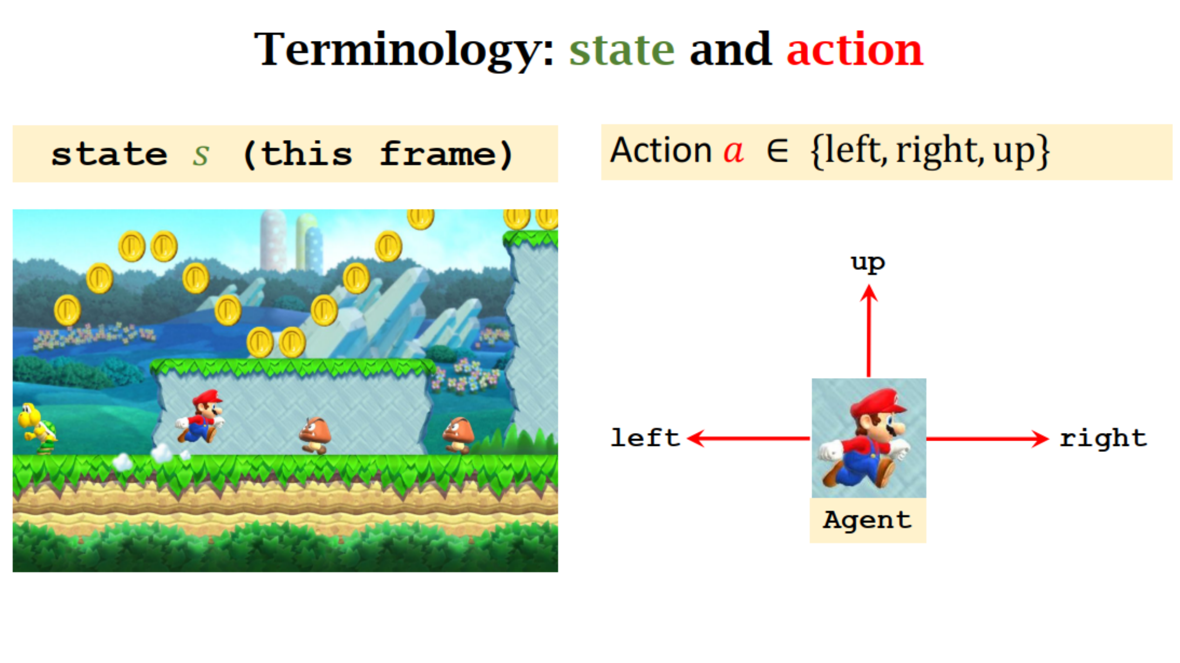
# Reinforcement Learning

## 基本名词

### State and Action

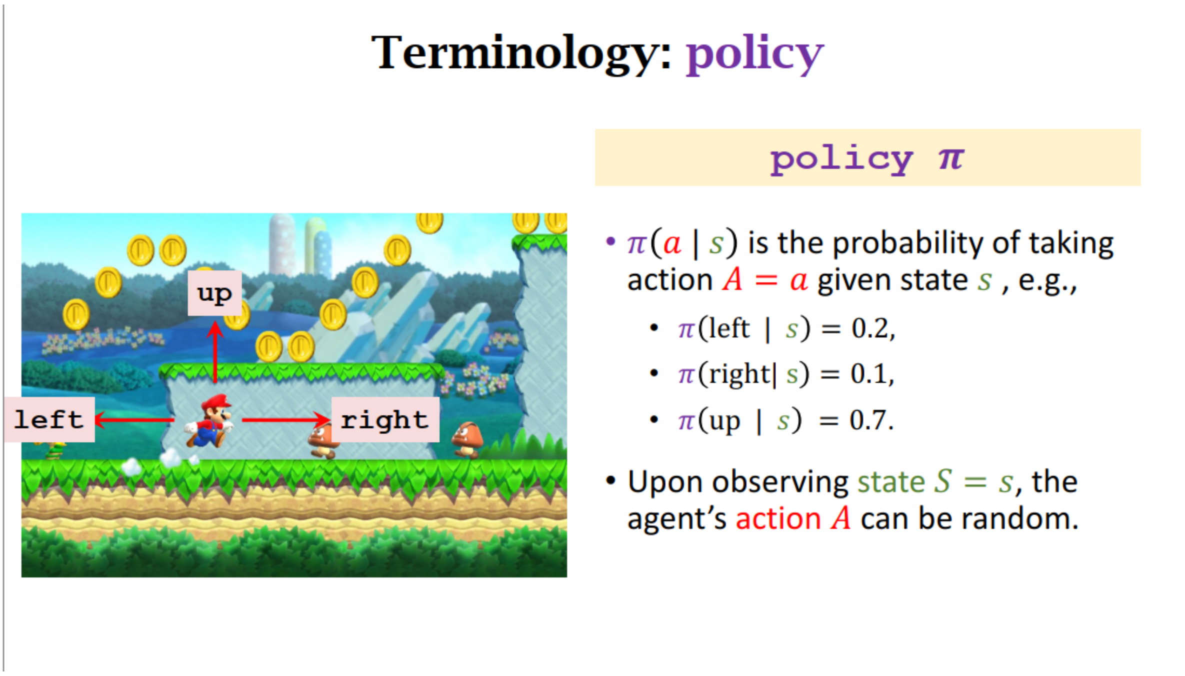



**state**是场景

**agent**为智能体，也就是执行操作的对象，此处是马里奥，也可以是机器人，车等

