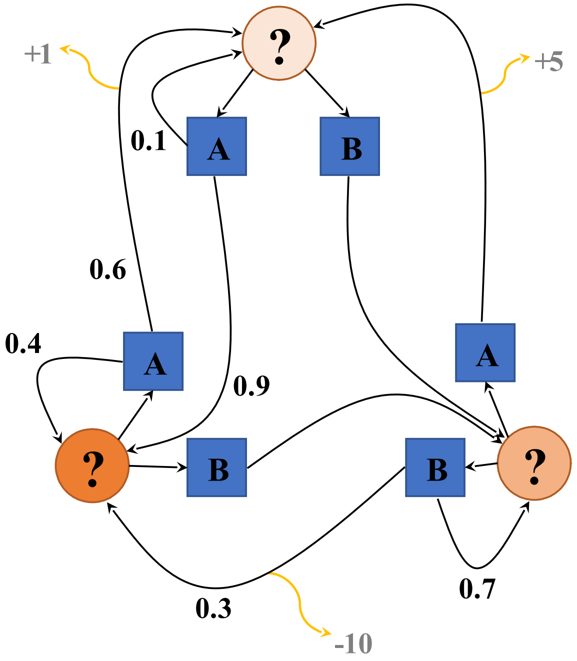
# Incomplete Information Dynamic Games

# Incomplete Information

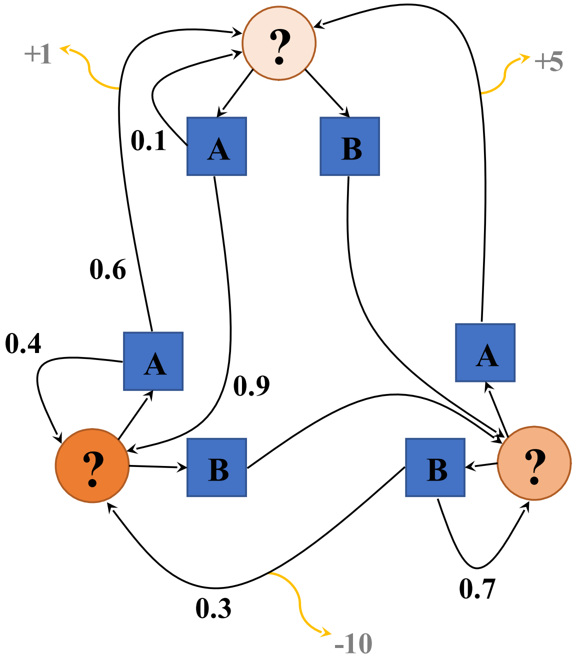
# Partially Observable Markov Decision Process (POMDP)




- $\mathcal{S}$ - State space
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ - Transition probability distribution
- $\mathcal{A}$ - Action space
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - Reward

**Alleatory**

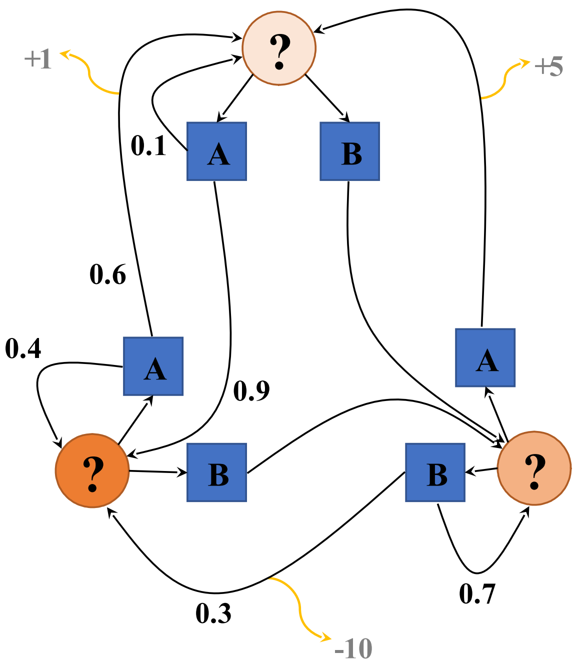
# Partially Observable Markov Decision Process (POMDP)




- $\mathcal{S}$ - State space
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ - Transition probability distribution
- $\mathcal{A}$ - Action space
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - Reward
- $\mathcal{O}$ - Observation space

**Alleatory**

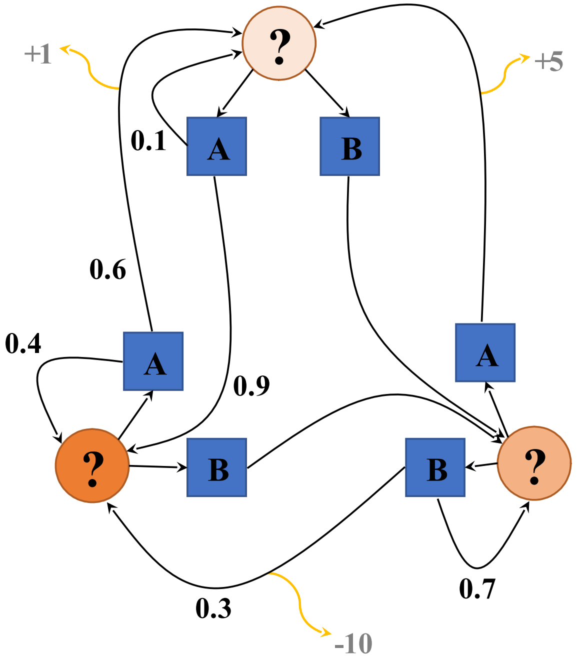
# Partially Observable Markov Decision Process (POMDP)




- $\mathcal{S}$ - State space
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}$ - Transition probability distribution
- $\mathcal{A}$ - Action space
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ - Reward
- $\mathcal{O}$ - Observation space
- $Z : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{O} \rightarrow \mathbb{R}$ - Observation probability distribution

**Alleatory**

# Partially Observable Markov Decision Process (POMDP)




- $\mathcal{S}$ - State space
- $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ - Transition probability distribution
- $\mathcal{A}$ - Action space
- $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ - Reward
- $\mathcal{O}$ - Observation space
- $Z : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{O} \to \mathbb{R}$ - Observation probability distribution

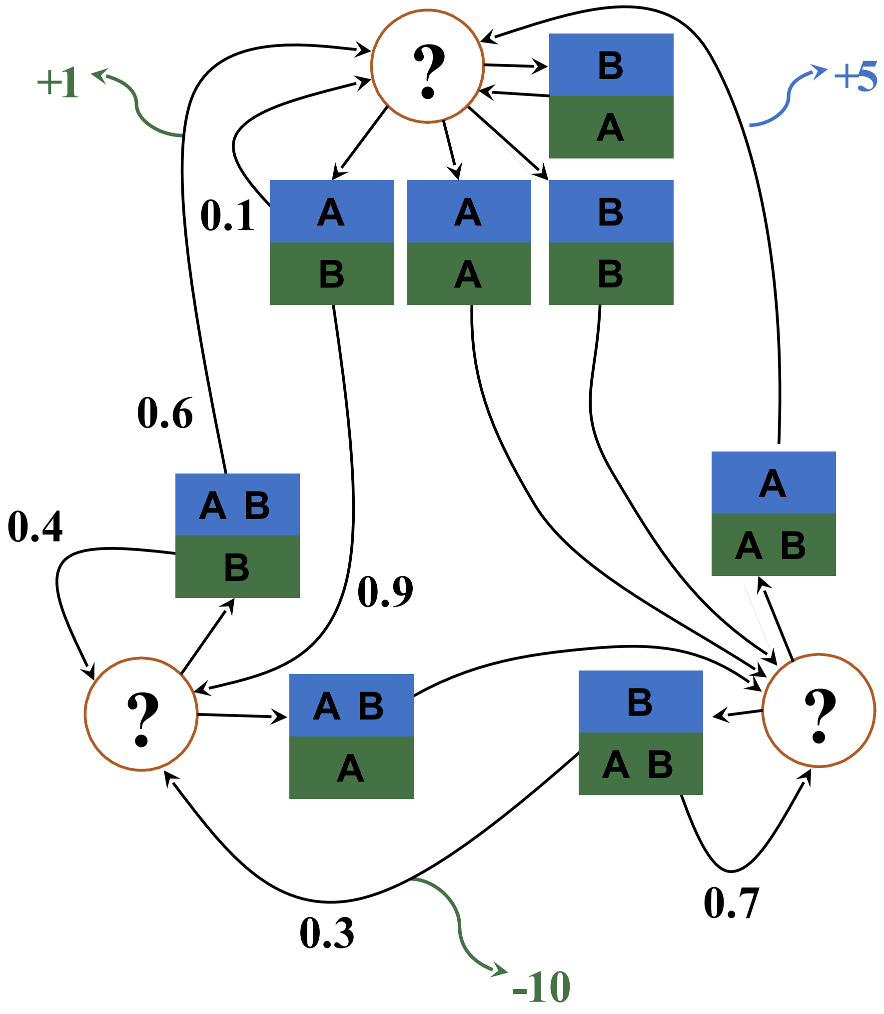
Alleatory

Epistemic (Static)

