

Dynamic Games

- 2 Player zero sum turn taking games
- Markov Games
- Extensive Form Games

Turn-taking Games

Turn-taking Games



Terminology

Turn-taking Games



Terminology

- Max and Min players

Turn-taking Games



Terminology

- Max and Min players
- deterministic

Turn-taking Games



Terminology

- Max and Min players
- deterministic
- two player

Turn-taking Games



Terminology

- Max and Min players
- deterministic
- two player
- zero-sum

Turn-taking Games



Terminology

- Max and Min players
- deterministic
- two player
- zero-sum
- perfect information

Minimax Trees

Minimax Trees

MDP Expectimax Tree

Minimax Trees

MDP Expectimax Tree

Minimax Tree

Minimax Trees

MDP Expectimax Tree

$$V(s) = \max_{a \in \mathcal{A}} (R(s, a) + \mathbb{E}[V(s')])$$

Minimax Tree

Minimax Trees

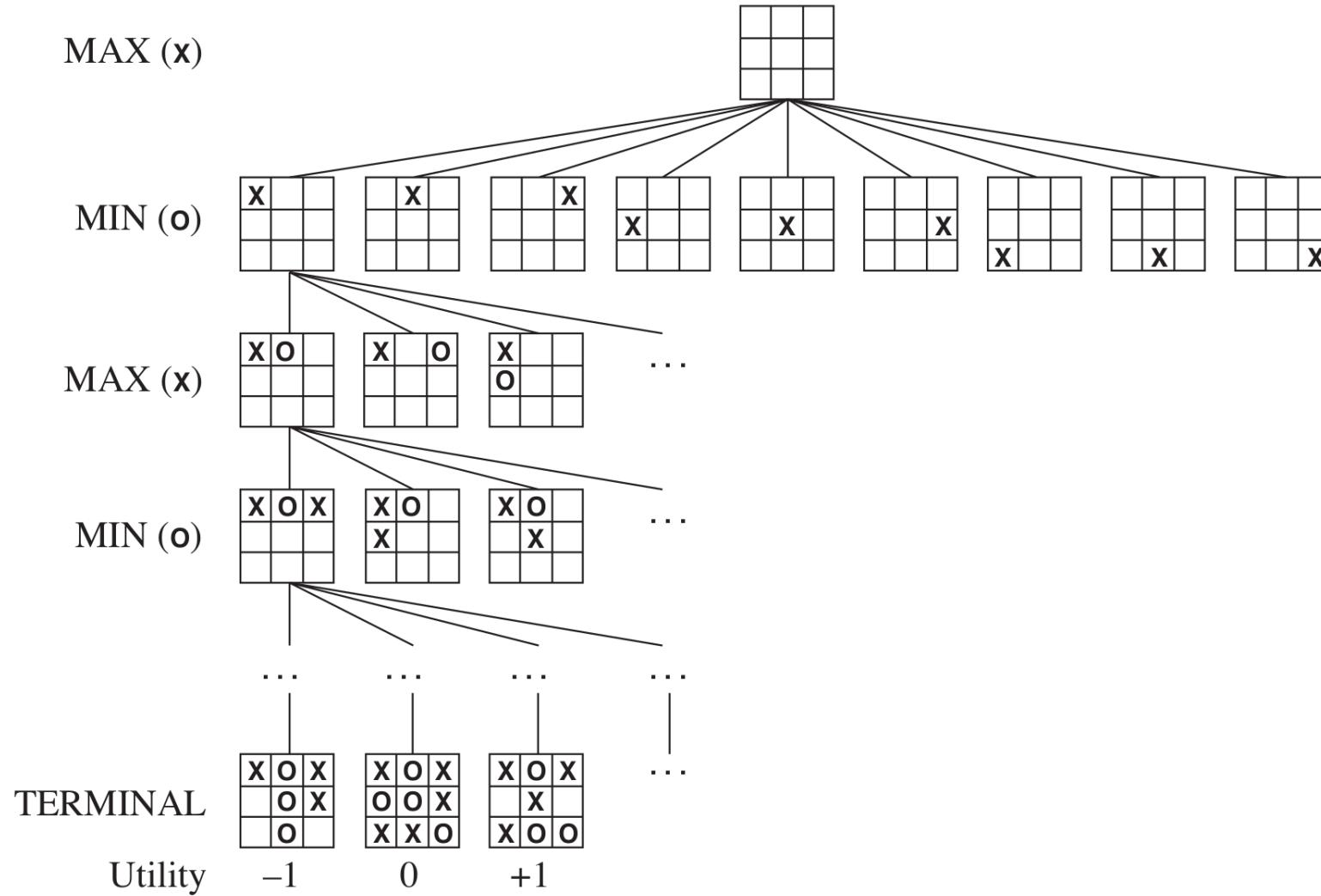
MDP Expectimax Tree

$$V(s) = \max_{a \in \mathcal{A}} (R(s, a) + \mathbb{E}[V(s')])$$

Minimax Tree

$$V(s) = \max_{a \in \mathcal{A}_1} (R(s, a) + \min_{a' \in \mathcal{A}_2} (R(s', a') + V(s'')))$$

Tic-Tac-Toe Example



Tree Backup Example

Tree Backup Example

```
function MINIMAX-DECISION(state) returns an action
    return  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(s, a))$ 
```

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow -\infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

```
function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow \infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

Tree Backup Example

```
function MINIMAX-DECISION(state) returns an action
    return  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(s, a))$ 
```

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow -\infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

```
function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow \infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

3 12 8 2 4 6 19 5 2

Tree Backup Example

```
function MINIMAX-DECISION(state) returns an action
    return  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(s, a))$ 
```

```
function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow -\infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

```
function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
     $v \leftarrow \infty$ 
    for each a in ACTIONS(state) do
         $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
    return v
```

3 12 8 2 4 6 19 5 2

Why is this harder than an MDP? (think
back to sparse sampling)

Alpha-Beta Pruning

```
function ALPHA-BETA-SEARCH(state) returns an action
  v  $\leftarrow$  MAX-VALUE(state,  $-\infty$ ,  $+\infty$ )
  return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v  $\leftarrow -\infty$ 
  for each a in ACTIONS(state) do
    v  $\leftarrow$  MAX(v, MIN-VALUE(RESULT(s,a),  $\alpha$ ,  $\beta$ ))
    if v  $\geq \beta$  then return v
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  return v
```

