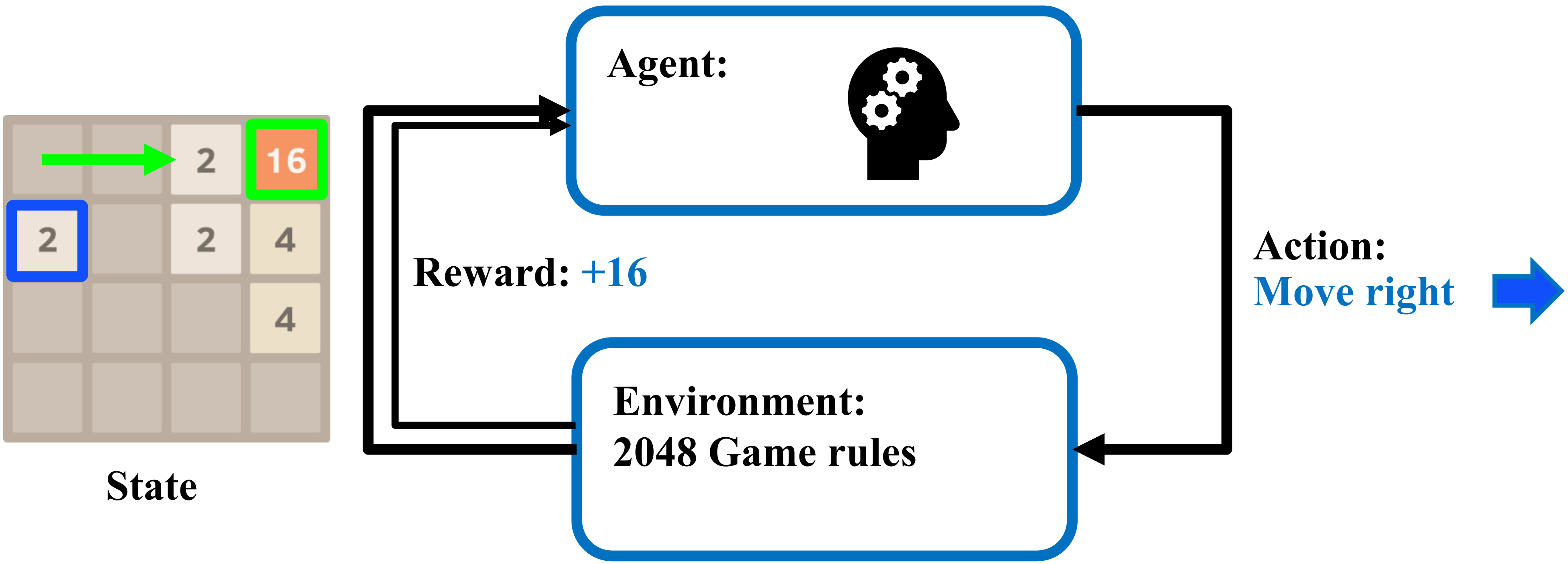
Agent-Environment Interaction Framework

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)



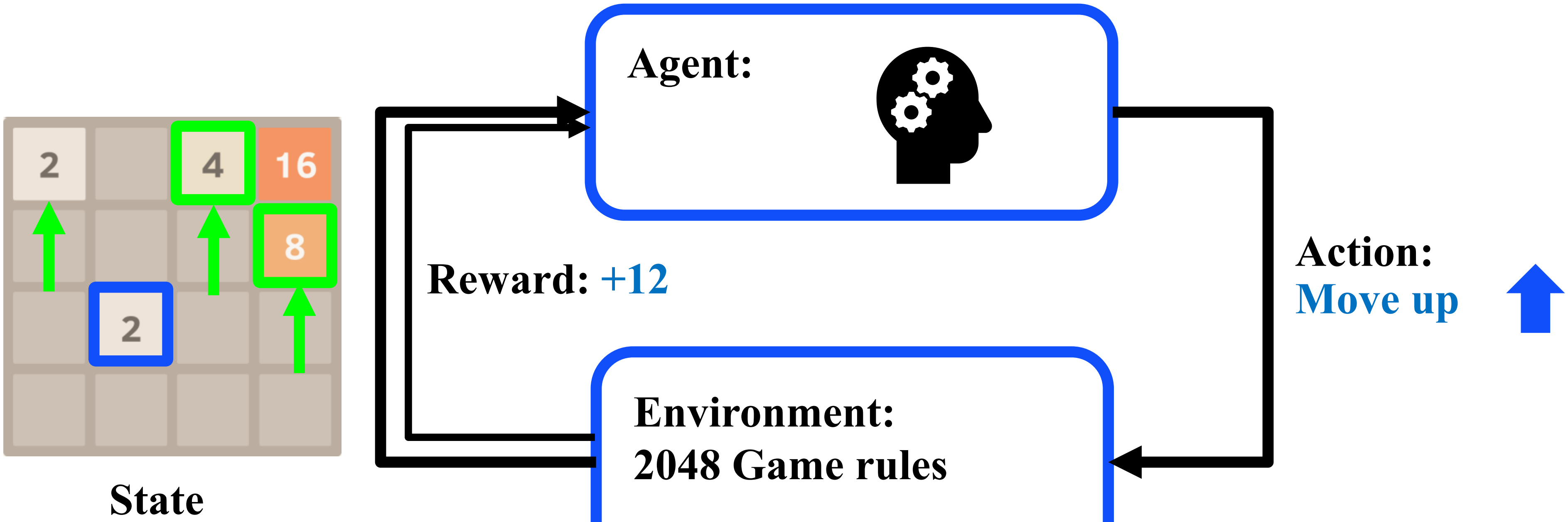
Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)



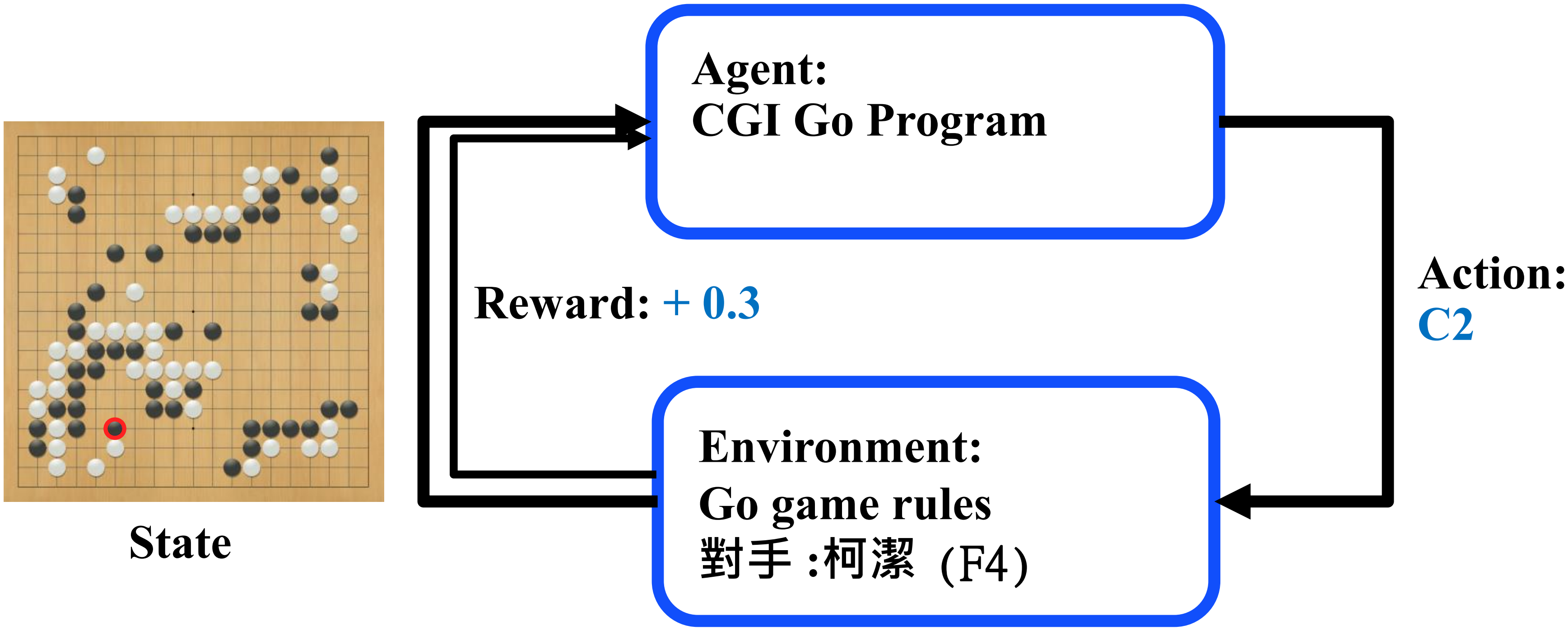
Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)



Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)