Epistemic (Dynamic)

# Partially Observable Markov Game

Stochastic




- $\mathcal{S}$ - State space
- joint action
- $T(s' \mid s, \boldsymbol{a})$ - Transition probability distribution
- $\mathcal{A}^i,\ i \in 1..k$ - Action spaces
- $R^i(s, \boldsymbol{a})$ - Reward function
- $\mathcal{O}^i,\ i \in 1..k$ - Observation space
- $Z(o^i \mid \boldsymbol{a}, s')$ - Observation probability distribution

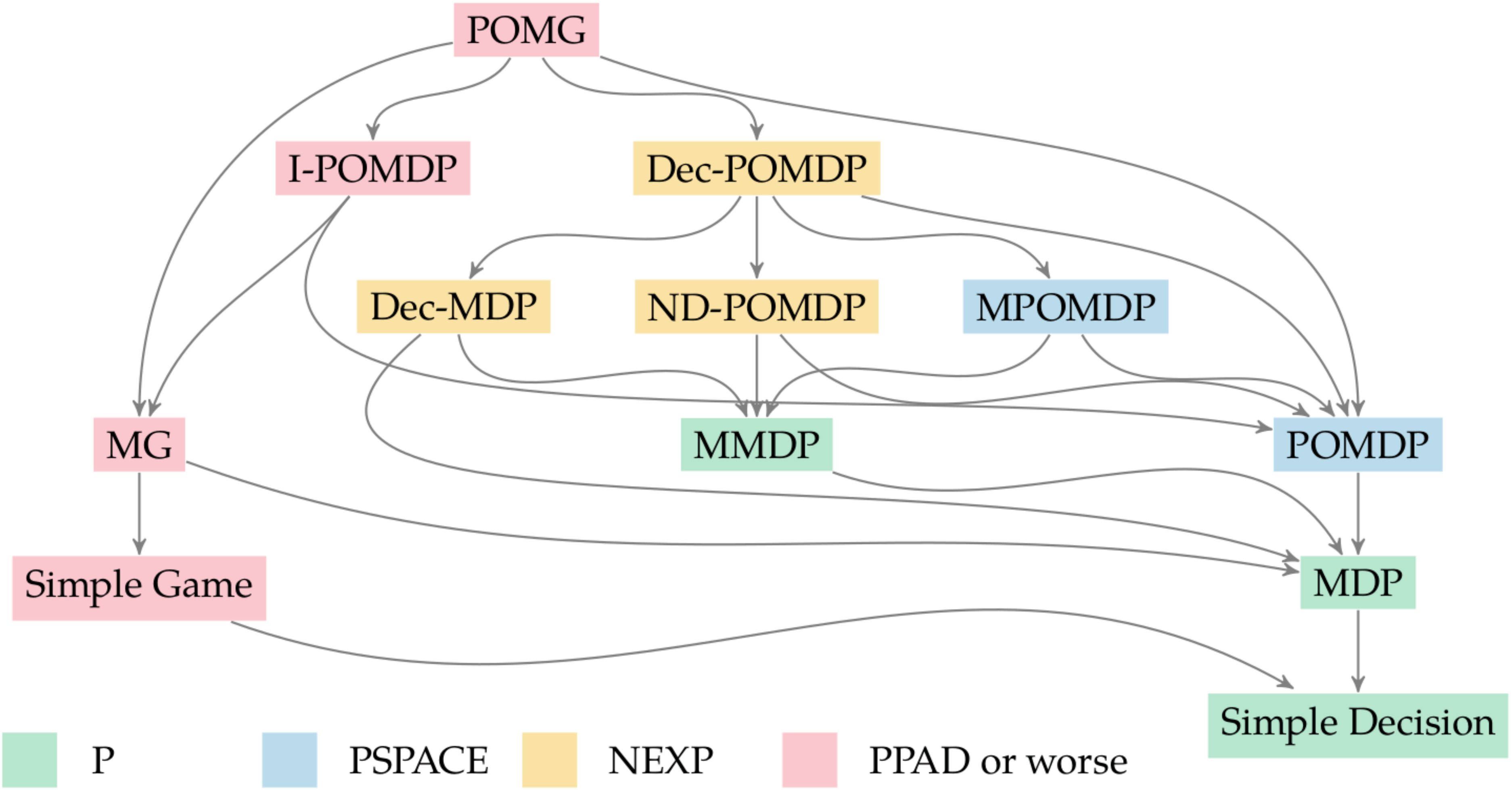
Alleatory

Epistemic (Static)

Epistemic (Dynamic)

Interaction

# Hierarchy of Problems



# Belief updates?

POMDP : $b'(s') \propto Z(o \mid a, s') \sum_{s \in S} T(s' \mid s, a)\, b(s)$

POMG : $b'(s') \propto Z(o^i \mid \underset{\text{joint action}}{\vec{a}}, s') \sum_{s \in S} T(s' \mid s, \underset{\text{joint action}}{\vec{a}})\, b(s)$ ✗

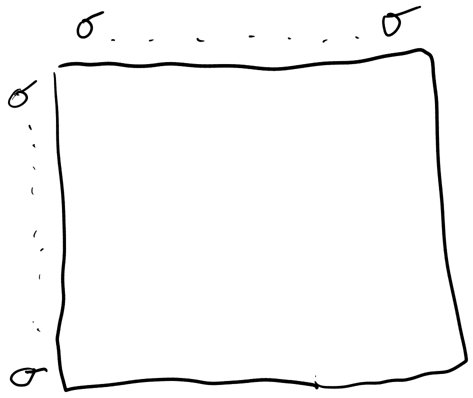
$\pi^{-i}$

Problems

1. Requires reasoning about other player's actions
2. Requires reasoning about other player's observations
3. Usually trying to solve for $\pi^{-i}$ at the same time as choosing our actions