**action**为agent做出的动作

### Policy

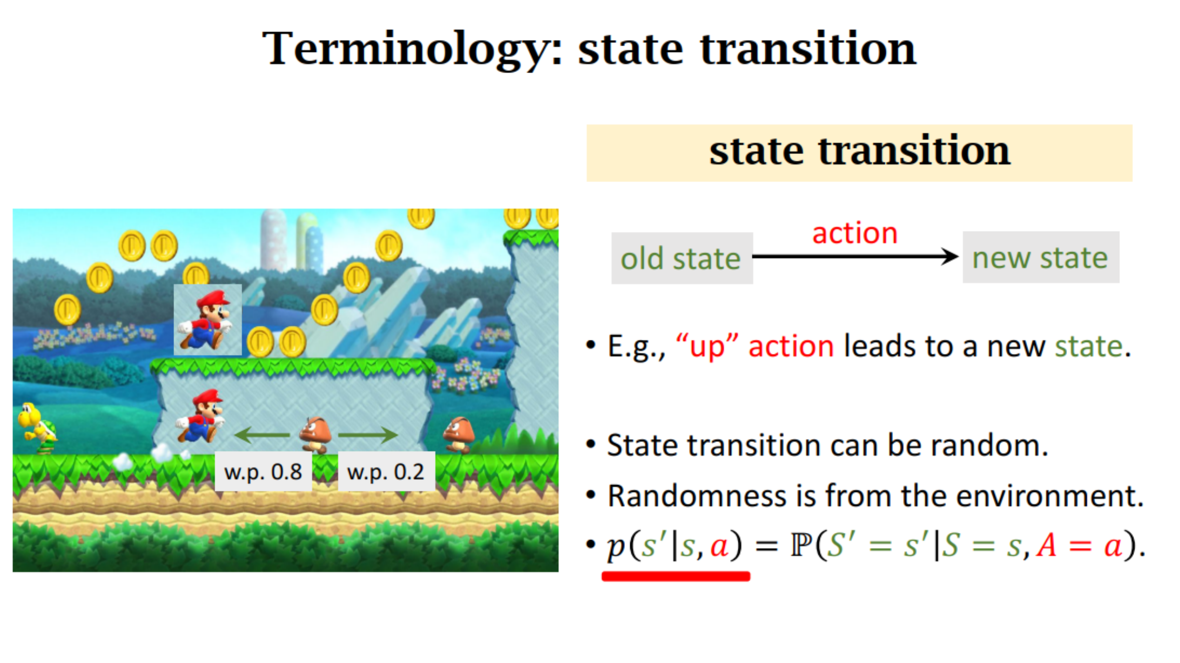


Policy函数，即策略函数，通常是一个概率分布函数，如该ppt中，在当前state下，马里奥选择往左走的策略的概率为0.2。该state下马里奥有三种策略，会在其中随机抽样，随机性使得policy更加灵活，难以被预测

