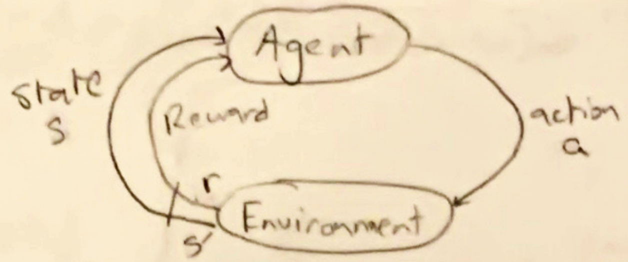

states: $\{s \in S\}$

actions: $\{a \in A\}$

reward: $R(s)$

policy: $\pi(a|s) \rightarrow$ confidence of this being the right action

discount factor: $\gamma < 1$

transitional probability: $p(s', r|s, a)$

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

Value function: $V^\pi(s) = E^\pi[G_t | S_t = s]$ $\quad E[X] = \sum_i p(x = x_i) x_i$

$$= E^\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

$$= E^\pi\left[R_{t+1} + \sum_{k=1}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

$$= E^\pi\left[R_{t+1} + \sum_{k=0}^{\infty} \gamma^{k+1} R_{t+k+2} \middle| S_t = s\right]$$

$$= E^\pi\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_t = s\right]$$

$$= \sum_a \pi(a|s) E^{s',r}\left[R_{t+1} + \gamma E^\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \middle| S_{t+1} = s'\right]\right]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma E^\pi\left[\sum \gamma^k R_{(t+1)+k+1} \middle| S_t = s'\right]\right]$$

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma V^\pi(s')\right]$$

Bellman Equation: $\boxed{V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\left[r + \gamma V^\pi(s')\right] \ \forall s \in S}$

When the environment is determinalistic $p(s', r|s, a) \in \{0, 1\}$ and an action in a specific state will always lead to the same resultant state

If $\pi(a|s) \in \{0, 1\}$, which is a definite policy, and the environment is determinalistic then a simpler bellman equation can be used

$$V^\pi(s) = \max_a\left[r + \gamma V^\pi(s')\right] \ \forall s \in S$$

Iterative Policy Evaluation: finding $V^\pi(s)$ from $\pi(a|s)$

- Random Policy: $\pi(a|s) = \dfrac{1}{\text{len(actions(s))}} \;\forall s \in S,\ a \in \text{actions}(s)$

```
while True:
  ∇=0
  for s ∈ S:
    old_v = V^π(s)
    if s is not terminal:
      new_v=0
      π(a|s) = 1/len(actions(s)) ∀ a ∈ actions(s)
      for a ∈ actions(s):
        grid.set_state(s)
        r = grid.move(a)
        new_v += π(a|s)[r + γ V^π(s')]
      V^π(s) = new_v
      ∇ = max(∇, |new_v-old_v|)
  if ∇<ε:
    break
```

determinalistic
$p(s', r|s,a) \in \{0, 1\}$

- Definite Policy: $\pi(a|s) \in \{0,1\} \;\forall s \in S,\ a \in \text{actions}(s) \,\Big|\,$ given $\pi(a|s)$

```
while True:
  ∇=0
  for s ∈ S:
    old_v = V^π(s)
    if s is not terminal:
      new_v=0
      for a ∈ actions(s):
        grid.set_state(s)
        r= grid.move(a)
        new_v+=p(s',r|s,a)[r+γV(s')]
      V^π(s) = new_v
      ∇ = max(∇, |new_v-old_v|)
  if ∇<ε:
    break
```

not determinalistic
$p(s', r|s,a) \in [0, 1]$

definite policies can be written as

$\pi(s) = a$
where $\pi(a|s) = 1$

Policy Improvement: finding a better policy π' so that $V^{\pi}(s') \leq V^{\pi'}(s')$

Policy Iteration: going from $V^{\pi}(s) \to \pi'$ Solving the control problem

```
policy_changed = False
for s ∈ S
    old_a = π(s)
    π(s) = argmax_a [ Σ_s' Σ_r p(s',r|s,a) [r + γV(s')] ]
    if π(s) != old_a:
        policy_changed = True
```

```
new_a = None
best_value = -∞
for a ∈ actions(s):
    v = 0
    for a2 ∈ actions(s):
        grid.set_state(s)
        r = grid.move(a2)
        v += p(s',r|s,a)[r + γV(s')]
    if v > best_value
        new_a = a
        best_value = v
π(s) = new_a
```

a is the action the agent attempted

a2 is what actually happend

Workflow

```
① initialize V^π(s) and π(s)
while True:
    ② V^π(s) = iterative_policy_evaluation(π)
    ③ π(s), policy_changed = policy_iteration(V^π)
    ④ if not policy_changed:
           break
```

Value Iteration: alternative to policy iteration and iteritive policy evaluation

↳ combines policy evaluation and policy improvement into one step

$$V_{k+1}(s) = \max_a \sum_{s'} \sum_r p(s',r|s,a) \{r + \gamma V_k(s')\}$$

↳ This is only possible since policy iteration uses argmax and policy evalution will just use the value at this maximum, by using in max stright into $V(s)$ allows use to skip calculating $\pi(s)$ till the optimal value function is found

```
initialize V(s) = 0  ∀ s ∈ S
while True:
    Δ = 0
    for s ∈ S:
        old_v = V(s)
        V^π(s) = max_a Σ_s' Σ_r p(s',r|s,a) {r + γV^π(s')}
        Δ = max(Δ, |V^π(s) - old_v|)
    if Δ < ε:
        break
```

```
new_v = -∞
for a ∈ actions(s)
    grid.set_state(s)
    r = grid.move(a)
    v = r + γV(s')
    if v > new_v:
        new_v = v
V^π(s) = new_v
```

calculating $V^{\pi}(s)$

```
for s ∈ S:
    π(s) = argmax_a Σ_s' Σ_r p(s',r|s,a) {r + γV^π(s')}
```

Finding $\pi(s)$ from $V^{\pi}(s)$

↳ Policy Iteration ~~Policy Evaluation~~