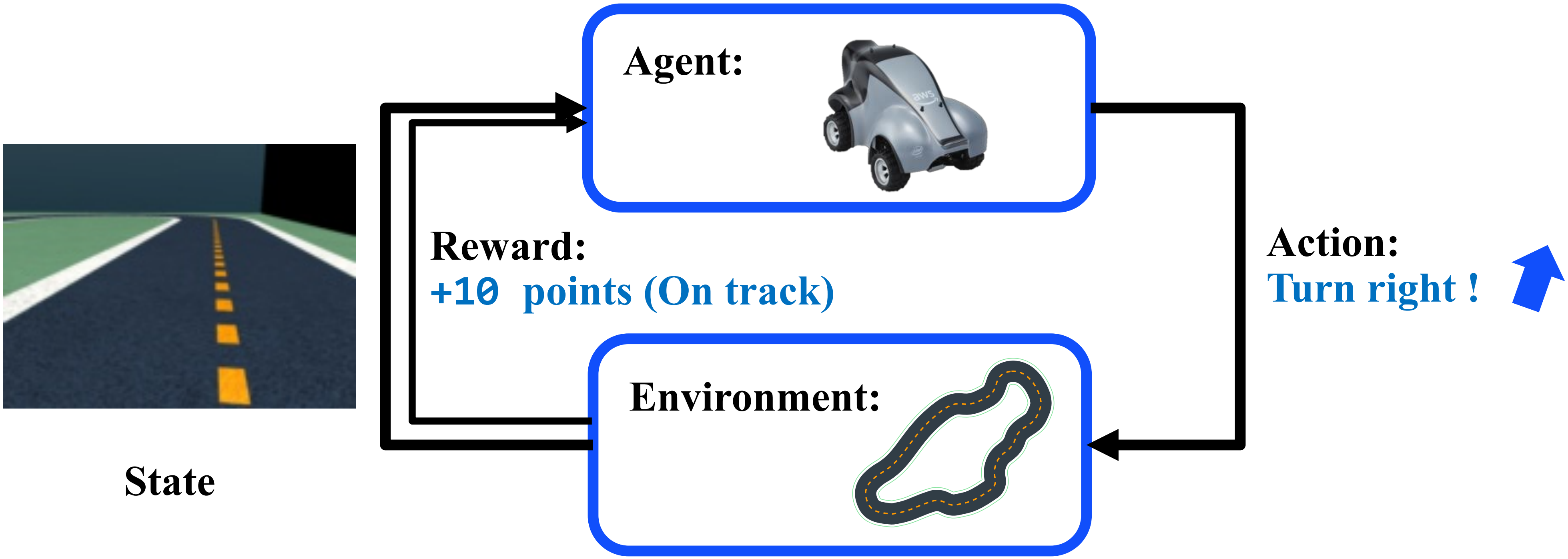


Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)

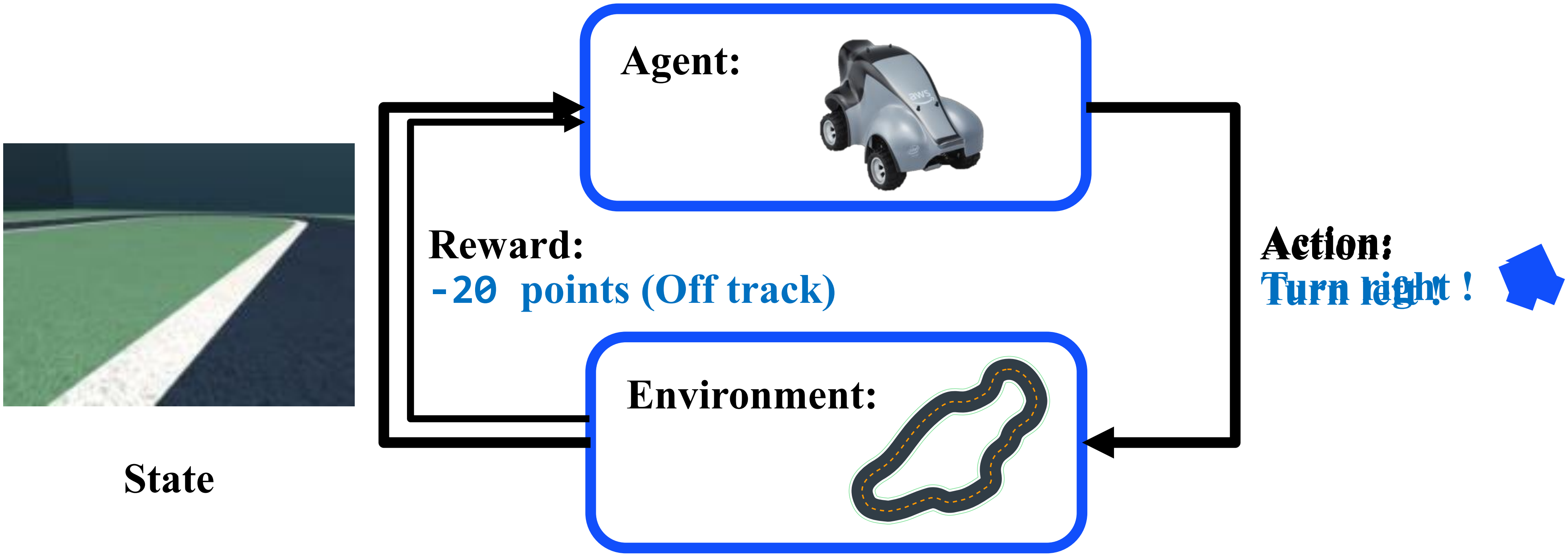
Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)



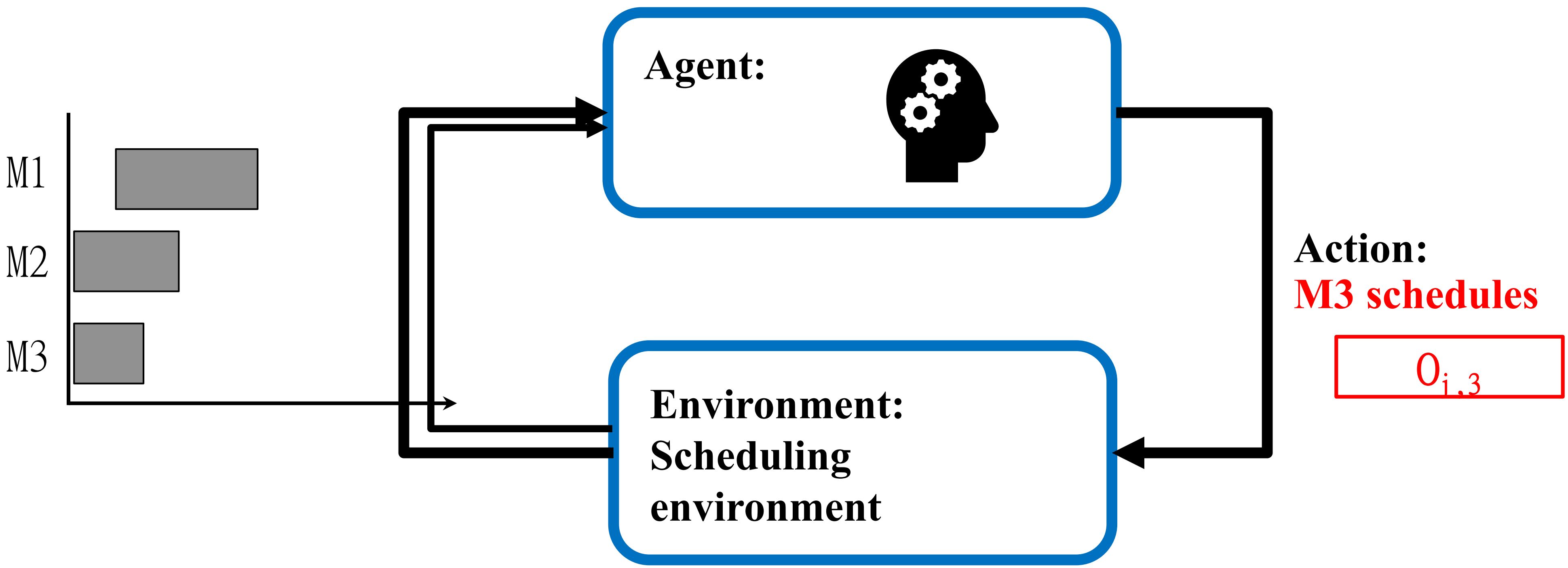
Reinforcement Learning (RL)

- A kind of AI **computational approach** to learning from interaction
- Agent-Environment Interaction Framework (代理者-環境 互動框架)