# Reduction to Simple Game

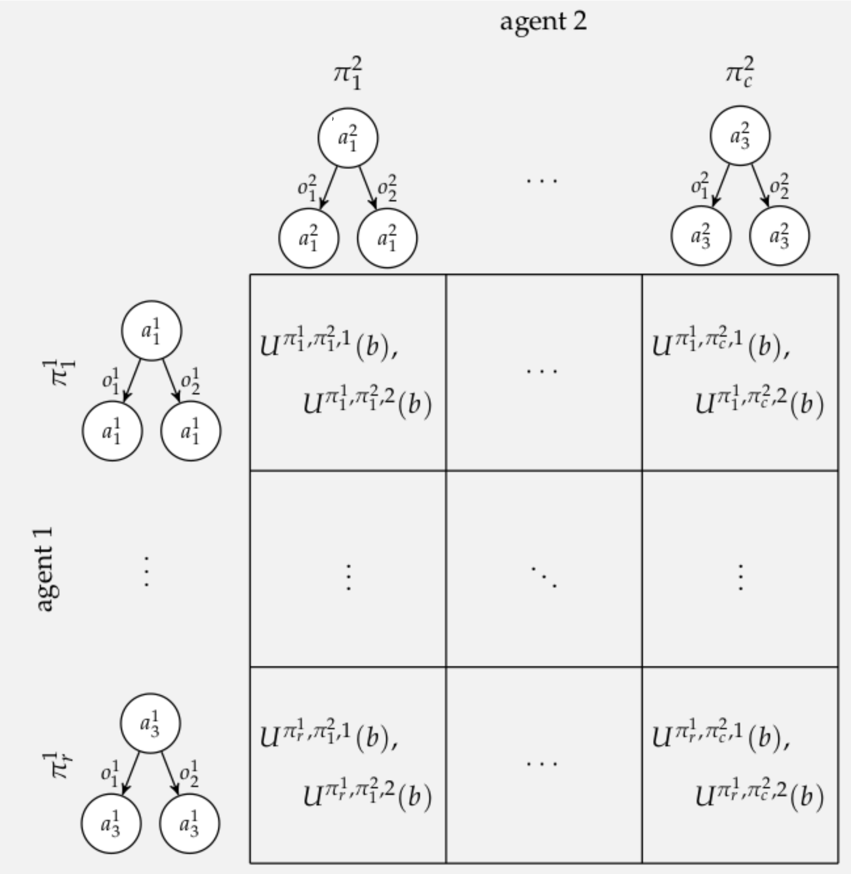
MDP



# Reduction to Simple Game

# Reduction to Simple Game

2 step



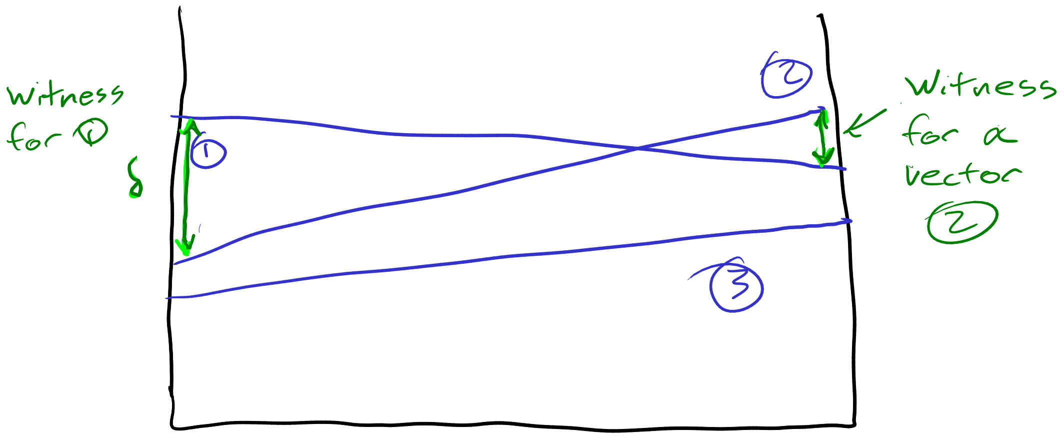
# Pruning in Dynamic Programming

# Pruning in Dynamic Programming

$$\sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) U^{\pi^{i\prime}, \boldsymbol{\pi}^{-i}, i}(s) \geq \sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) U^{\pi^{i}, \boldsymbol{\pi}^{-i}, i}(s)$$

# Pruning in Dynamic Programming

From POMDP




Pruning

If there exists $\pi^{i\prime}$ such that

$$\sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) U^{\pi^{i\prime}, \boldsymbol{\pi}^{-i}, i}(s) \geq \sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) U^{\pi^{i}, \boldsymbol{\pi}^{-i}, i}(s)$$

for all "beliefs", we can prune $\pi^i$.

$$\begin{aligned}
\underset{\delta, b}{\text{maximize}} \quad & \delta \\
\text{subject to} \quad & b(\boldsymbol{\pi}^{-i}, s) \geq 0 \text{ for all } \boldsymbol{\pi}^{-i}, s \\
& \sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) = 1 \\
& \sum_{\boldsymbol{\pi}^{-i}} \sum_{s} b(\boldsymbol{\pi}^{-i}, s) \left( U^{\pi^{i\prime}, \boldsymbol{\pi}^{-i}, i}(s) - U^{\pi^{i}, \boldsymbol{\pi}^{-i}, i}(s) \right) \geq \delta \text{ for all } \pi^{i\prime}
\end{aligned}$$

probability

Dynamic Programming for POMGs

Start with all possible N=1-step policies

Loop until $N = \vec{N}$

Evaluate N+1 step policies → U

Prune Dominated Policies

Solve Matrix Game for all $\bar{N}$-step policies

# 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

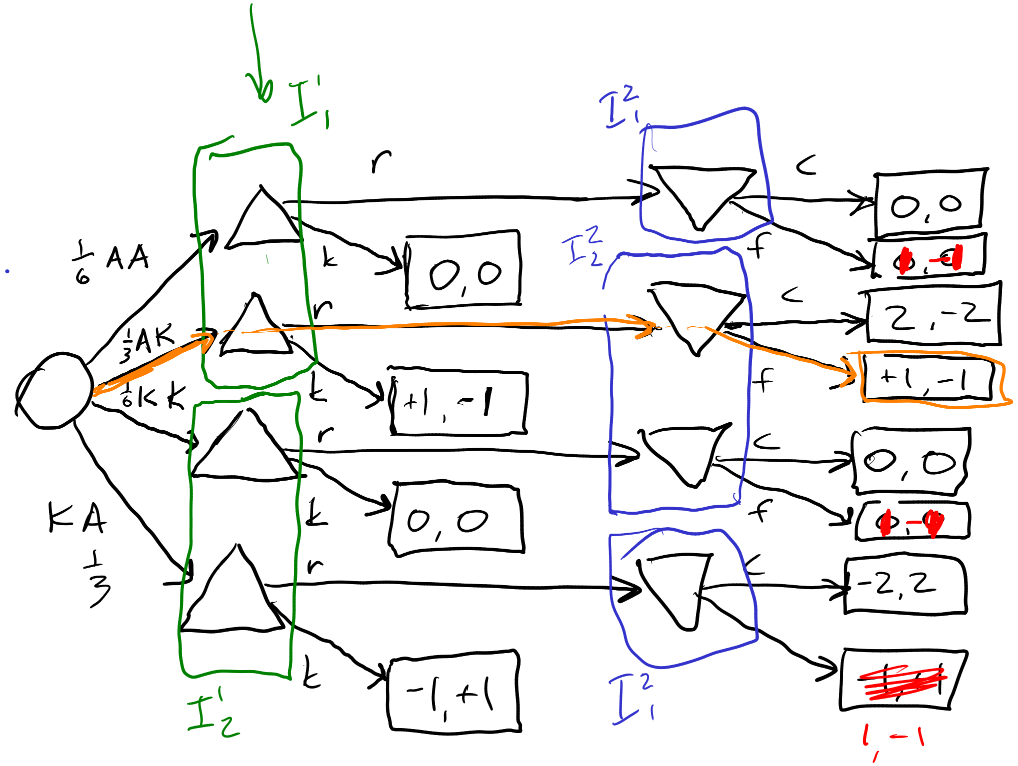
Aces High

r - raise
k - check

c - call
f - fold

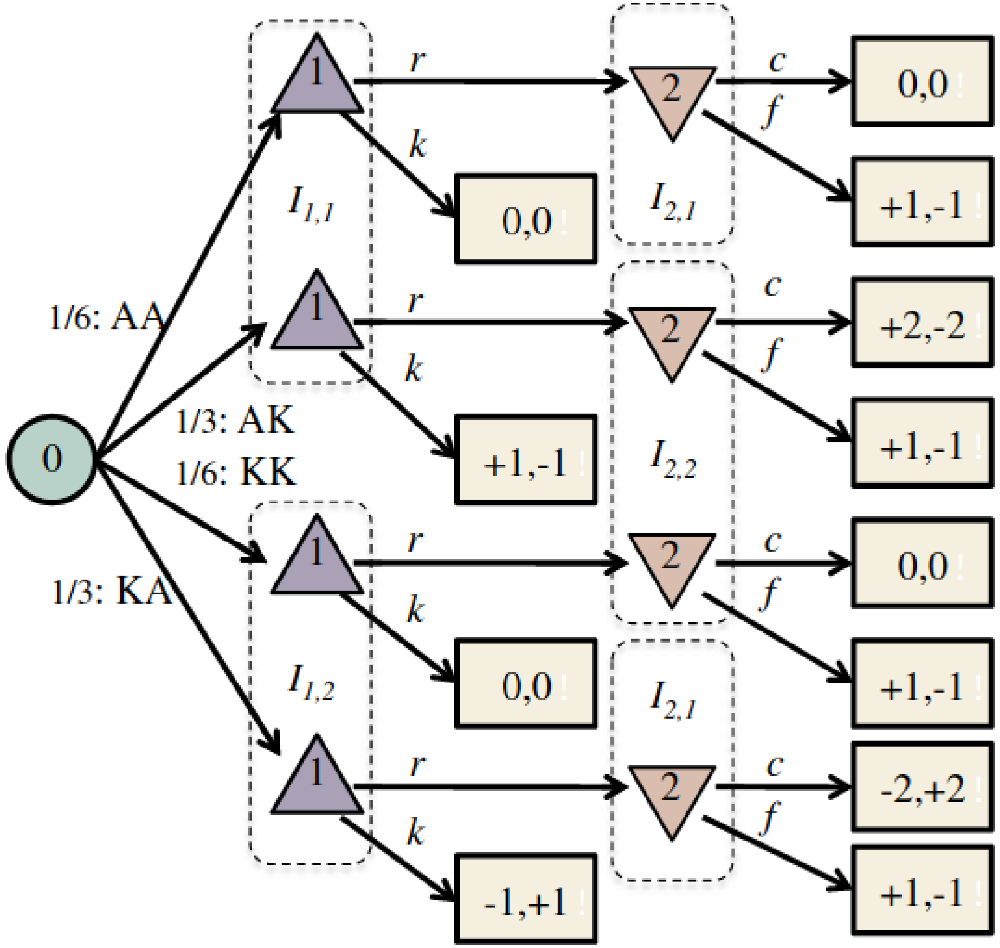
- 4 Cards: 2 Aces, 2 Kings
- Each player is dealt a card
- P1 can either *raise* ($r$) the payoff to 2 points or *check* ($k$) the payoff at 1 point
- If P1 raises, P2 can either *call* ($c$) Player 1's bet, or *fold* ($f$) ~~the payoff back to 1 point~~ Player 1 wins 1 point
- The highest card wins