## Reward


reward为奖励函数，如此处吃到金币+1分，通关+1w分，碰到敌人（game over）扣1w分

## State Transition



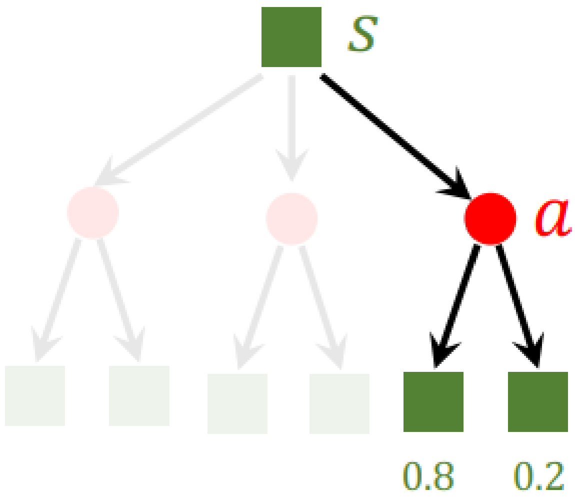

状态转移，顾名思义是old state变为new state，上图P为条件概率密度函数，意思是如果观测到当前的状态S以及动作A，下一个状态变成S一撇的概率

由于env以及action的随机性，所以状态转移具有随机性

## Two Sources of Randomness

整个过程中的随机性主要来源于两点：

1. 是action的随机性，这个很好理解
2. 是state的随机性，在马里奥的例子中可以理解为敌人移动也是随机的，无法被agent知晓




- The randomness in action is from the policy function:

$$A \sim \pi(\cdot \mid s) .$$

- The randomness in state is from the state-transition function:

$$S' \sim p(\cdot \mid s, a) .$$

## Trajectory

整个过程可以认为是

1. 观察state
2. 做出action
3. 观察到新的state并得到奖励（或惩罚）

## Play game using AI

- (state, action, reward) trajectory:

$$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \cdots \quad , \quad s_n, a_n, r_n.$$

- One episode is from the the beginning to the end (Mario wins or dies).

$$s_1 \longrightarrow a_1 \longrightarrow s_2 \longrightarrow a_2 \longrightarrow s_3 \longrightarrow a_3 \longrightarrow s_4 \longrightarrow \cdots$$
$$a_1 \searrow r_1, \quad a_2 \searrow r_2, \quad a_3 \searrow r_3$$

## Rewards and Returns

**returns**定义为未来所有cumulative future reward未来的累积奖励

- $U_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots$

但其实之后的奖励对当前时刻不是同等重要的，如选择现在给你100元和一年后给你100元，大部分人会选择立刻得到一百

所以引入折扣回报Discounted return

**Definition:** Discounted return (at time $t$).

- $U_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^{n-t} R_n.$

gamma介于0到1，为超参数

R 与 S 和 A有关

- Reward $R_i$ depends on $S_i$ and $A_i$.
- States can be random: $S_i \sim p(\cdot \mid s_{i-1}, a_{i-1})$.
- Actions can be random: $A_i \sim \pi(\cdot \mid s_i)$.
- If either $S_i$ or $A_i$ is random, then $R_i$ is random.

所以U 与未来所有S A 有关

At time $t$, the rewards, $R_t, \cdots, R_n$, are random, so the return $U_t$ is random.

- Reward $R_i$ depends on $S_i$ and $A_i$.
- $U_t$ depends on $R_t, \ R_{t+1}, \ \cdots, \ R_n$.
- ➔ $U_t$ depends on $S_t, A_t, \ S_{t+1}, A_{t+1}, \ \cdots, \ S_n, A_n$.