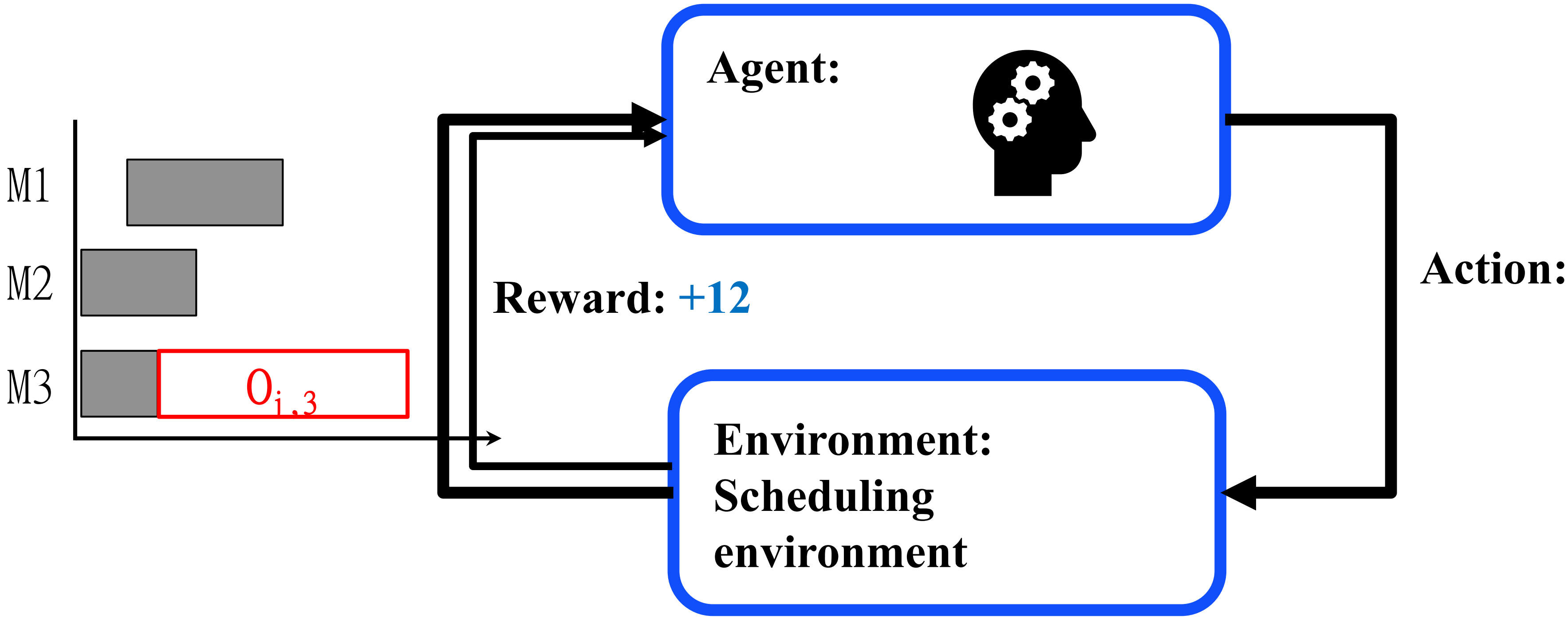
Reinforcement Learning (RL)

- A kind of AI **computational approach** to learn from interaction
- Agent-Environment Interaction Framework



Reinforcement Learning (RL)

- A kind of AI **computational approach** to learn from interaction
- Agent-Environment Interaction Framework