```
function MIN-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v  $\leftarrow +\infty$ 
  for each a in ACTIONS(state) do
    v  $\leftarrow$  MIN(v, MAX-VALUE(RESULT(s,a),  $\alpha$ ,  $\beta$ ))
    if v  $\leq \alpha$  then return v
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return v
```

Alpha-Beta Pruning

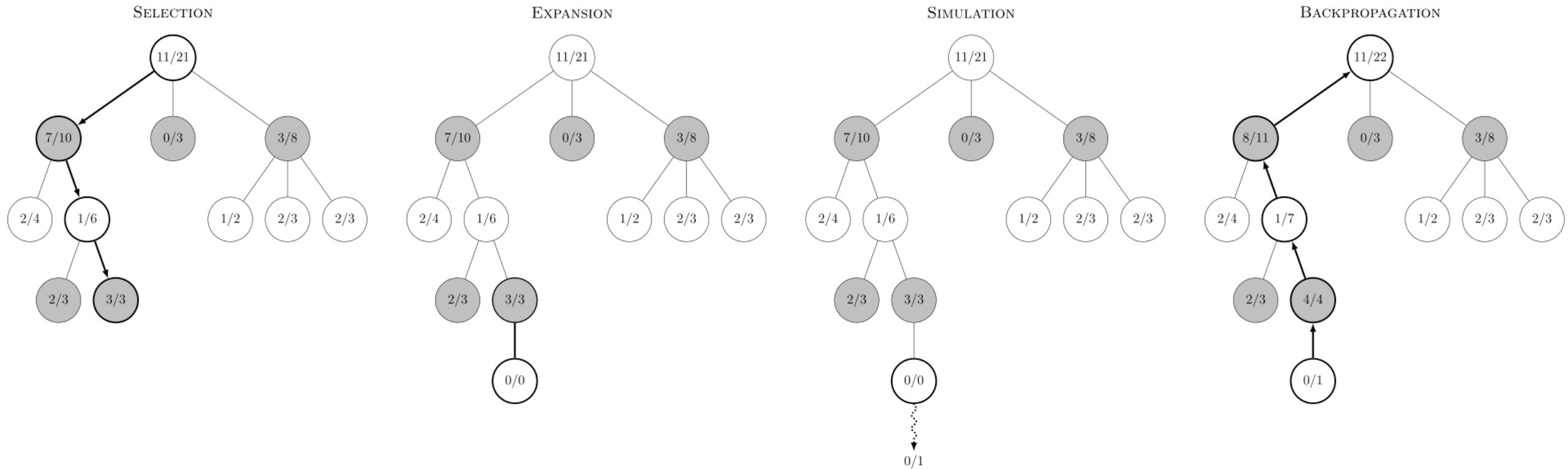
```
function ALPHA-BETA-SEARCH(state) returns an action
  v  $\leftarrow$  MAX-VALUE(state,  $-\infty$ ,  $+\infty$ )
  return the action in ACTIONS(state) with value v
```

```
function MAX-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v  $\leftarrow -\infty$ 
  for each a in ACTIONS(state) do
    v  $\leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$ 
    if v  $\geq \beta$  then return v
     $\alpha \leftarrow \text{MAX}(\alpha, v)$ 
  return v
```