## Value Function

价值函数

### Action-value function

Ut其实是一个随机变量，他依赖于之后的所有动作和状态，t时刻我们并不知道Ut是什么，所以我们可以对Ut求期望得到一个函数，即Action-value function，动作价值函数，Qpi

**Definition:** Action-value function for policy $\pi$.

- $Q_\pi(s_t, a_t) = \mathbb{E}\left[U_t | S_t = s_t, A_t = a_t\right]$.

- Return $U_t$ depends on actions $A_t, A_{t+1}, A_{t+2}, \cdots$ and states $S_t, S_{t+1}, S_{t+2}, \cdots$
- Actions are random: $\mathbb{P}[A = a \mid S = s] = \pi(a|s)$. (Policy function.)
- States are random: $\mathbb{P}[S' = s' | S = s, A = a] = p(s'|s, a)$. (State transition.)

此时我们Ut未知，St和At是变量，且他们的概率分布已知（S-t，A-t分布），则可以用积分的方式把将来的SA对当前时刻Ut的影响通过积分求期望的方式获得

将t时刻之后的随机变量A， S都用积分积掉，之后得到的Qpi就只与当前时刻t的SA以及pi有关

Action-value function的实际意义，一直policy函数pi以及当前t时刻的s，则可以通过Action-value function Qpi对每个action打分，看做出哪个action，最终Ut的期望最高

### Optimal Action-value function

最优动作价值函数

之前说的Action-value function与pi，SA有关，而我们有很多个policy函数pi，我们要使得Action-value function最大，则可以做一个动态规划，取得最优的policy函数pi，使得Action-value function最大，这样就可以消除pi对Action-value function的影响（因为policy函数pi已经确定了，不再是变量），得到一个最优的Action-value function，即Optimal Action-value function

**Definition:** Optimal action-value function.

- $Q^\star(s_t, a_t) = \max_\pi Q_\pi(s_t, a_t)$.

**State-value function**

**Definition:** State-value function.

- $V_\pi(s_t) = \mathbb{E}_A\left[Q_\pi(s_t, A)\right]$

**Definition:** State-value function.

- $V_\pi(s_t) = \mathbb{E}_A\left[Q_\pi(s_t, A)\right] = \sum_a \pi(a|s_t) \cdot Q_\pi(s_t, a)$. (Actions are discrete.)
- $V_\pi(s_t) = \mathbb{E}_A\left[Q_\pi(s_t, A)\right] = \underline{\int \pi(a|s_t) \cdot Q_\pi(s_t, a)\, da}$ . (Actions are continuous.)

将Qpi对A求期望（将A当作随机变量），从而消掉A

物理意义在于可以评估目前状态的胜算

**Conclusion**

# Understanding the Value Functions

- Action-value function: $Q_\pi(s, a) = \mathbb{E}\left[U_t | S_t = s, A_t = a\right]$.
- Given policy $\pi$, $Q_\pi(s, a)$ evaluates how good it is for an agent to pick action $a$ while being in state $s$.

- State-value function: $V_\pi(s) = \underline{\mathbb{E}_A\left[Q_\pi(s, A)\right]}$
- For fixed policy $\pi$, $V_\pi(s)$ evaluates how good the situation is in state $s$.
- $\underline{\mathbb{E}_S\left[V_\pi(S)\right]}$ evaluates how good the policy $\pi$ is.

## How does AI control the agent

1. policy-based learning 策略学习：已知pi，S，可以求得每个A的概率，再随机抽样
2. value-based learning 价值学习：求Optimal Action-value function Q-star

# How does AI control the agent?

Suppose we have a good policy $\pi(a|s)$.

- Upon observe the state $s_t$,
- random sampling: $a_t \sim \pi(\cdot|s_t)$.

Suppose we know the optimal action-value function $Q^\star(s, a)$.

- Upon observe the state $s_t$,
- choose the action that maximizes the value: $a_t = \mathrm{argmax}_a \, Q^\star(s_t, a)$.