States and Actions in the Framework

Environment: reaction

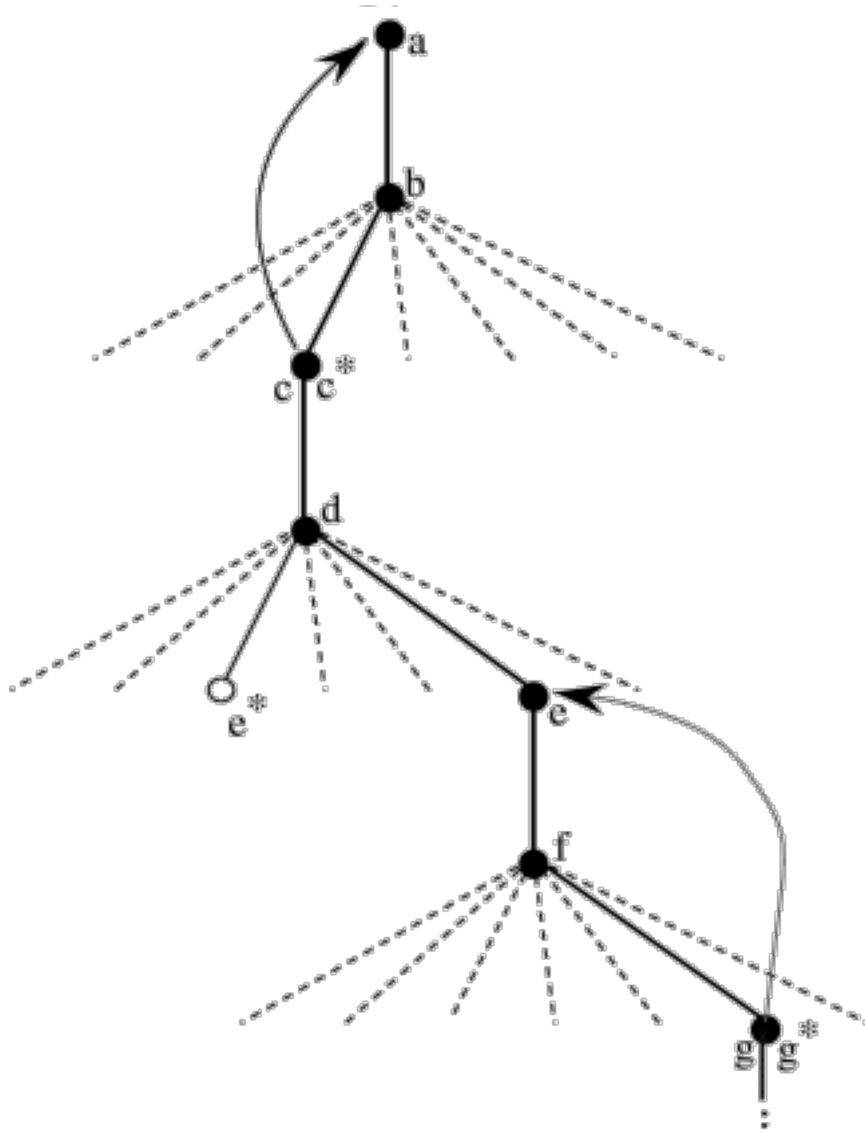
Agent: action

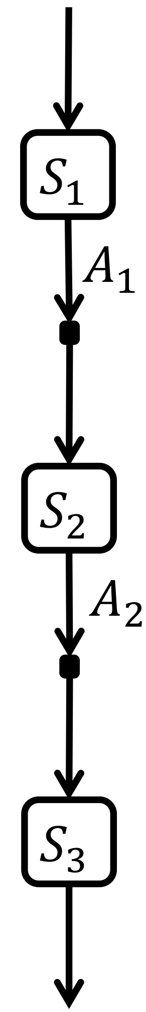
Environment: reaction

Agent: action

Environment: reaction

Agent: action





2048

Environment: Tile generation

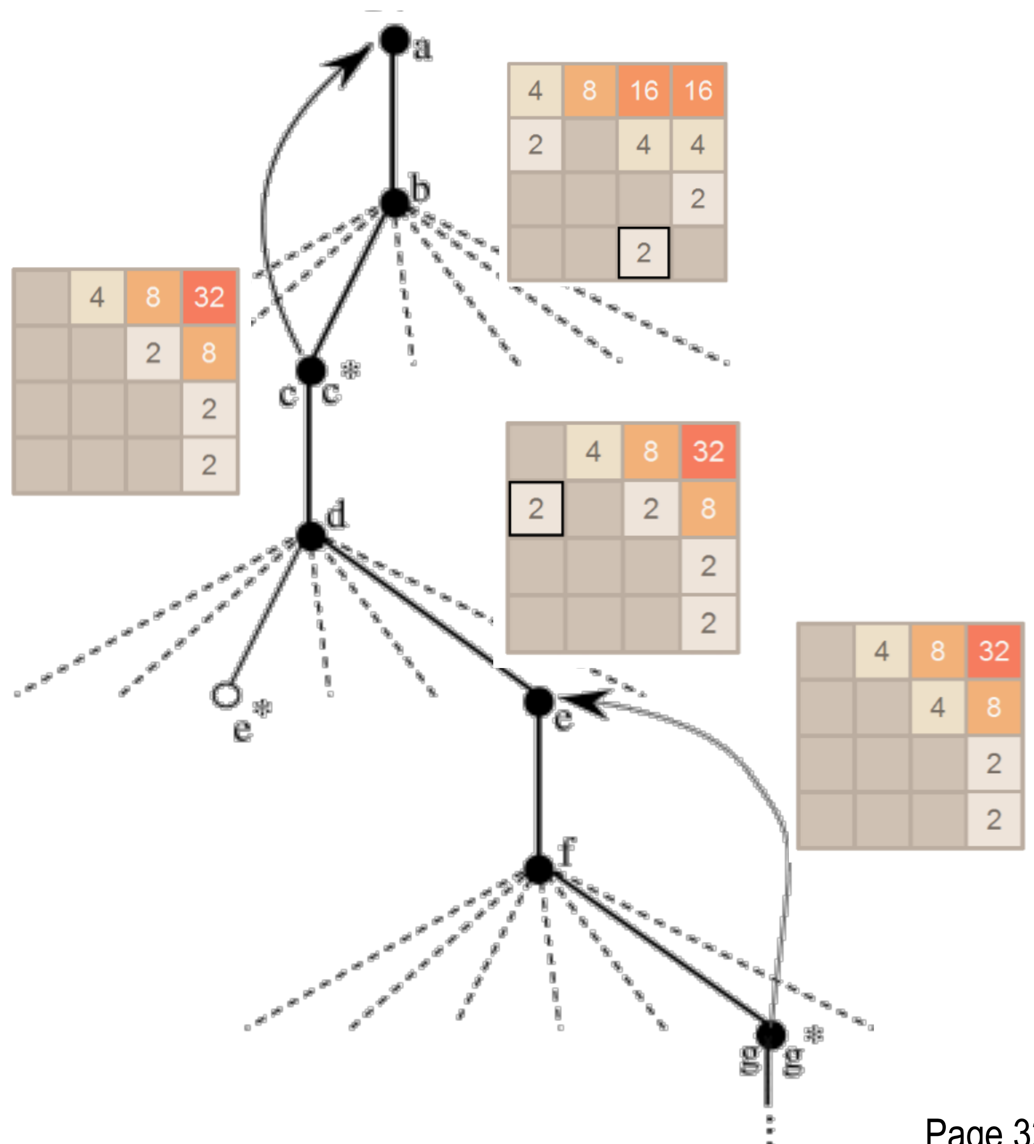
Agent: our move

Environment: Tile generation

Agent: our move

Environment: Tile generation

Agent: our move



Go

Environment: opponent's move

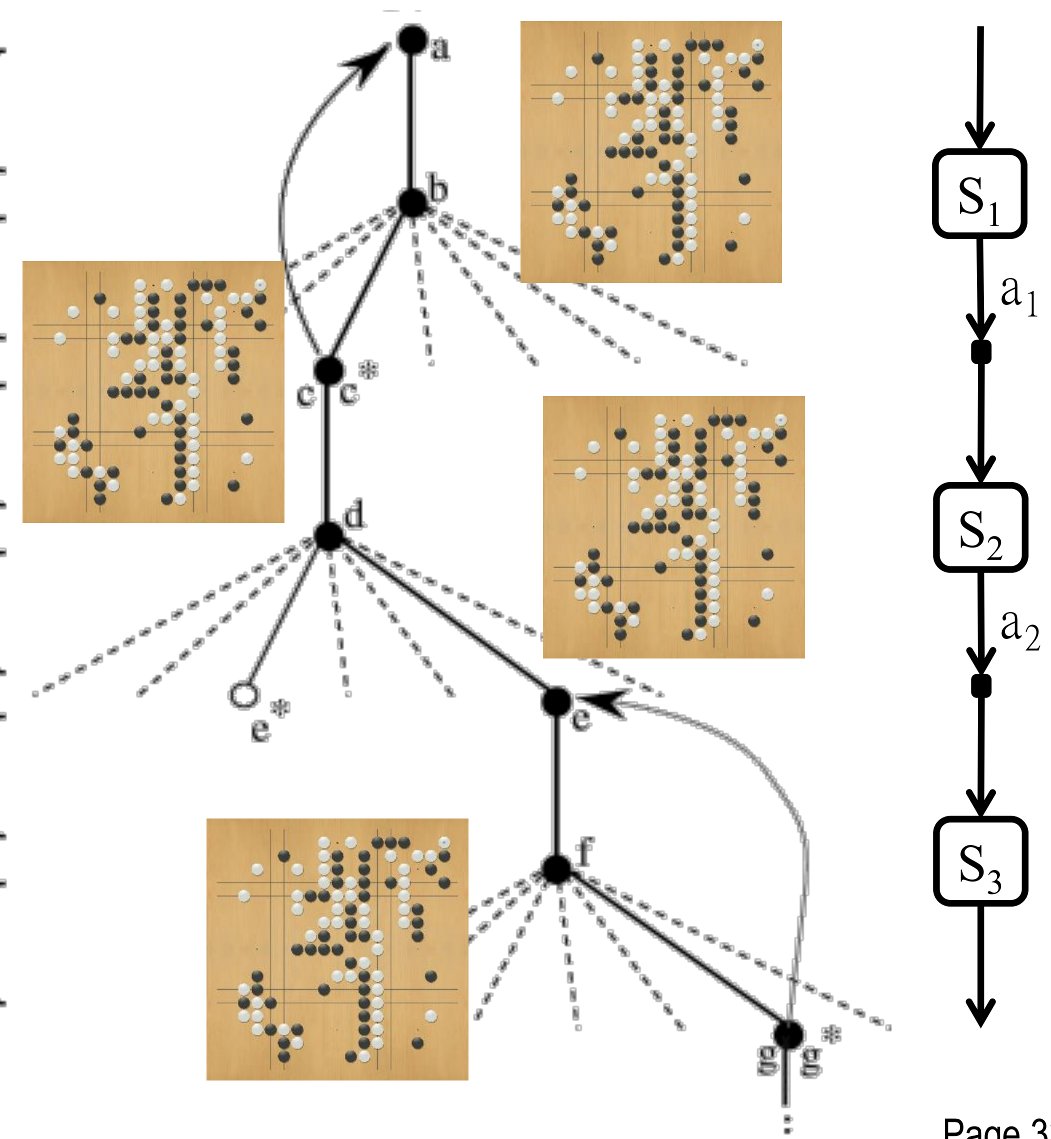
Agent: our move

Environment: opponent's move

Agent: our move

Environment: opponent's move

Agent: our move



Robot

Environment: Dynamics

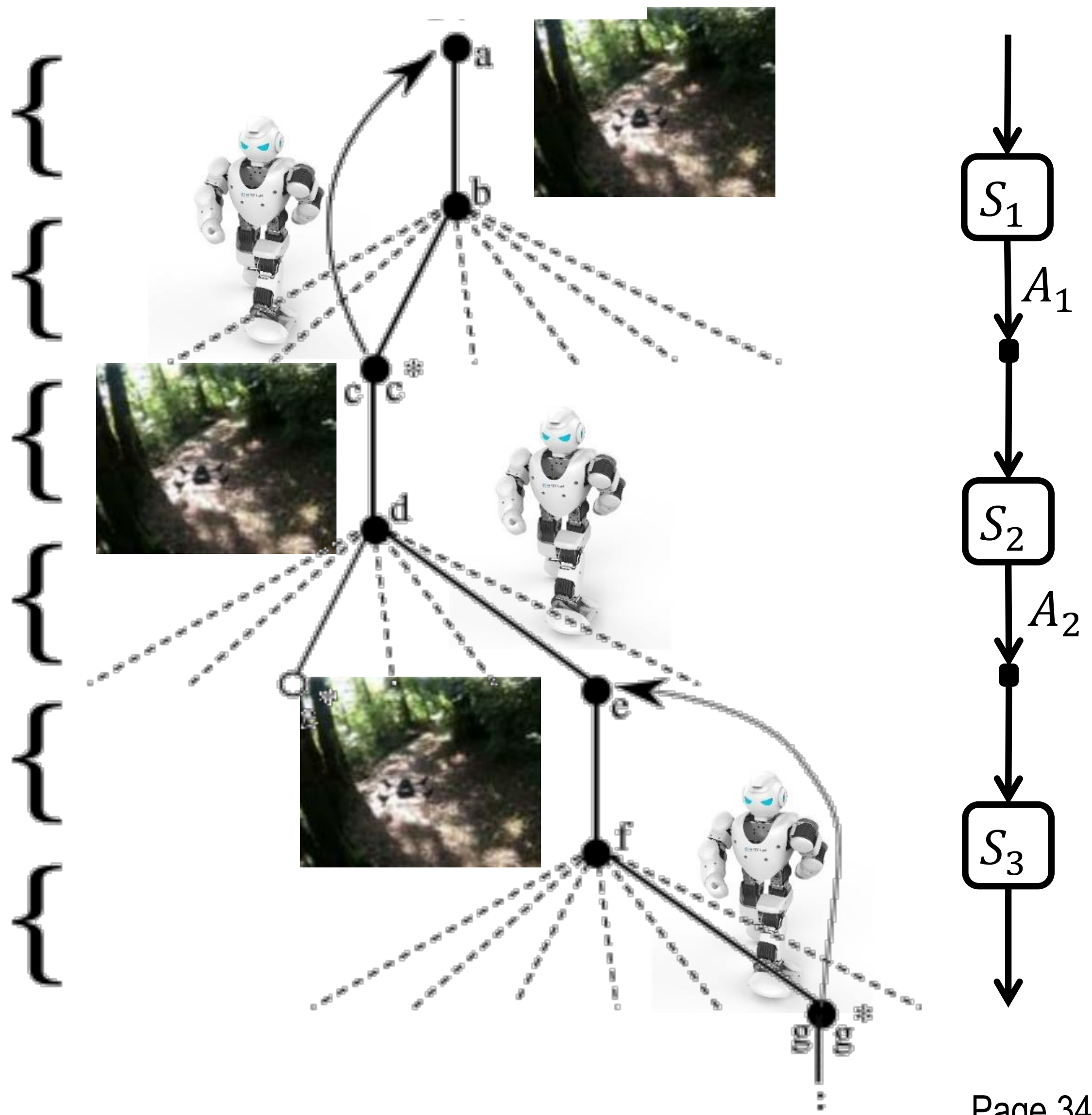
Agent: Navigate

Environment: Dynamics

Agent: Navigate

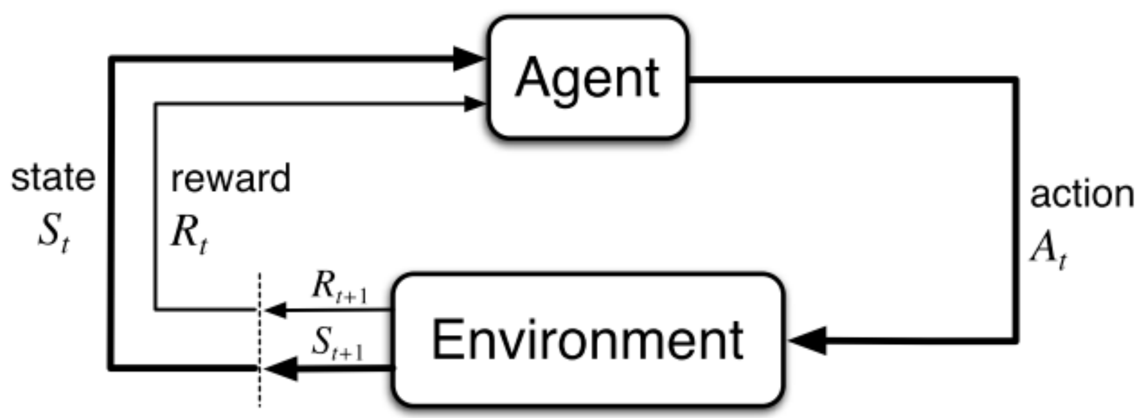
Environment: Dynamics

Agent: Navigate

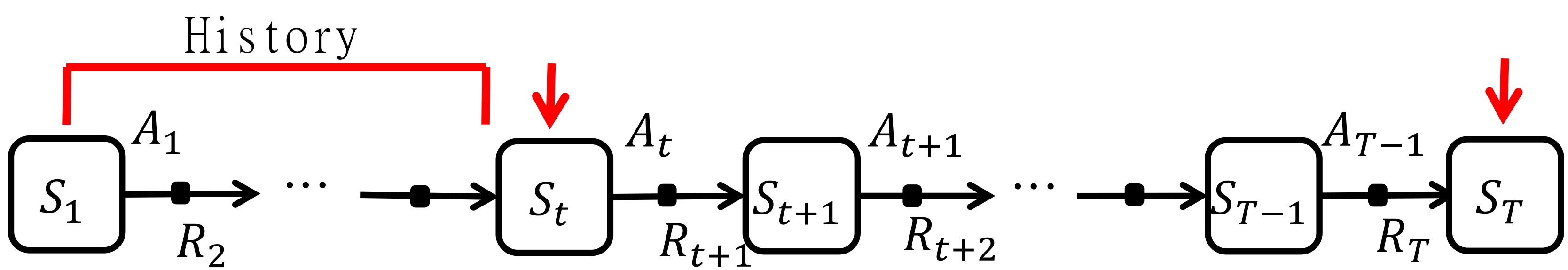


Markov Decision Processes (MDP)

- A (Finite) Markov Decision Process is a tuple $<\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma>$
  - $\mathcal{S}$ is a (finite) set of states
  - $\mathcal{A}$ is a (finite) set of actions
  - $\mathcal{P}$ is a state transition probability matrix (part of the environment),
    $\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
  - $\mathcal{R}$ is a reward function,