```
function MIN-VALUE(state,  $\alpha$ ,  $\beta$ ) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v  $\leftarrow +\infty$ 
  for each a in ACTIONS(state) do
    v  $\leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$ 
    if v  $\leq \alpha$  then return v
     $\beta \leftarrow \text{MIN}(\beta, v)$ 
  return v
```

3 12 8 2 4 6 19 5 2

MCTS for Games



Note: the above example does not follow UCB exploration

Markov Games

Markov Game Definition

MDP: (S, A, T, R, γ)

MG : (I, S, A, T, R, γ)

"stochastic
game"

I: set of players $1..n$

S: state space

A: Joint action space

$A^1 \times A^2 \times \dots \times A^n$

$T(s'|s, a)$

\nwarrow vector \uparrow joint action

$R(s, a)$

\nwarrow joint action

$R^i(s, a)$

Turn taking

$R(s, a) = 0$ ~~at~~ except terminal

Transitions are deterministic
Only 1 player's action matters
in T

Markov Game Definition

$$(I, S, A, T, R, \gamma)$$

- I = set of players
- S = state space (combined for all players)
- A = *joint* action space
- T = transition distributions: $T(s' | s, a)$ is the distribution of s' given state s and *joint* action a
- R = joint reward function: $R^i(s, a)$ is the reward for agent i in state s when *joint* action a is taken

Nash equilibrium at every state

$$\underset{\boldsymbol{\pi}^U}{\text{minimize}} \quad \sum_{i \in \mathcal{I}} \sum_s \left(U^i(s) - Q^i(s, \boldsymbol{\pi}(s)) \right)$$