Chance node

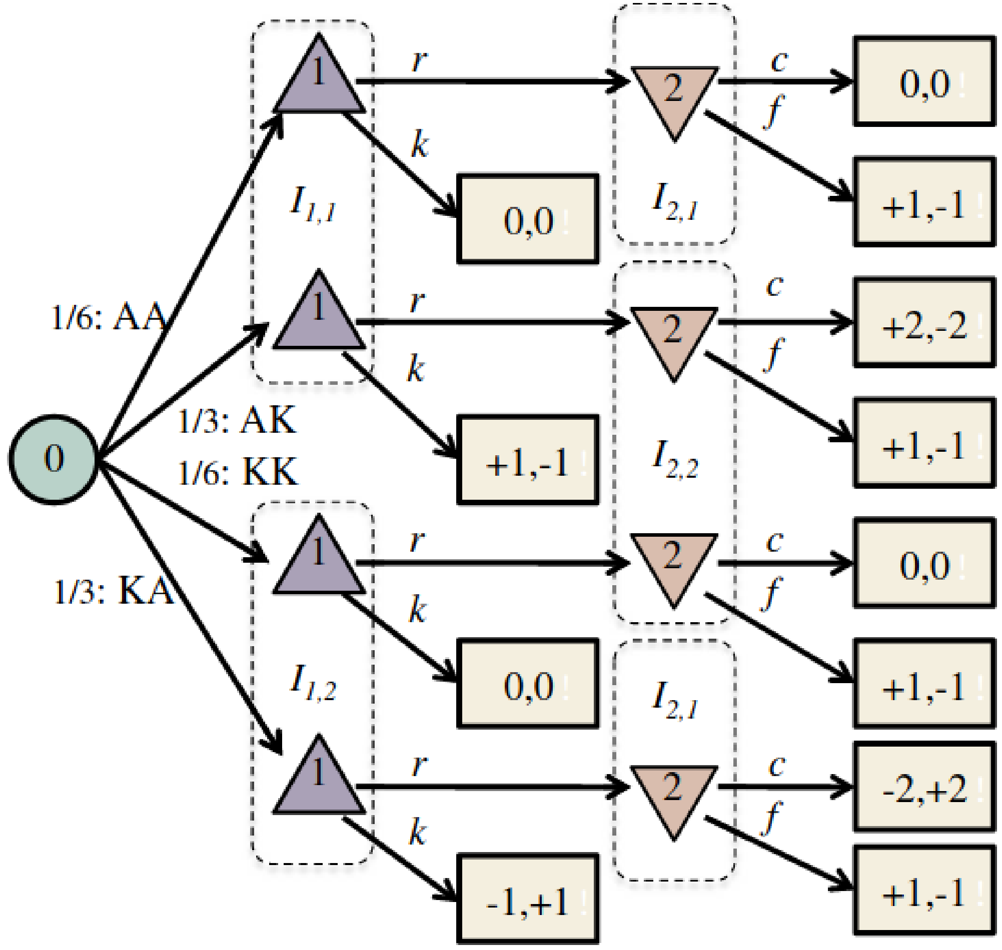



Information sets group nodes that are indistinguishable to a player

# Extensive to Matrix Form



# Extensive to Matrix Form



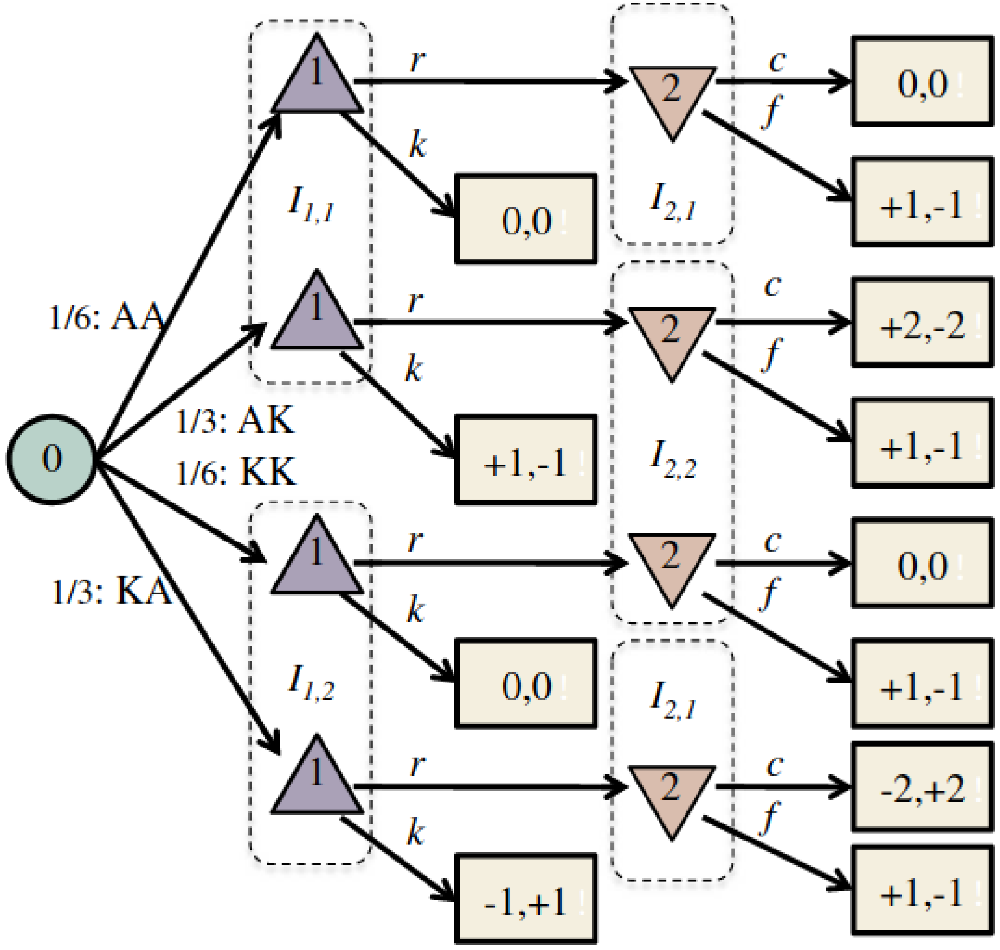

Arrows point to a best response in a row or column

Bars indicate responses are equally the best

If a cell has no arrows leaving, it is a pure NE

| | 2:cc | AK 2:cf | AK 2:ff | AK 2:fc |
|---|---|---|---|---|
| AK 1:rr | 0 | -1/6 | 1 | 7/6 |
| AK 1:kr | -1/3 | -1/6 | 5/6 | 2/3 |
| AK 1:rk | 1/3 | $\boxed{0}$ | 1/6 | 1/2 |
| AK 1:kk | 0 | 0 | 0 | 0 |

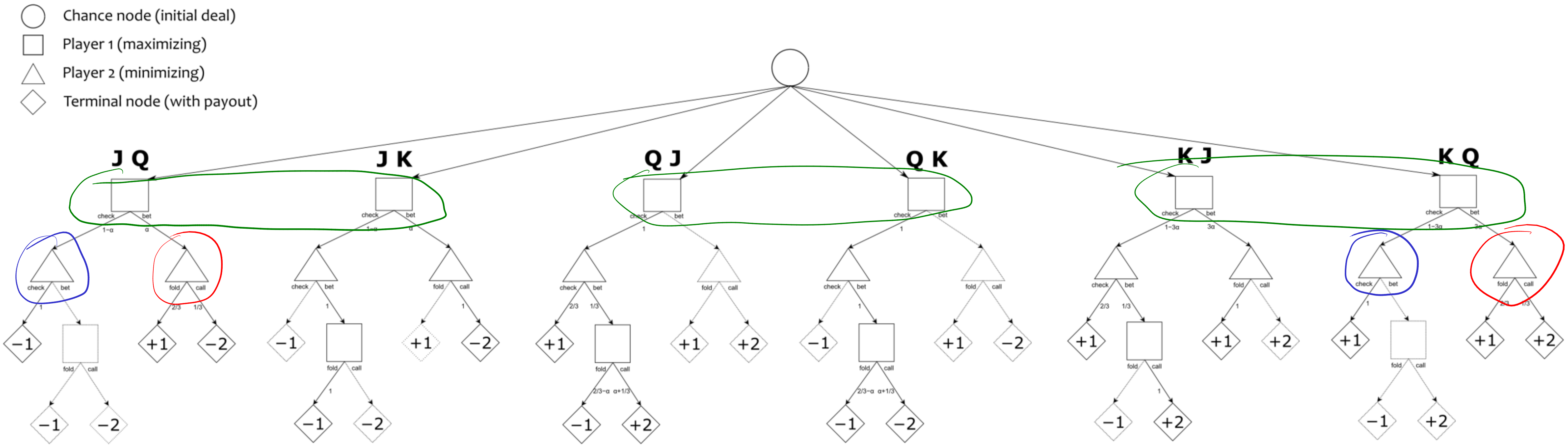
# Extensive to Matrix Form




| | 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!