    $\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
  - γ is a discount factor $\gamma \in [0, 1]$.



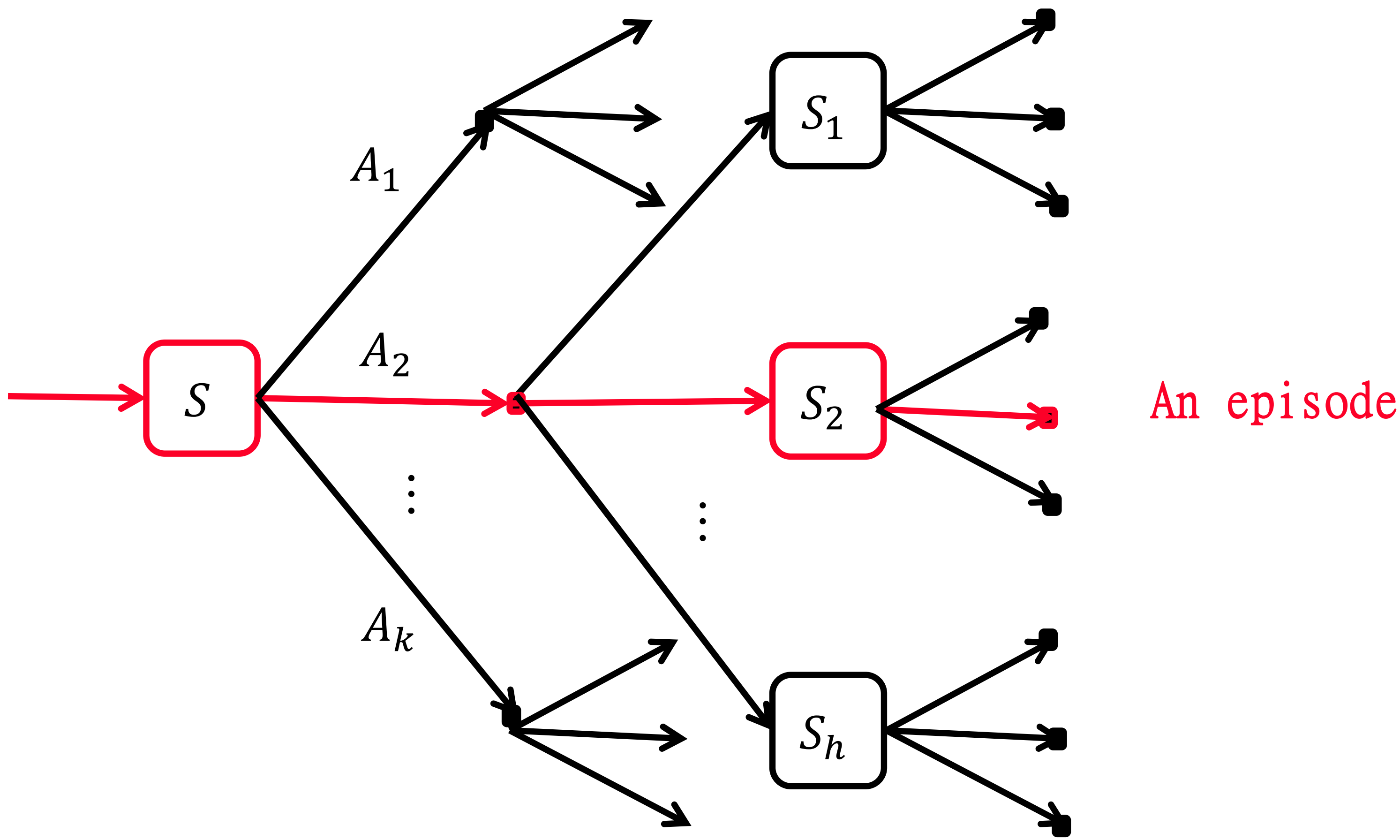
Markov Property



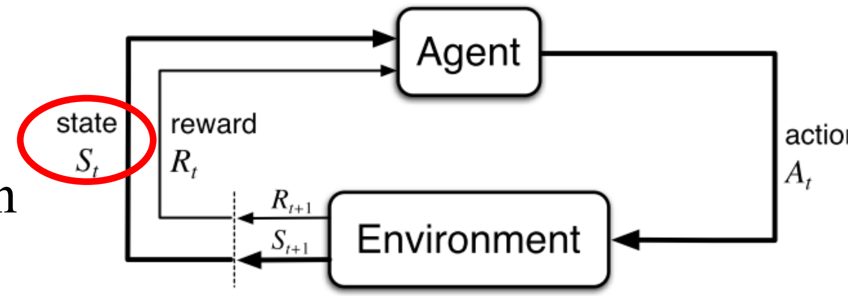
- An episode: (assuming finite and MDP here for simplicity)
  - States: S_i
    - Initial state: S_1
    - Current state: S_t
    - End state: S_T (not necessarily required)
  - Actions: A_i
  - History: $H_t = (S_1, A_1, R_2,\ S_2, A_2, R_3, S_3, \ldots, R_t)$
- Markov Property:
  - "The future is independent of the past given the present"
  - A state S_t is Markov if and only if

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

Episode and Space



Environment State vs. Agent State



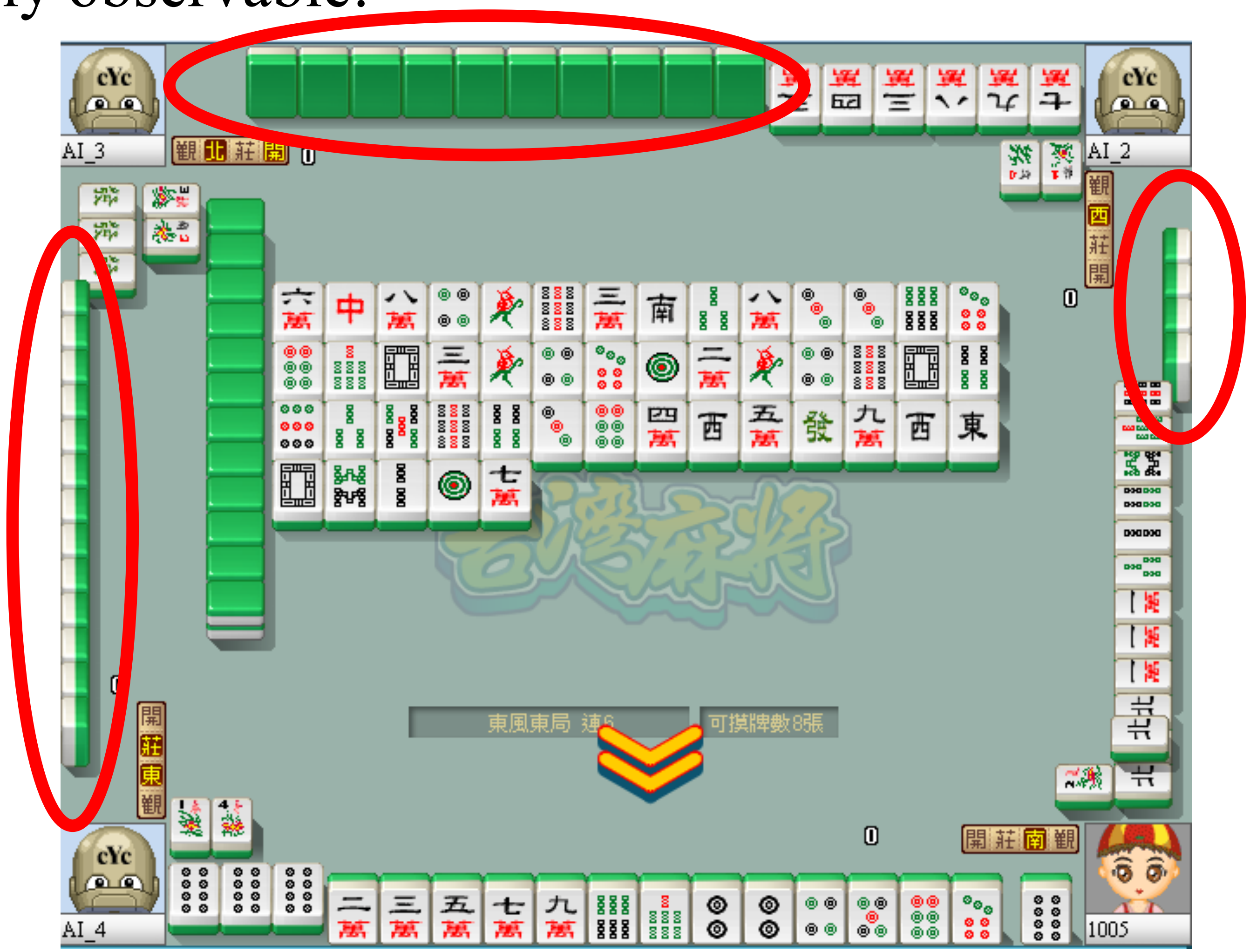
- The environment state S_t^e:
  - the environment's private representation
    - i.e. whatever data the environment uses to pick the next observation/reward
  - The environment state is not necessarily visible to the agent
    - Even if S_t^e is visible, it may contain irrelevant information
- The agent state S_t^a:
  - The agent's internal representation
    - i.e. whatever information the agent uses to pick the next action
    - i.e. it is the information used by reinforcement learning algorithms
  - It can be any function of history:

$$S_t^a = f(H_t)$$

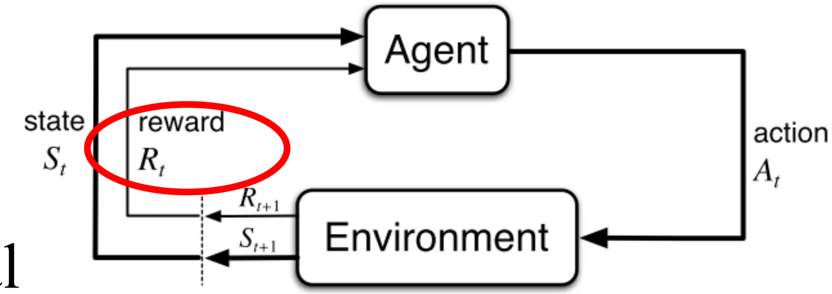
- Partially Observable: (not discussed here)
  - When $S_t^a \neq S_t^e$

Example: Mahjong

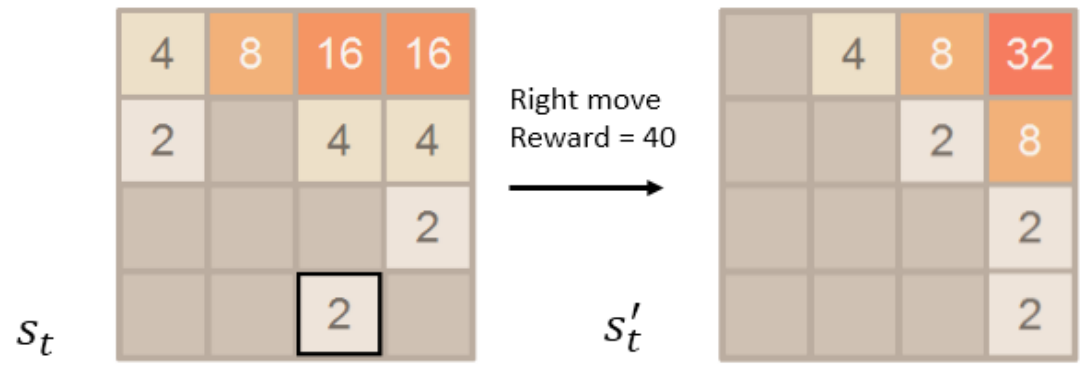
- Partially observable:



Rewards



- A reward R_t is a scalar feedback signal
  - Indicates how well agent is doing at step t
  - The agent's job is to maximize cumulative reward
  - Reinforcement learning is based on the reward hypothesis
  - Example: (2048)

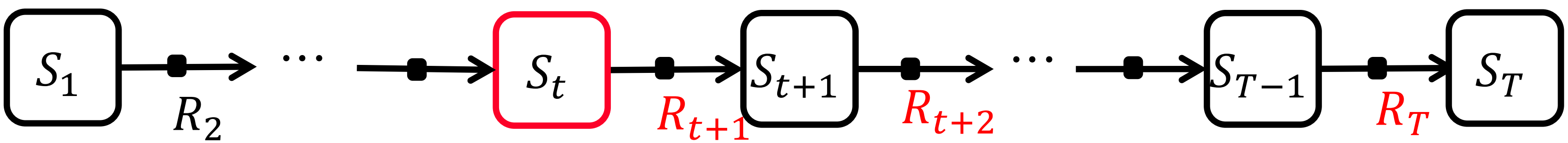


Definition (Reward Hypothesis)

- All goals can be described by the maximization of expected cumulative reward

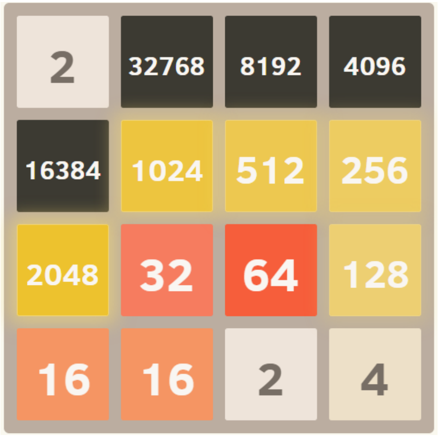
Sequential Decision Making

- Goal:
  - Select actions to maximize total future reward
- Maximize $R_{t+1} + R_{t+2} + \cdots + R_T$
  - assuming time = t.



- Notes:
  - Actions may have long term consequences
  - Reward may be delayed
  - It may be better to sacrifice immediate reward to gain more long-term reward

Sequential Decision Making – Examples



- Examples:
  - In 2048, establish a sequence of $(2^t, 2^{t-1}, 2^{t-2}, \dots)$
  - In chess, block opponent moves to help winning chances many moves from now.
  - In a financial investment, may take months to mature
  - In robotics, refuel a helicopter to prevent a crash.

Return

Definition

- The return G_t is the total discounted reward from time-step t.

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

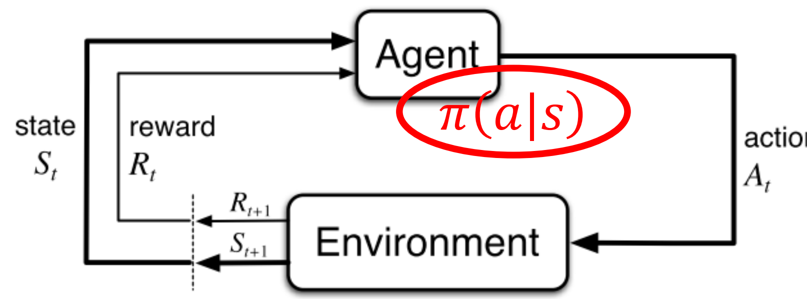
Notes:

- The discount $\gamma \in [0, 1]$ is the present value of future rewards
- The value of receiving reward R is diminishing
  - $\gamma^k R$, after $k + 1$ time-steps.
- This values immediate reward above delayed reward.
- Discount:
  - γ close to 0 leads to ”myopic” evaluation
  - γ close to 1 leads to ”far-sighted” evaluation
  - Important for infinite episodes.

Major Components of an RL Agent

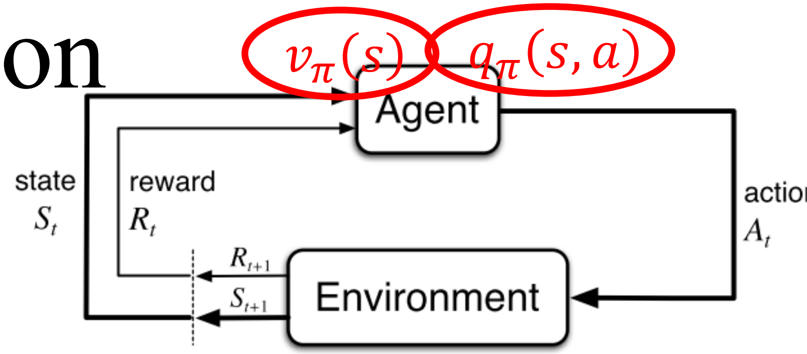
- Value function: how good is each state and/or action
- Policy: agent's behavior function
- Model: agent's representation of the environment

Policy



- A policy is the agent's behavior
  - It is a map from state to action,
- Policy types:
  - Deterministic policy: $a = \pi(s_i)$
  - Stochastic policy: $\pi(a|s) = \mathbb{P}[A_t = a|\ S_t = s]$
    - Sometimes, written in $\pi(s, a)$.
- Examples:
  - In 2048: Up/down/left/right
  - In robotics: angle/force/…

Value Function



- A value function is
  a prediction of future reward
  - Used to evaluate the goodness/badness of states
    - therefore to select between actions.
  - Return $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
- Types of value functions under policy π:
  - State value function: the expected return from s.