subject to $U^i(s) \geq Q^i(s, a^i, \boldsymbol{\pi}^{-i}(s))$ for all i, s, a^i

$$\sum_{a^i} \pi^i(a^i \mid s) = 1 \text{ for all } i, s$$

$$\pi^i(a^i \mid s) \geq 0 \text{ for all } i, s, a^i$$

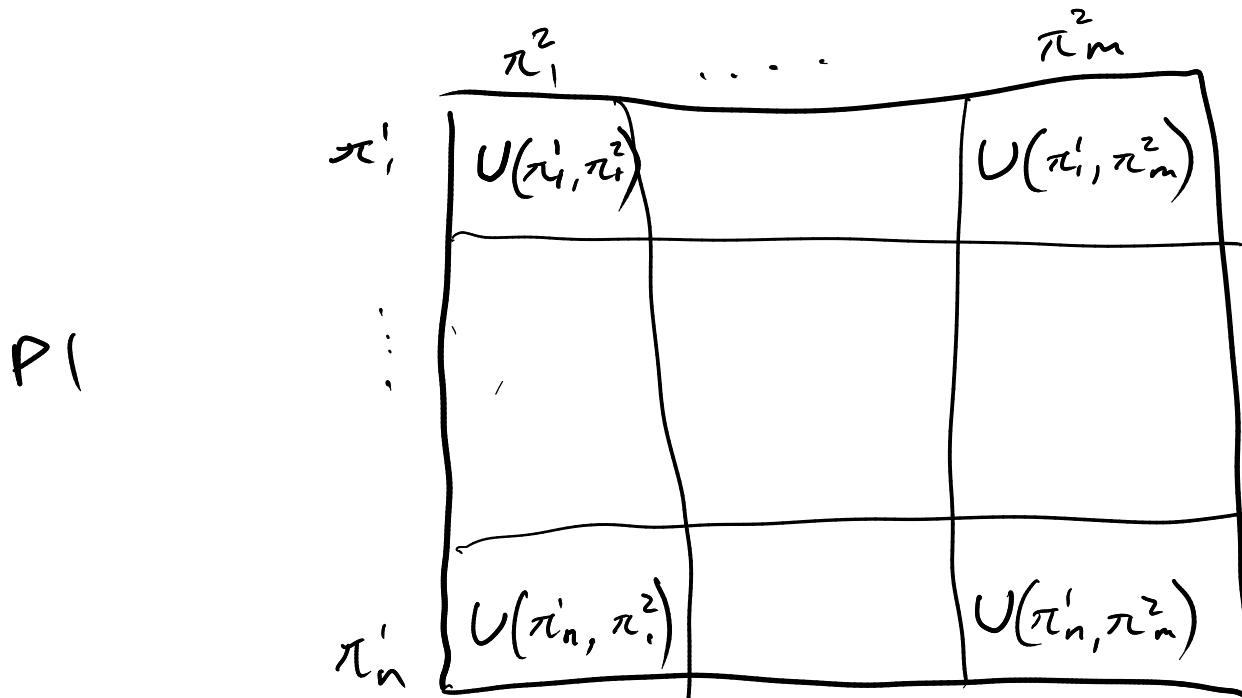
where

$$Q^i(s, \boldsymbol{\pi}(s)) = R^i(s, \boldsymbol{\pi}(s)) + \gamma \sum_{s'} T(s' \mid s, \boldsymbol{\pi}(s)) U^i(s')$$

Reduction to Simple Game

$|A|^{1s}$

P2



$$U(\pi^1, \pi^2)$$

$$= E \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]$$

$$\uparrow P(a_t | s_t) = \prod_{i \in I} \pi_i^i(a_t^i | s_t)$$

Best response in a Markov Game

Given MG (I, S, A, T, R, r) and π^{-i} , calculate $\max_{\pi^i} U^i(\pi^i, \pi^{-i})$.

$(MG + \pi^{-i} = MDP \text{ for agent } i)$

1) form MDP called $M' = (S, A^i, T', R', \gamma)$

$$T'(s'|s, a^i) = T(s'|s, a^i, \pi^{-i}(s))$$

$$R'(s, a^i) = \sum_{a^{-i}} R^i(s, a^i, a^{-i}) \pi^i(a^{-i}|s)$$

2) solve M'

Fictitious Play in a Markov Game

initialize $N(j, a^j, s)$ to 0

initialize π^{BR}

Loop:

1. Simulate MG with π^{BR}

2. Update $N(j, a^j, s) \leftarrow N$ should reflect all past simulations

3. $\pi^j(a^j | s) \propto N(j, a^j, s) \quad \forall j$

4. $\pi^{BR} \leftarrow$ best response to π

Markov Game Recap

1. Definition (I, S, A, T, R, γ)
2. Reduction to simple game