# A more interesting example: Kuhn Poker

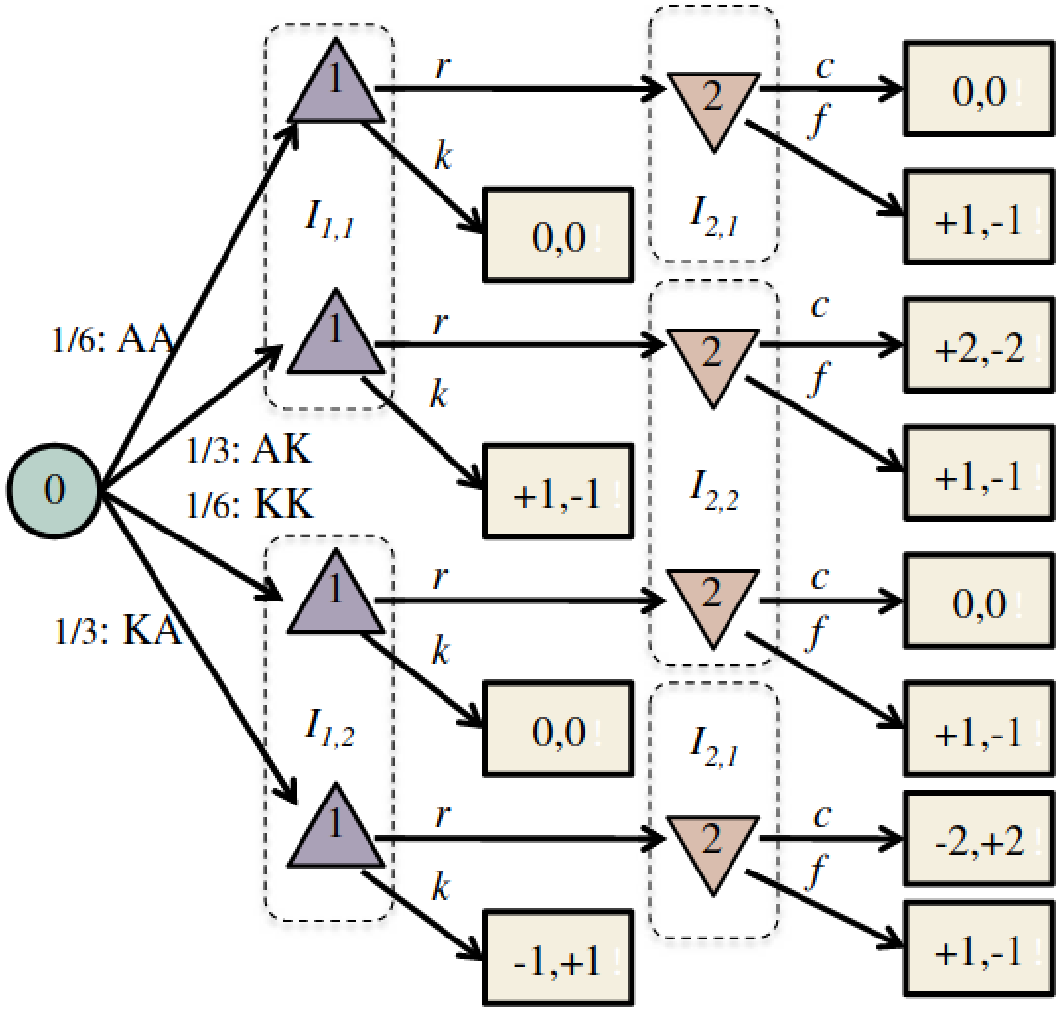


# Fictitious Play in Extensive Form Games

**Algorithm 2** General Fictitious Self-Play

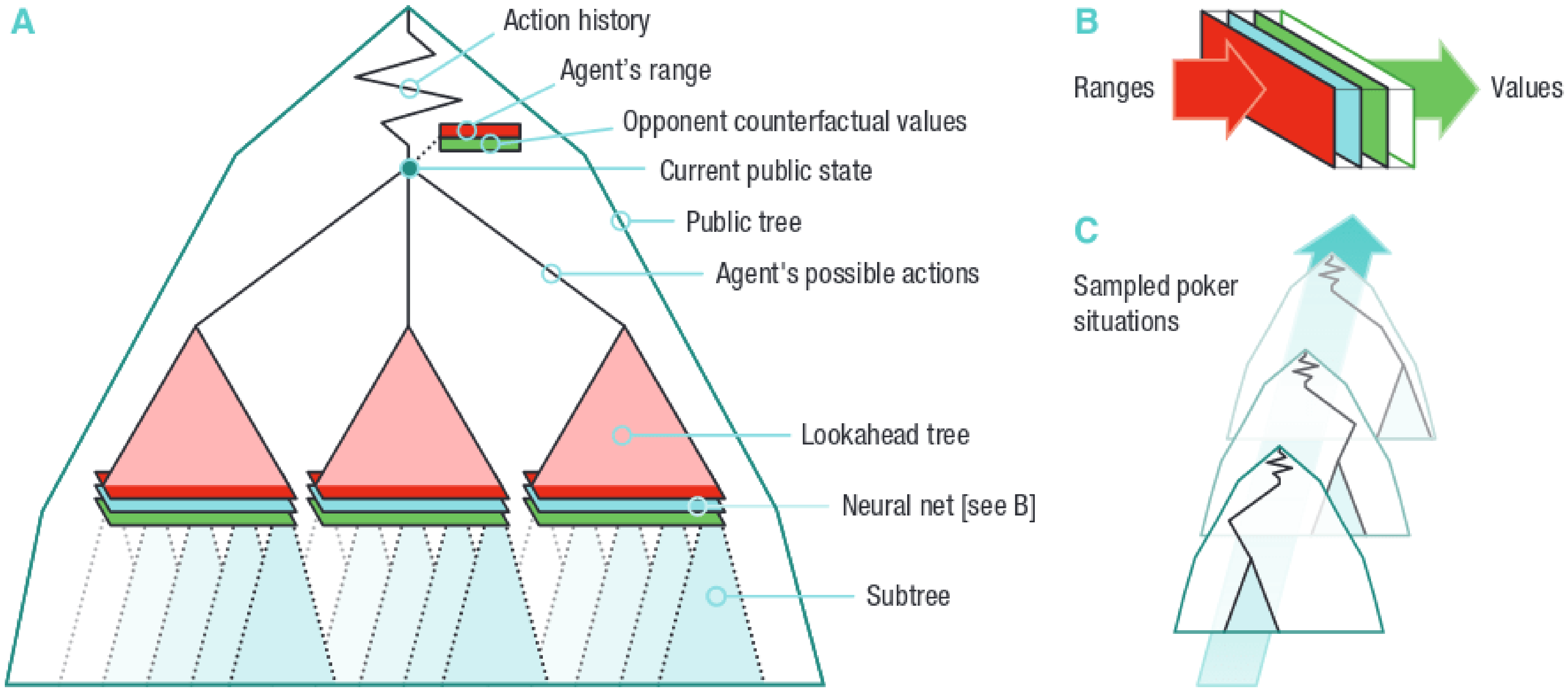
**function** FICTITIOUSSELFPLAY$(\Gamma, n, m)$
  Initialize completely mixed $\pi_1$
  $\beta_2 \leftarrow \pi_1$
  $j \leftarrow 2$
  **while** within computational budget **do**
    $\eta_j \leftarrow$ MIXINGPARAMETER$(j)$
    $\mathcal{D} \leftarrow$ GENERATEDATA$(\pi_{j-1}, \beta_j, n, m, \eta_j)$
    **for** each player $i \in \mathcal{N}$ **do**
      $\mathcal{M}^i_{RL} \leftarrow$ UPDATERLMEMORY$(\mathcal{M}^i_{RL}, \mathcal{D}^i)$
      $\mathcal{M}^i_{SL} \leftarrow$ UPDATESLMEMORY$(\mathcal{M}^i_{SL}, \mathcal{D}^i)$
      $\beta^i_{j+1} \leftarrow$ REINFORCEMENTLEARNING$(\mathcal{M}^i_{RL})$
      $\pi^i_j \leftarrow$ SUPERVISEDLEARNING$(\mathcal{M}^i_{SL})$
    **end for**
    $j \leftarrow j + 1$
  **end while**
  **return** $\pi_{j-1}$
**end function**