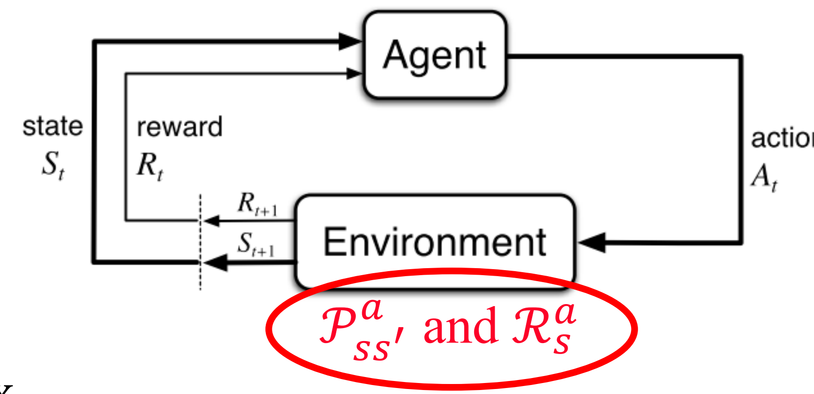
$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s]$$
$$= \mathbb{E}_\pi[G_t \mid S_t = s]$$

  - Q-Value function: the expected return from s taking action a.

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

- Examples:
  - In 2048, the expected score from a board S_t.

Model



- A model predicts what the environment will do next
  - $\mathcal{P}$ is a state transition probability matrix, $\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$
    - predicts the next state
  - $\mathcal{R}$ is a reward function, $\mathcal{R}_s^a = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$
    - predicts the next (immediate) reward
- Examples:
  - In 2048:
    - After a move, $\mathcal{P}$ is to generate a tile randomly as follows:
      - 2-tile: with probability of 9/10
      - 4-tile: with probability of 1/10

Categorizing RL Agents (Policy & Value)

- Value Based
  - No Policy (Implicit)
  - Value Function
- Policy Based
  - Policy
  - No Value Function (Implicit)
- Actor Critic
  - Policy
  - Value Function

Categorizing RL Agents (Model)

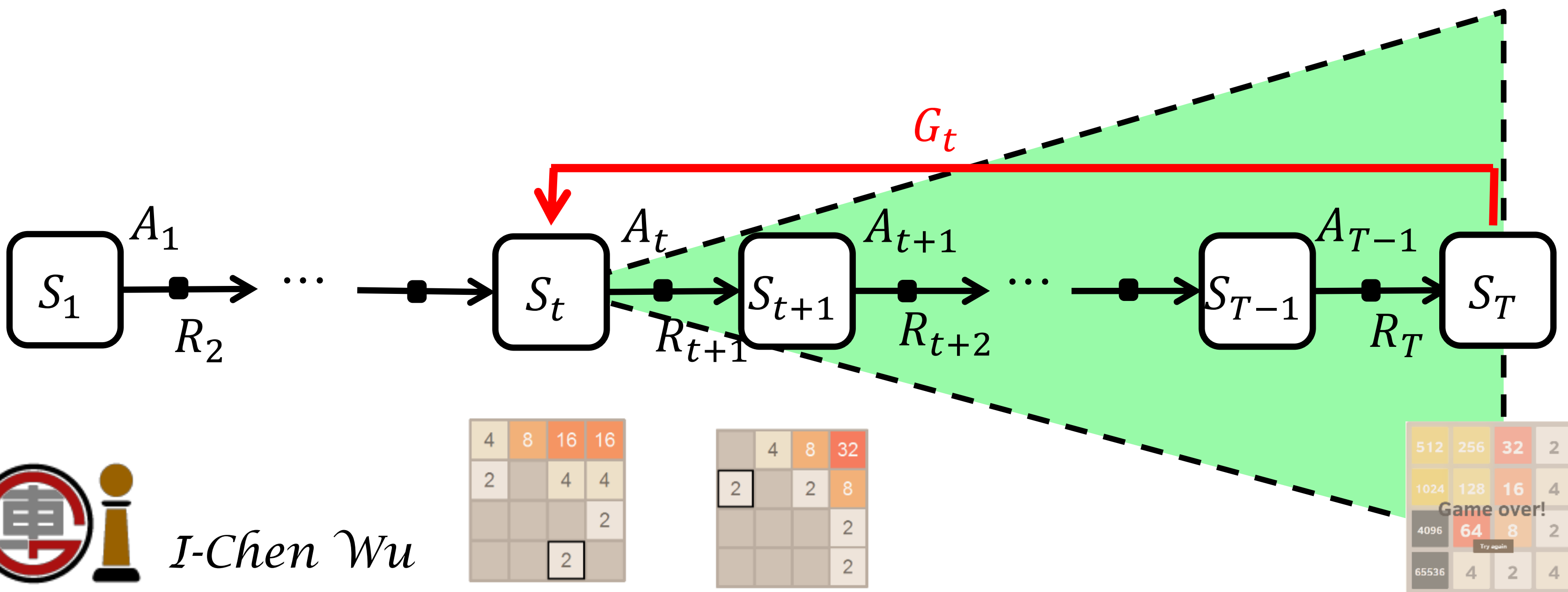
- Model Free
  - Policy and/or Value Function
  - No Model
- Model Based
  - Policy and/or Value Function
  - Model

Model-free Reinforcement Learning

- Temporal Difference (TD) Learning
  - TD methods learn directly from episodes of experience
  - TD is model-free: no knowledge of MDP transitions / rewards
  - TD learns from incomplete episodes, by bootstrapping
  - TD updates a guess towards a guess
- Monte-Carlo (MC) Learning
  - MC methods learn directly from episodes of experience
  - MC is model-free: no knowledge of MDP transitions / rewards
  - MC learns from complete episodes: no bootstrapping
  - MC uses the simplest possible idea: value = mean return
  - Caveat: can only apply MC to episodic MDPs
    - All episodes must terminate
  - Monte-Carlo Tree Search (MCTS) is a successful one based on MC learning.

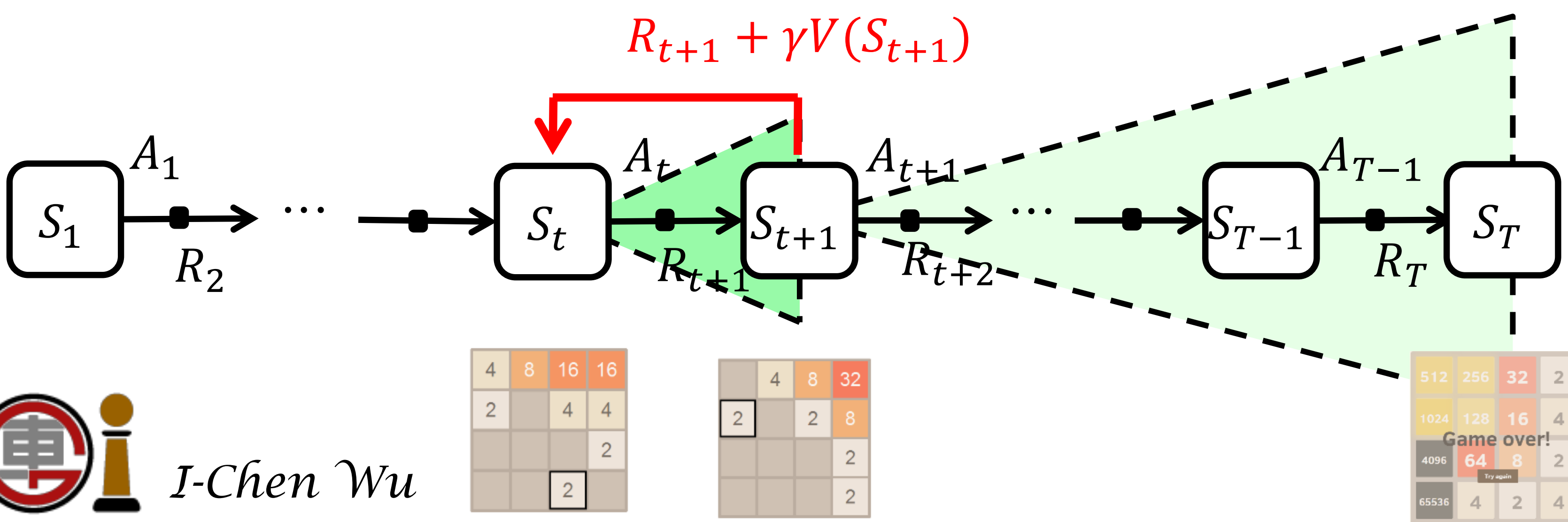
Monte-Carlo Learning

- Incremental Monte-Carlo
  - Update value $V(S_t)$ toward actual return G_t
    $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$
  - α: learning rate, or called step size.
- Unbiased, but high variance.