3. Computing a best response means solving
an MDP - know how to construct the MDP

Incomplete Information



Partially Observable Markov Game

MG: (I, S, A, T, R, γ)

POMG: $(I, S, A, T, O, Z, R, \gamma)$

"P.O.S.G"

$$\begin{array}{c} \text{↑} \quad \text{↑} \\ O^1 \times O^2 \dots O^n \end{array} \curvearrowright Z^i(O^i | a, s')$$

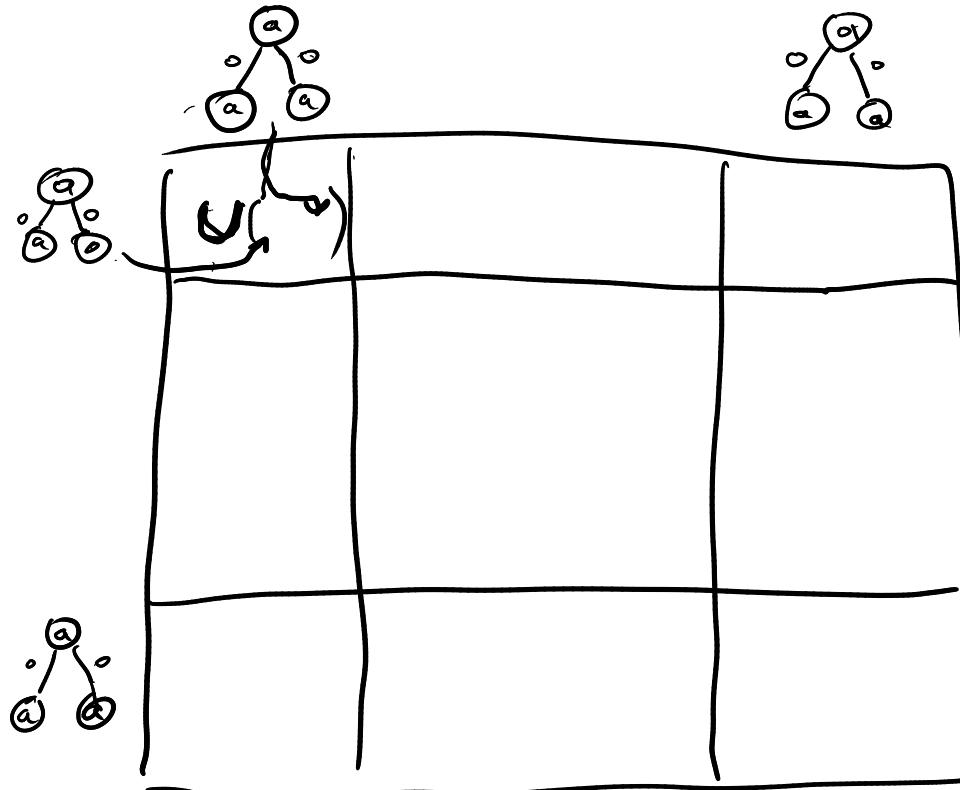
Belief updates?

$$b'(s') \propto Z(o|a, s') \sum_s T(s'|s, a) b(s)$$

$$b''(s') \propto Z''(o'|a^i, \pi^i) \quad \text{either } b'$$

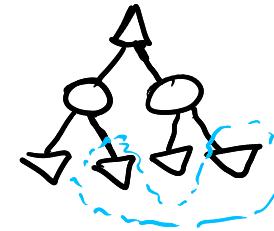
Beliefs in POMDP terms are generally too difficult to calculate in POMG

Reduction to Simple Game



Extensive Form Game

(Alternative to POMGs that is more common in the literature)



- Similar to a minimax tree for a turn-taking game
- Chance nodes
- Information sets ←

King-Ace Poker Example

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card

King-Ace Poker Example

- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point

King-Ace Poker Example

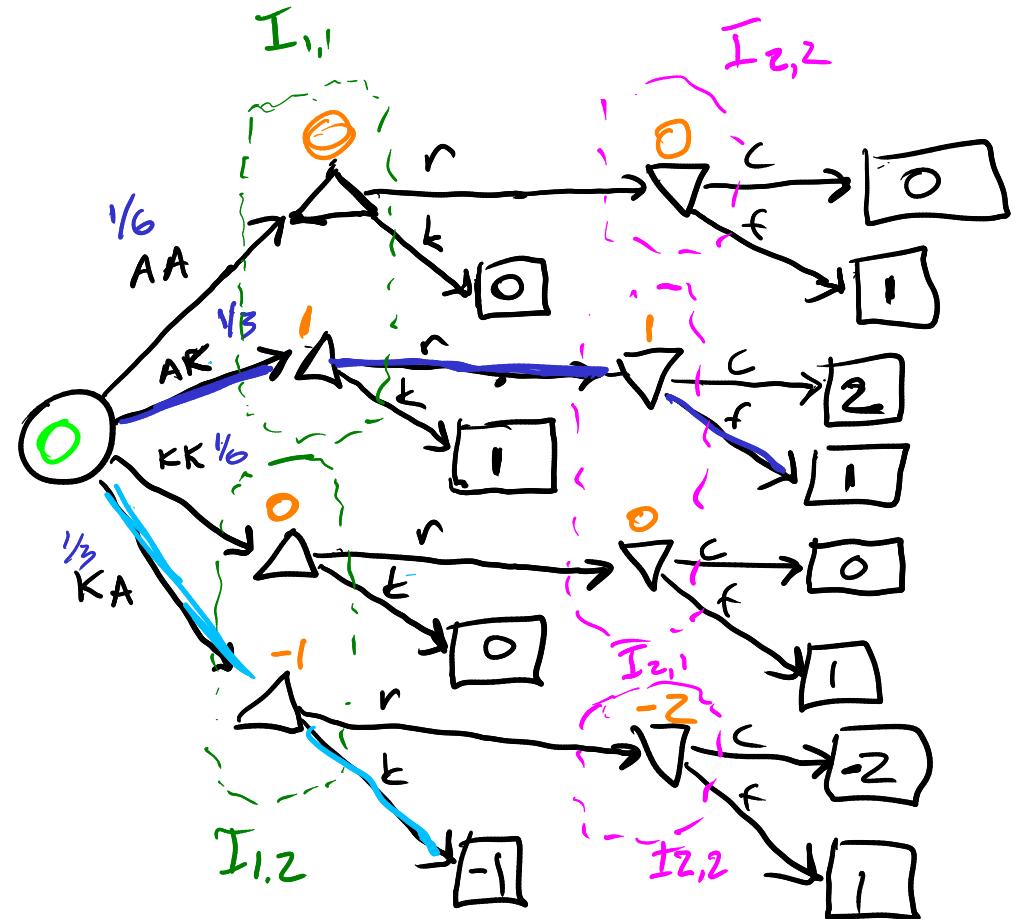
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point
- If P1 raises, P2 can either *call* (c) Player 1's bet, or *fold* (f) the payoff back to 1 point

King-Ace Poker Example

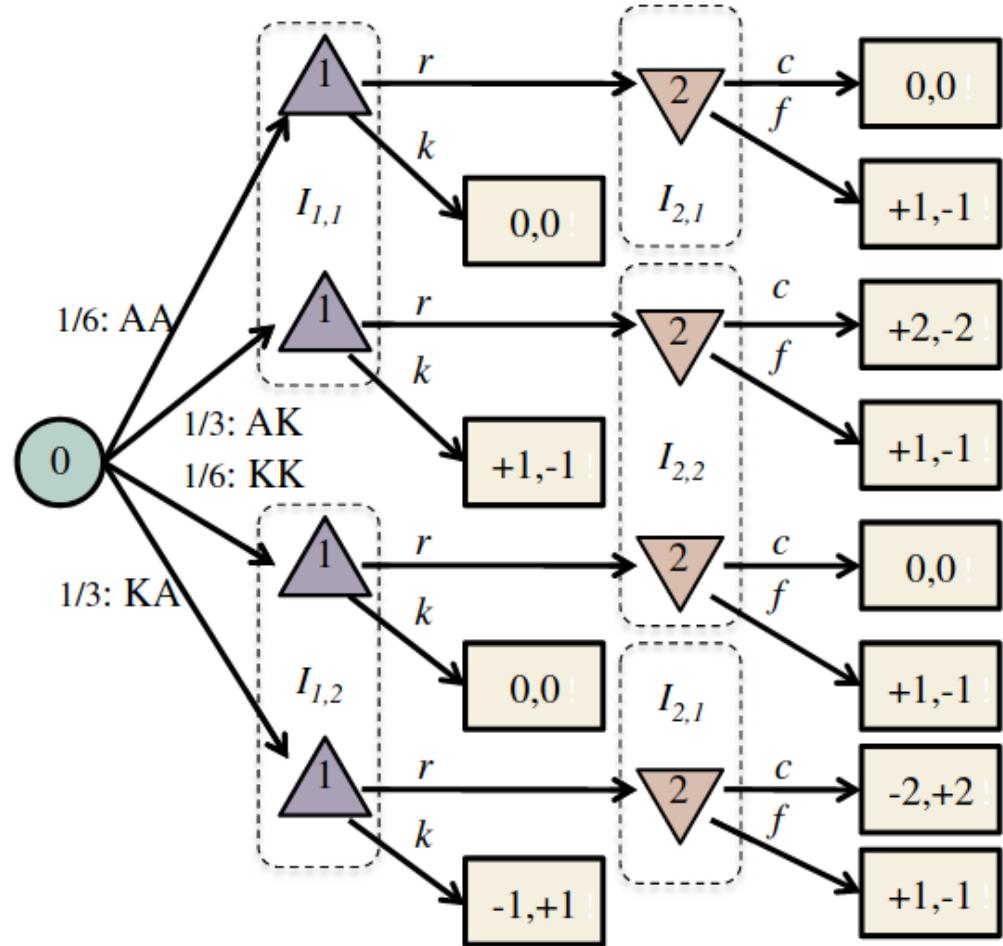