**function** GENERATEDATA$(\pi, \beta, n, m, \eta)$
  $\sigma \leftarrow (1-\eta)\pi + \eta\beta$
  $\mathcal{D} \leftarrow n$ episodes $\{t_k\}_{1 \le k \le n}$, sampled from strategy profile $\sigma$
  **for** each player $i \in \mathcal{N}$ **do**
    $\mathcal{D}^i \leftarrow m$ episodes $\{t^i_k\}_{1 \le k \le m}$, sampled from strategy profile $(\beta^i, \sigma^{-i})$
    $\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \mathcal{D}$
  **end for**
  **return** $\{\mathcal{D}^k\}_{1 \le k \le N}$
**end function**




Heinrich et al. 2015 "Fictitious Self Play in Extensive-Form Games"

# Deep Stack: Scaling to Heads Up No Limit Texas Hold 'Em



# Tree-Based Planning in POSGs

[Becker & Sunberg, AAMAS 2025]

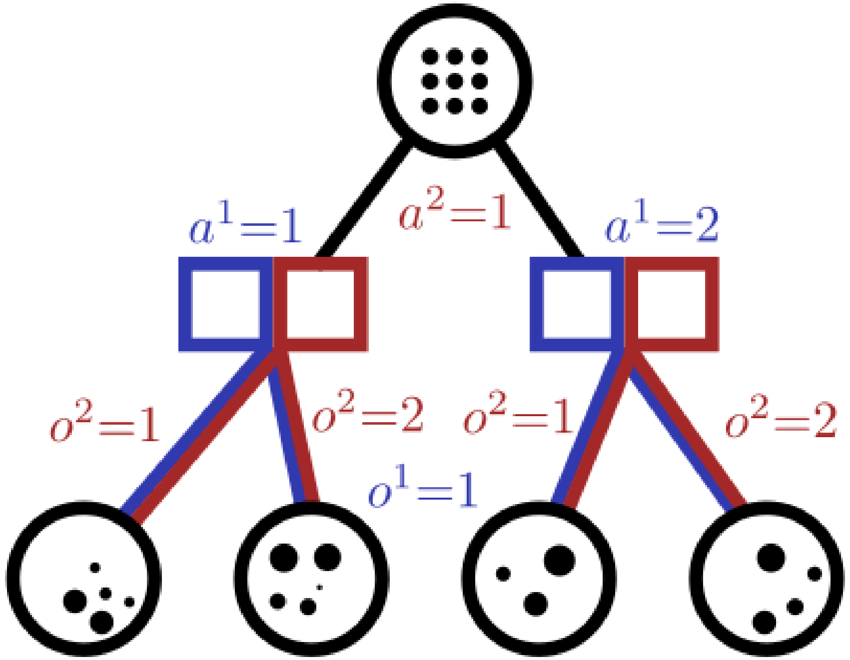
# Tree-Based Planning in POSGs

**Our approach: combine particle filtering and information sets**

[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

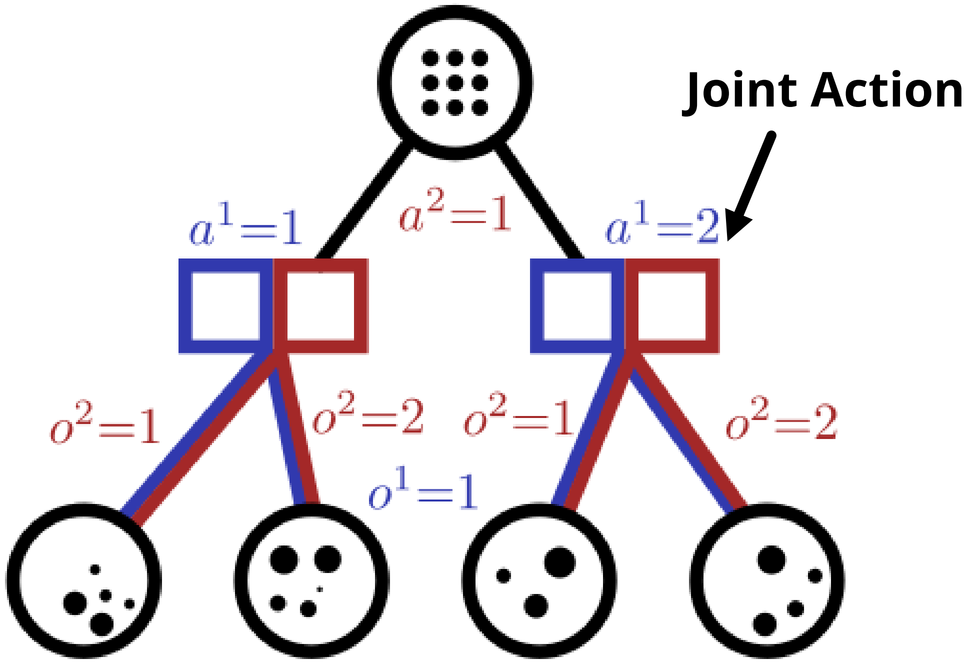
**Our approach: combine particle filtering and information sets**




[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

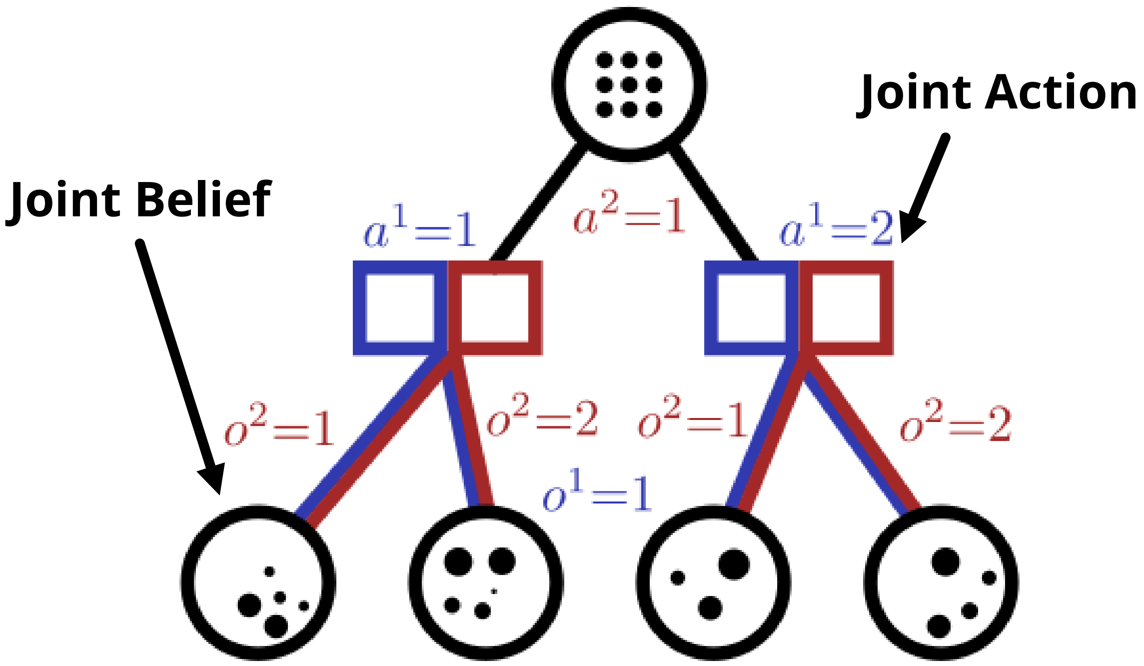
**Our approach: combine particle filtering and information sets**