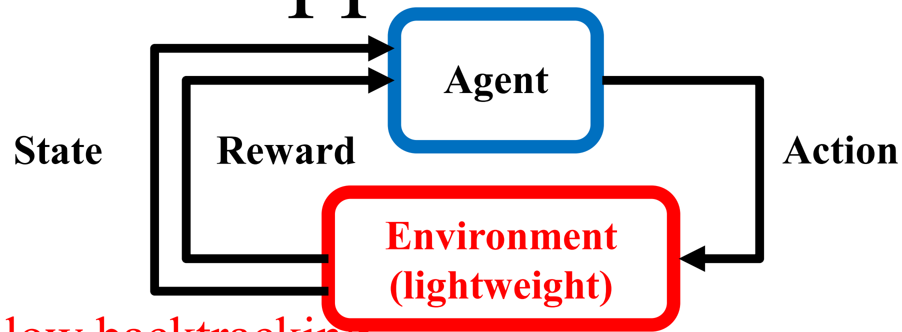
Temporal-Difference Learning

- Simplest temporal-difference learning algorithm: TD(0)
  - Update value $V(S_t)$ toward estimated return $R_{t+1} + \gamma V(S_{t+1})$
    $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
  - TD target: $R_{t+1} + \gamma V(S_{t+1})$
  - TD error: $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$
  - α: learning rate, or called step size.
- Biased, but lower variance

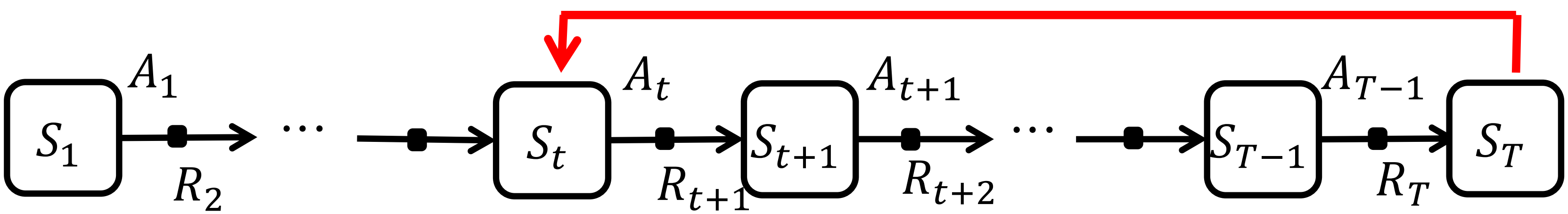


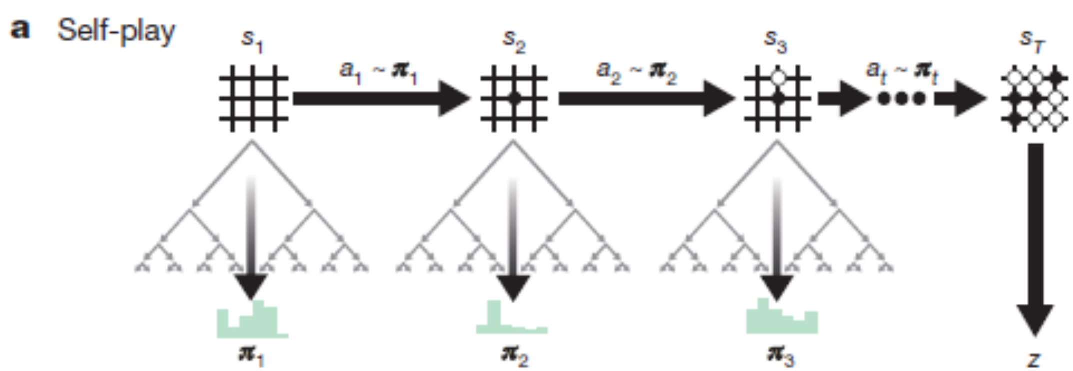
深度強化式學習應用類型
Application Classification of Deep Reinforcement Learning

Class 1: Lightweight-Model Applications

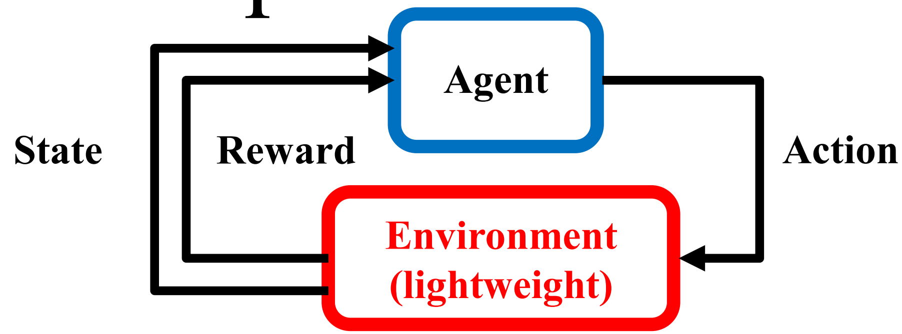


- Properties:
  - Model is well known or tractable
    - E.g., branching factor is limited.
  - Environments are simple to design, and allow backtracking
- Applications: Card/Board Games like Go, chess, etc.
- Possible Solutions: AlphaZero-like.





Related DRL Techniques

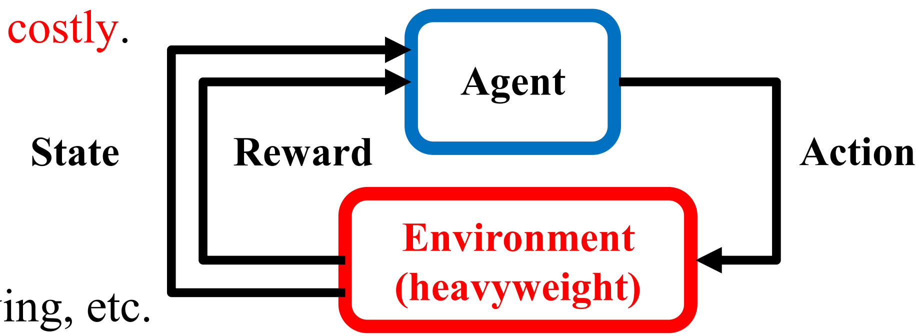


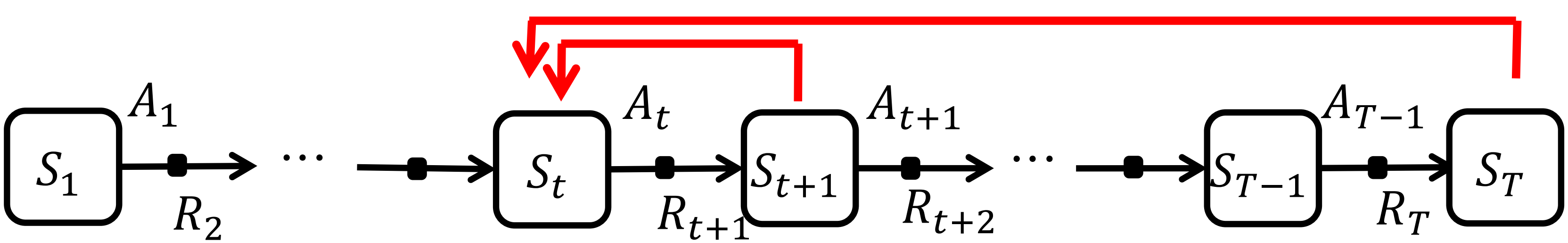
- Temporal Difference Learning
- Monte-Carlo Learning
- POMDP
- Monte-Carlo Tree Search (MCTS)
- AlphaGo/AlphaZero
- …

(more in RL2)

Class 2: Heavy-Weight-Model Applications

- Properties:
  - Model is well defined, but may be complex or intractable
    - E.g., environment dynamics are huge or continuous.
  - Simulators exist, but backtracking is hard and costly.
- Applications:
  - Video Games
  - ITM (intelligent traffic management)
  - Simulators for robots/drones/autonomous driving, etc.
  - Network resource allocation?
  - Mathematical optimization (like scheduling problems)?
- Related DRL Techniques (next pages)





Related DRL Techniques

- Value-Based:
  - **DQN**
  - DDQN
  - Deuling Network
  - Bootstrapped DQN
  - Gorrila: Distributed DQN
  - MFEC: Model-free episodic control (like 2048)
  - NEC: Neural Episodic Control
  - D3QN: Double Deuling DQN
  - Rainbow: A mix with all kinds of value-based algorithms.
  - C51: a kind of distributional method
  - QR-DQN: a kind of distributional method
  - IQN: a kind of distributional method
  - FQF: a kind of distributional method
  - Ape-X DQN: a distributed method with n-step and Double Dueling
  - R2D2: Recurrent Replay Distributed DQN

Related DRL Techniques

- Policy-based and Actor-Critic:
  - A3C: Asynchronous Advantage Actor-Critic
  - LASER: Off-Policy Actor-Critic with Shared Experience Replay (a kind of actor-critic that samples on-line sometimes)
  - ACER: Actor-Critic with Experience Replay
  - ACKTR: Actor Critic using Kronecker-Factored Trust Region (a kind of Natural Gradient)
  - TRPO: Trust-Region Policy Optimization
  - **PPO: Proximal Policy Optimization**
  - IMPALA: Importance Weighted Actor-Learner Architectures
- Miscellaneous:
  - NoisyNet and its variants
  - IDS: information directed sampling: Explore to the direction with information
  - RND: Random Network Distillation (for exploration)
  - NGU: Never Give up (for exploration; improving RND)
  - Agent57: Improve NGU
  - **muZero**

Class 3: Real-World-Model Applications

- Properties:
  - Model is unknown or too complex
  - Simulator does not exist or runs with expensive costs.
    - So, it is hard to produce a large data set.
- Applications:
  - Robots, Drones, Autonomous driving, etc.
- Related DRL Techniques:
  - Curriculum learning
  - Imitation Learning
  - Behavior Cloning
  - Transfer Learning (Sim2Real)
  - Meta Learning (one-shot/few-shot)
  - …