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* (r) the payoff to 2 points or *check* (k) the payoff at 1 point
- If P1 raises, P2 can either *call* (c) Player 1's bet, or *fold* (f) the payoff back to 1 point
- The highest card wins

Nathan
111

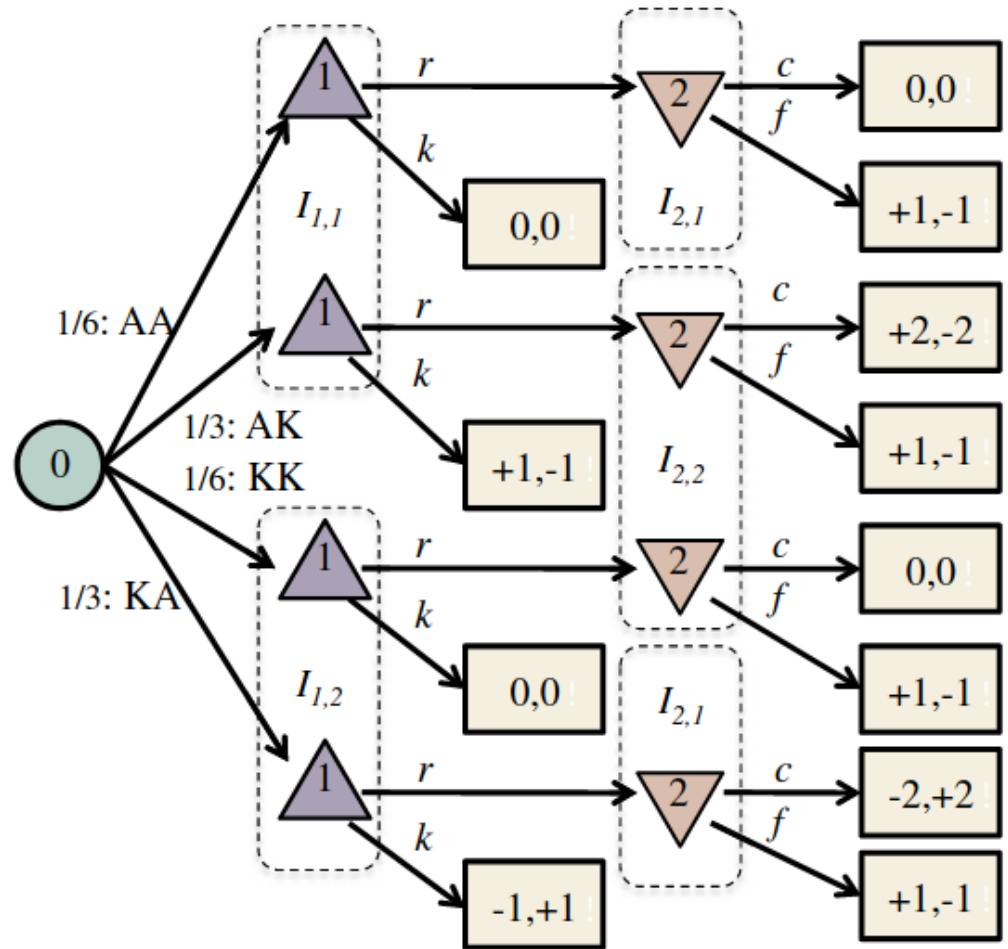
Ryan
1



Extensive to Matrix Form

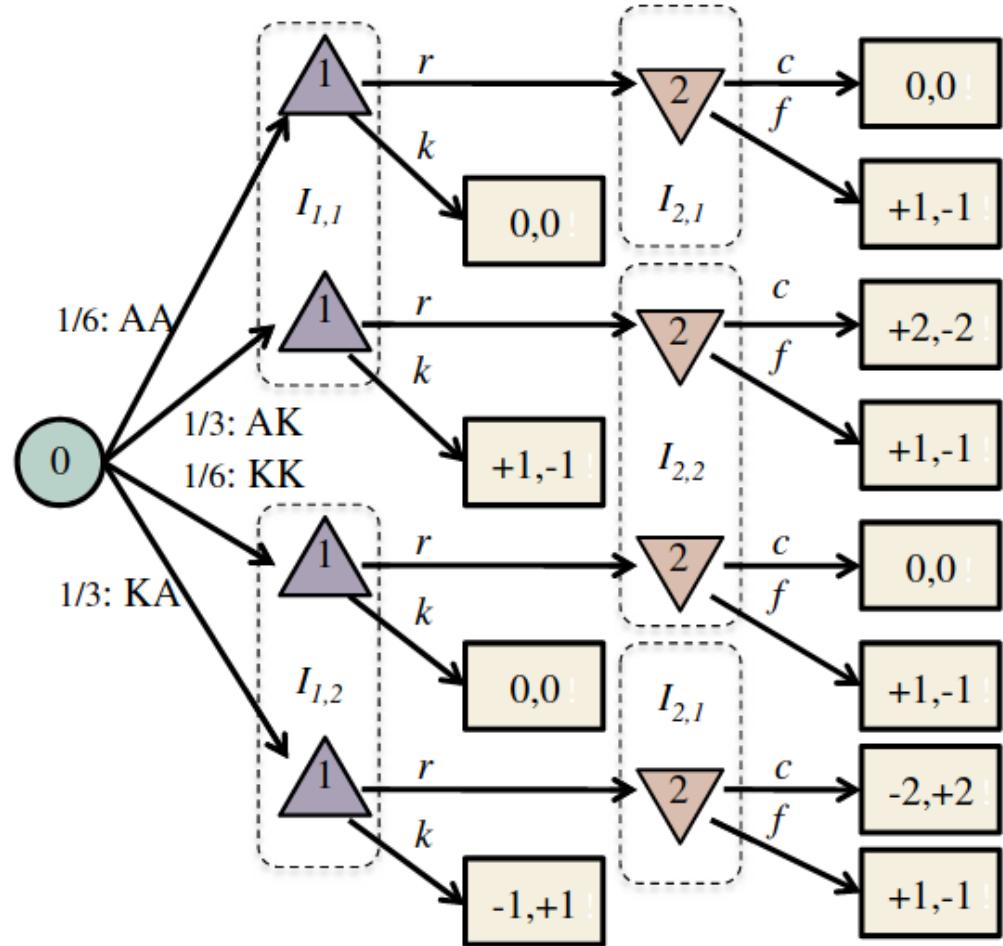


Extensive to Matrix Form



	2:cc	2:cf	2:ff	2;fc
1:rr	0	-1/6	1	7/6
1:kr	-1/3	-1/6	5/6	2/3
1:rk	1/3	0	1/6	1/2
1:kk	0	0	0	0

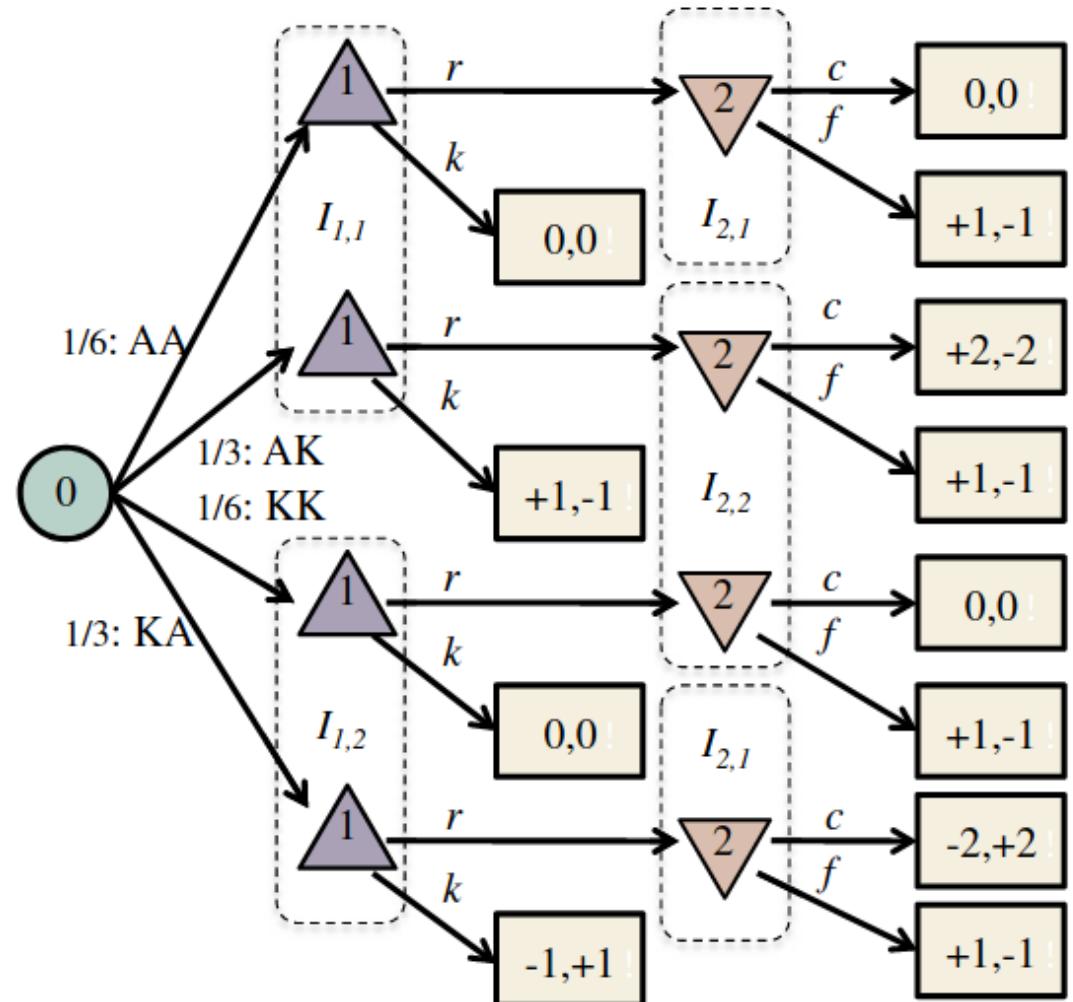
Extensive to Matrix Form



	2:cc	2:cf	2:ff	2:fc
1:rr	0	-1/6	1	7/6
1:kr	-1/3	-1/6	5/6	2/3
1:rk	1/3	0	1/6	1/2
1:kk	0	0	0	0

Exponential in number of info states!

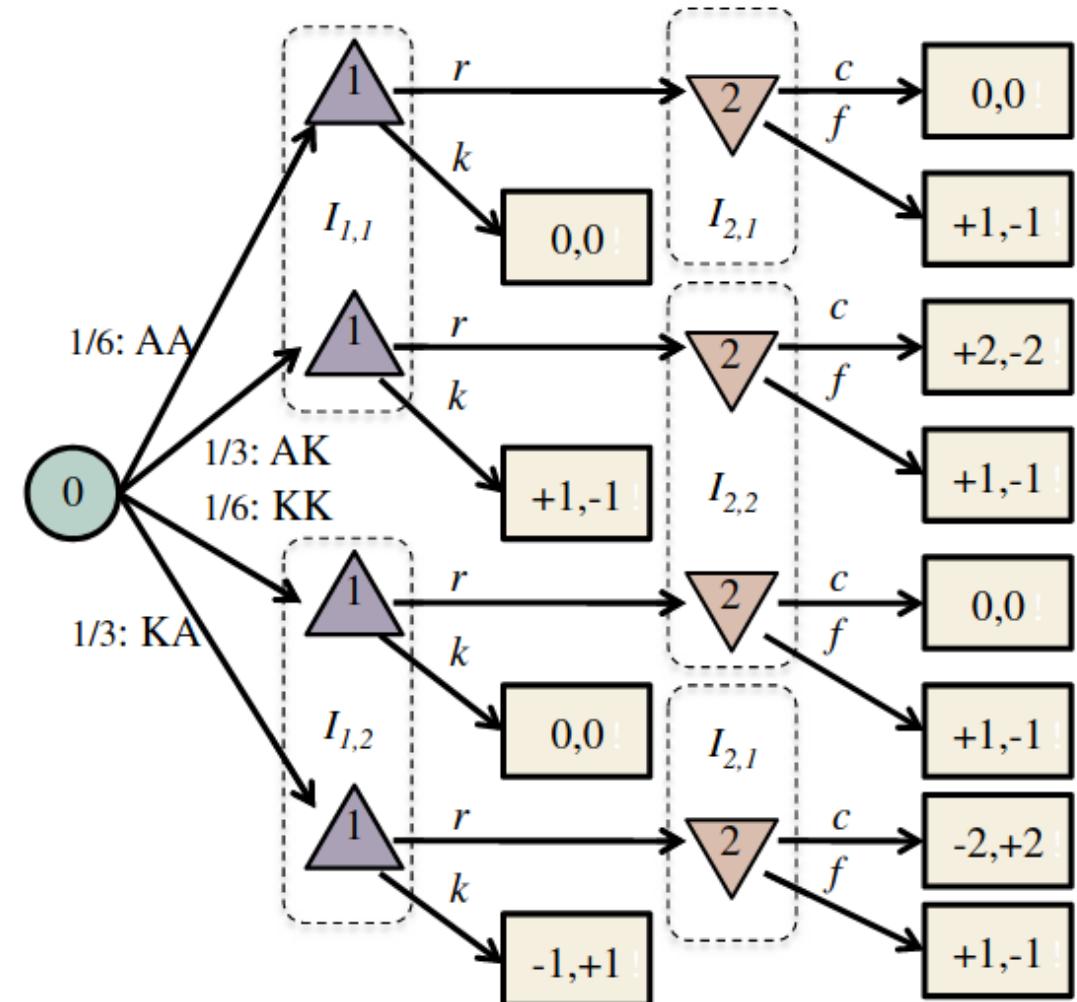
Counterfactual Regret Minimization



This slide not covered on exam

Counterfactual Regret Minimization

- 1: Initialize cumulative regret tables: $\forall I, r_I[a] \leftarrow 0$.
- 2: Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
- 3: Initialize initial profile: $\sigma^1(I, a) \leftarrow 1/|A(I)|$

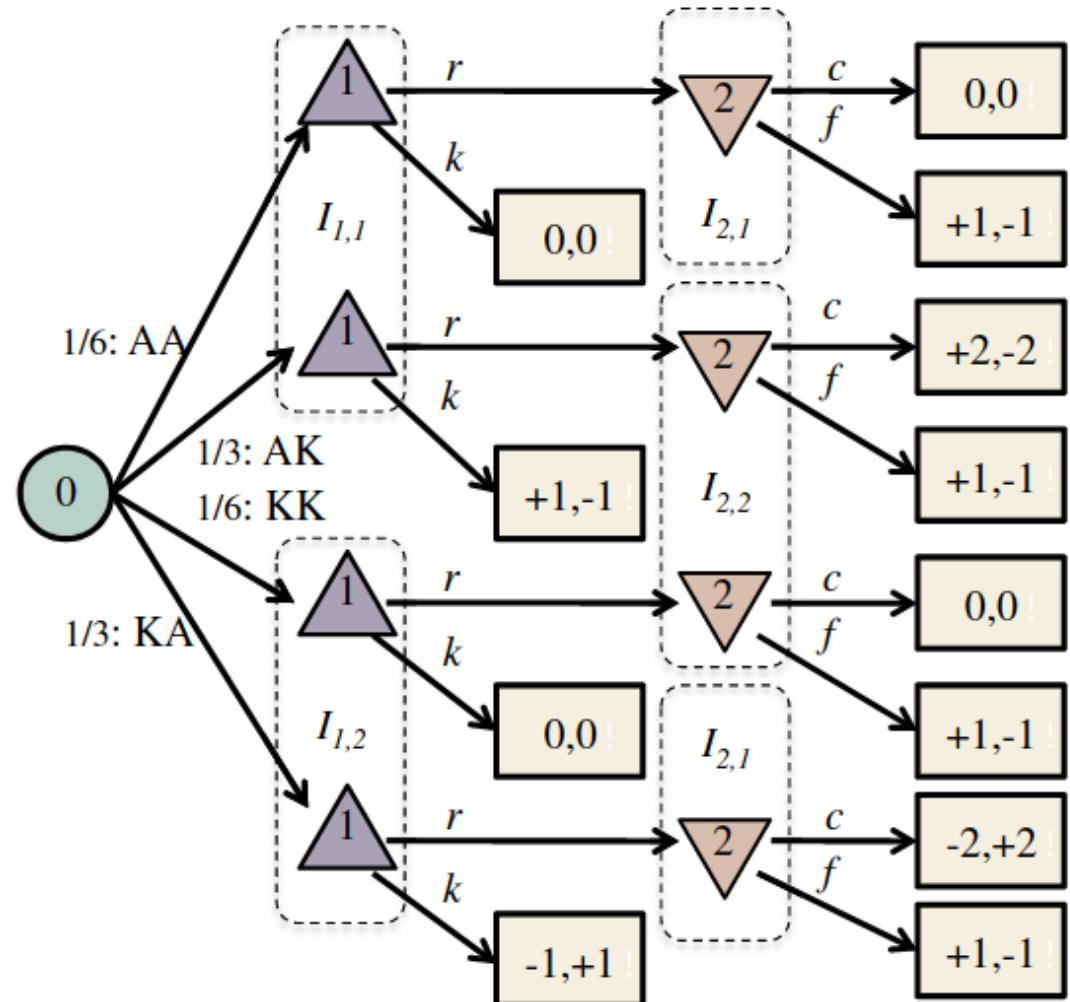


This slide not covered on exam

Counterfactual Regret Minimization

- 1: Initialize cumulative regret tables: $\forall I, r_I[a] \leftarrow 0$.
- 2: Initialize cumulative strategy tables: $\forall I, s_I[a] \leftarrow 0$.
- 3: Initialize initial profile: $\sigma^1(I, a) \leftarrow 1/|A(I)|$