[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

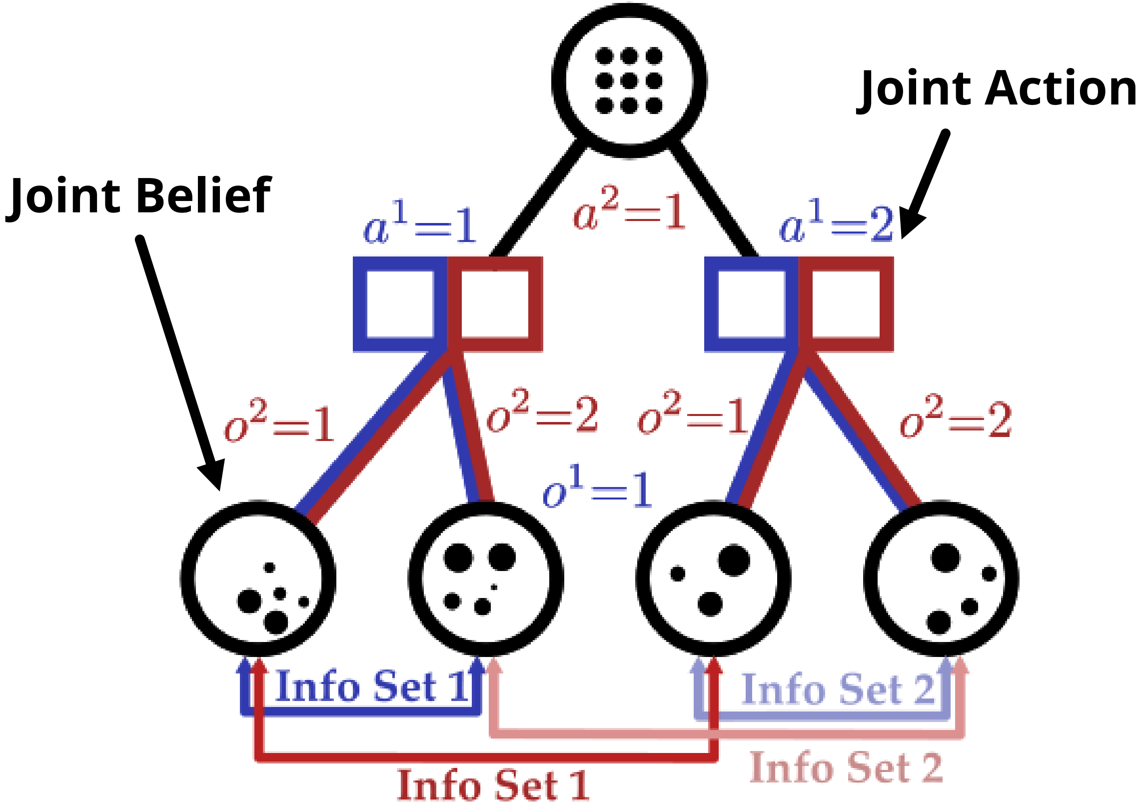
**Our approach: combine particle filtering and information sets**




[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

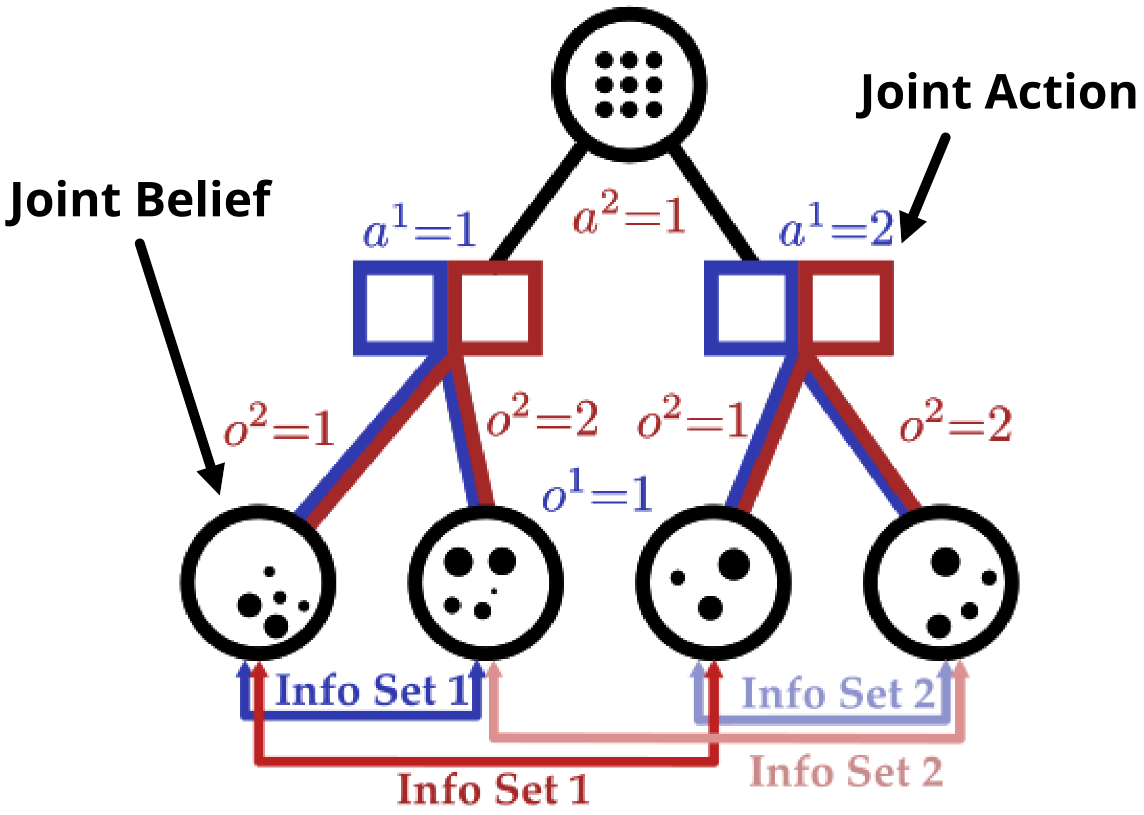
**Our approach: combine particle filtering and information sets**