```
32: function Solve():
33:   for  $t = \{1, 2, 3, \dots, T\}$  do
34:     for  $i \in \{1, 2\}$  do
35:       CFR( $\emptyset, i, t, 1, 1$ )
36:     end for
37:   end for
```



This slide not covered on exam

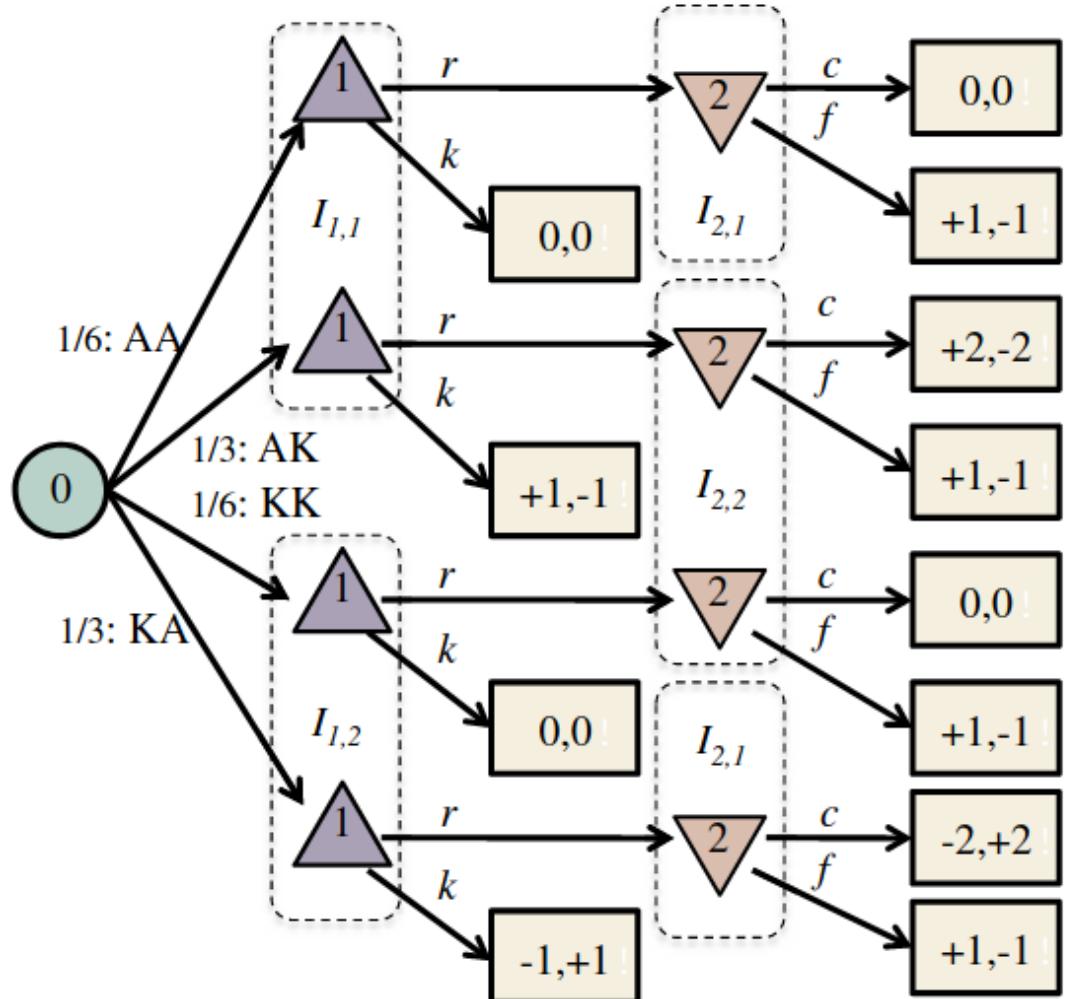
Counterfactual Regret Minimization

```

1: Initialize cumulative regret tables:  $\forall I, r_I[a] \leftarrow 0$ .
2: Initialize cumulative strategy tables:  $\forall I, s_I[a] \leftarrow 0$ .
3: Initialize initial profile:  $\sigma^1(I, a) \leftarrow 1/|A(I)|$ 

5: function CFR( $h, i, t, \pi_1, \pi_2$ ):
6:   if  $h$  is terminal then
7:     return  $u_i(h)$ 
8:   else if  $h$  is a chance node then
9:     Sample a single outcome  $a \sim \sigma_c(h, a)$ 
10:    return CFR( $ha, i, t, \pi_1, \pi_2$ )
11:   end if
12:   Let  $I$  be the information set containing  $h$ .
13:    $v_\sigma \leftarrow 0$ 
14:    $v_{\sigma_{I \rightarrow a}}[a] \leftarrow 0$  for all  $a \in A(I)$ 
15:   for  $a \in A(I)$  do
16:     if  $P(h) = 1$  then
17:        $v_{\sigma_{I \rightarrow a}}[a] \leftarrow \text{CFR}(ha, i, t, \sigma^t(I, a) \cdot \pi_1, \pi_2)$ 
18:     else if  $P(h) = 2$  then
19:        $v_{\sigma_{I \rightarrow a}}[a] \leftarrow \text{CFR}(ha, i, t, \pi_1, \sigma^t(I, a) \cdot \pi_2)$ 
20:     end if
21:      $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{\sigma_{I \rightarrow a}}[a]$ 
22:   end for
23:   if  $P(h) = i$  then
24:     for  $a \in A(I)$  do
25:        $r_I[a] \leftarrow r_I[a] + \pi_{-i} \cdot (v_{\sigma_{I \rightarrow a}}[a] - v_\sigma)$ 
26:        $s_I[a] \leftarrow s_I[a] + \pi_i \cdot \sigma^t(I, a)$ 
27:     end for
28:      $\sigma^{t+1}(I) \leftarrow \text{regret-matching values}$ 
29:   end if
30:   return  $v_\sigma$ 

```



This slide not covered on exam