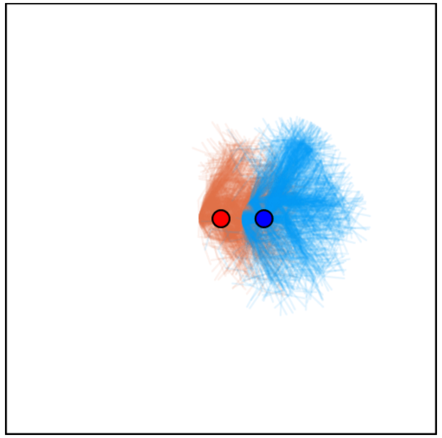



[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

**Our approach: combine particle filtering and information sets**

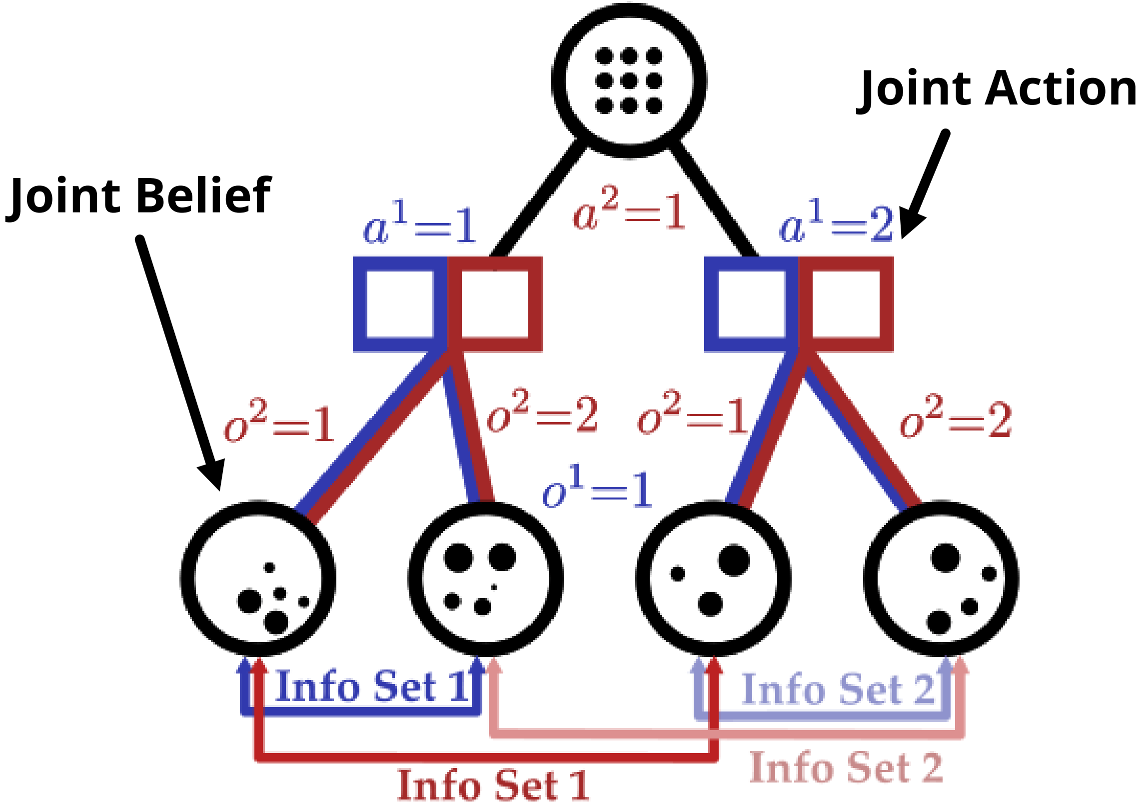


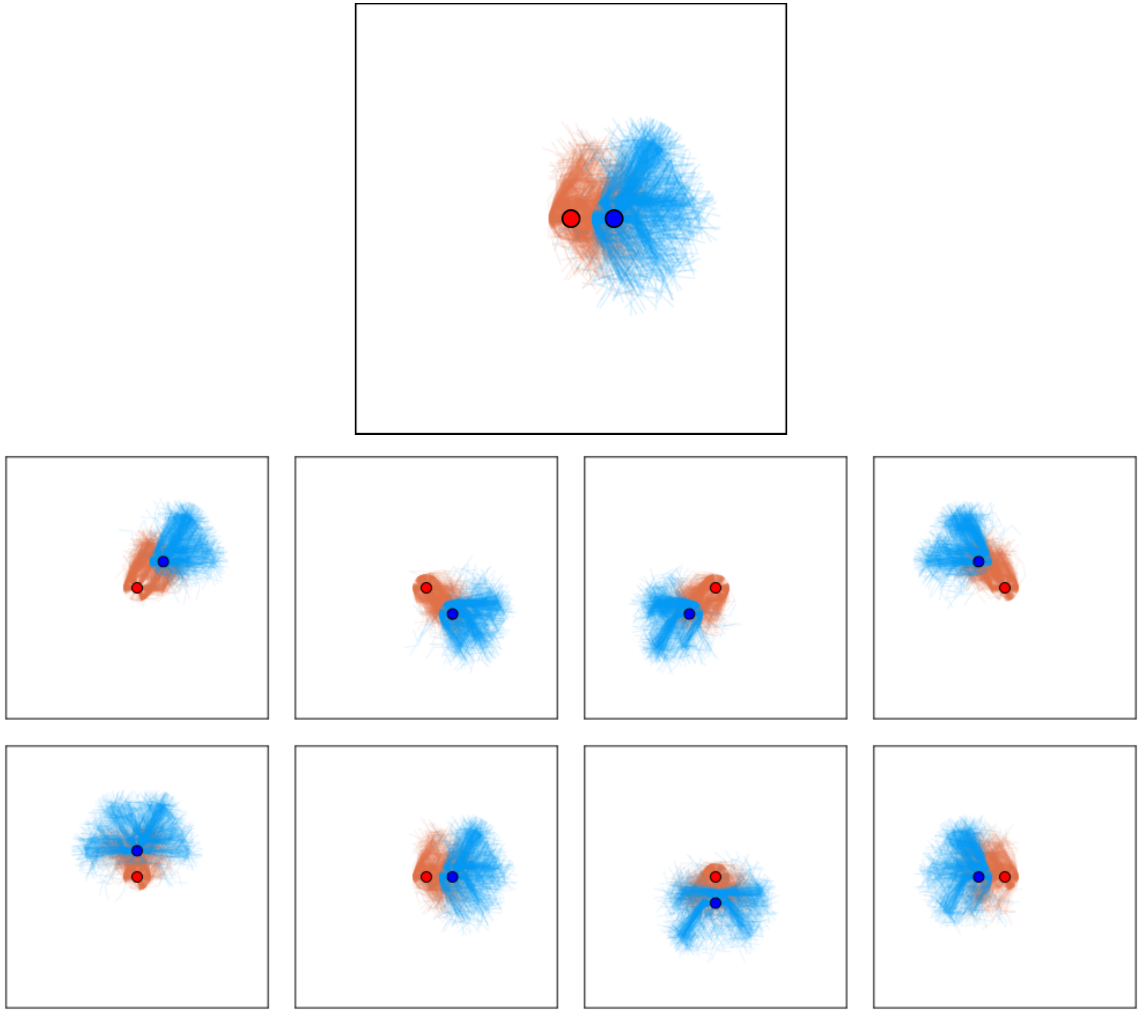



[Becker & Sunberg, AAMAS 2025]

# Tree-Based Planning in POSGs

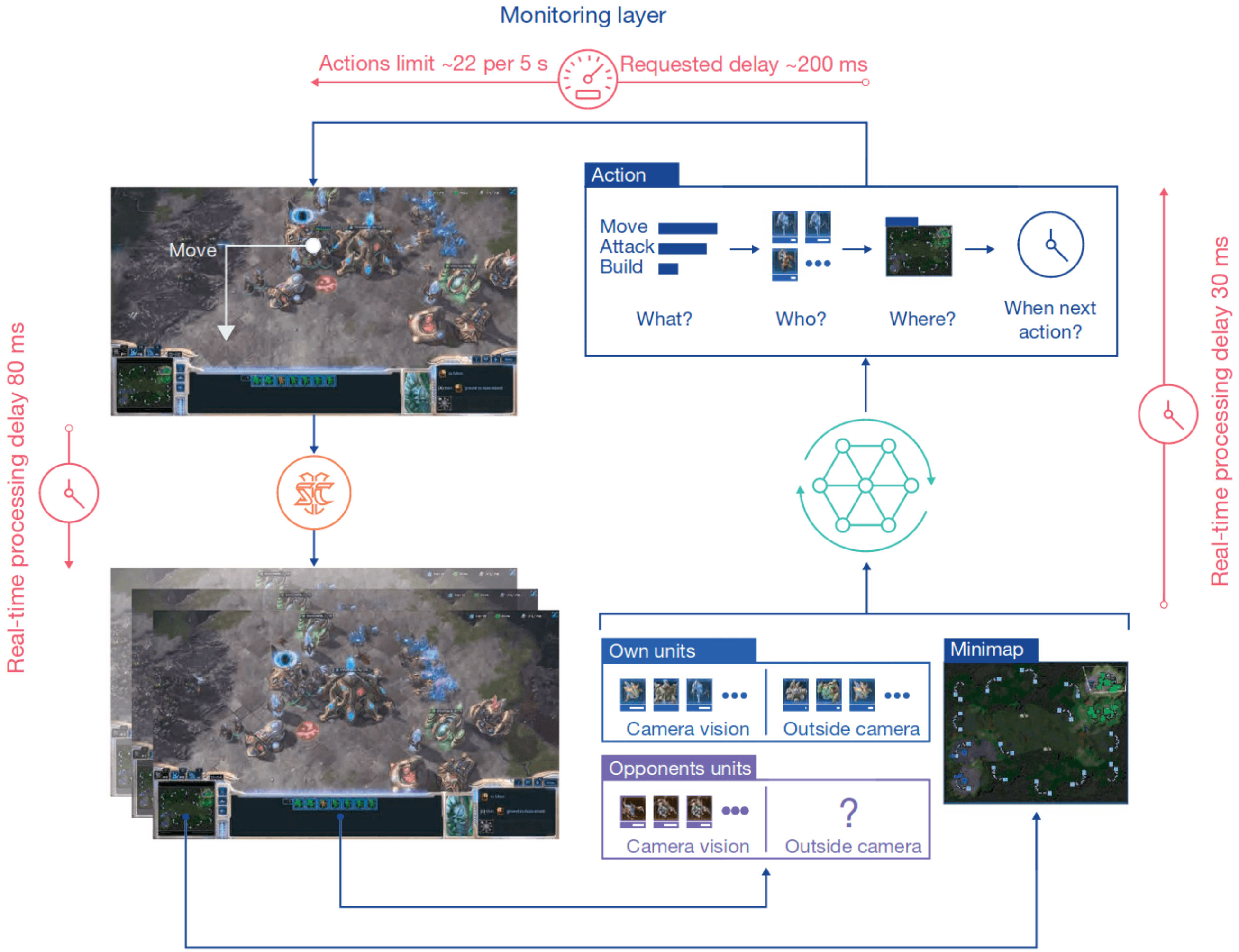
**Our approach: combine particle filtering and information sets**



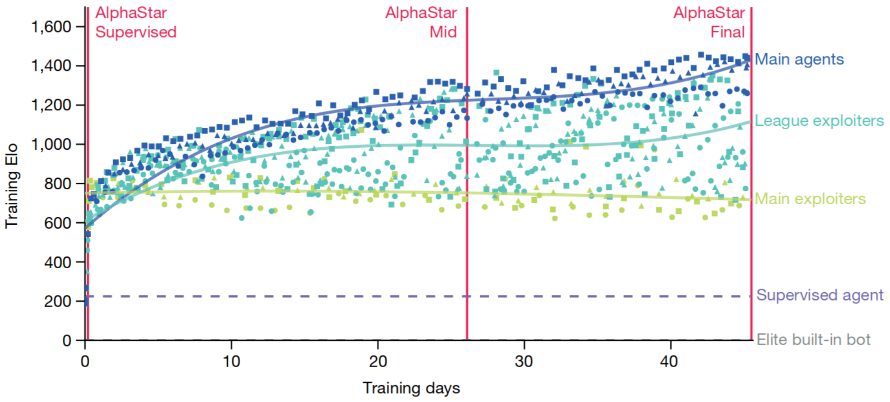



[Becker & Sunberg, AAMAS 2025]

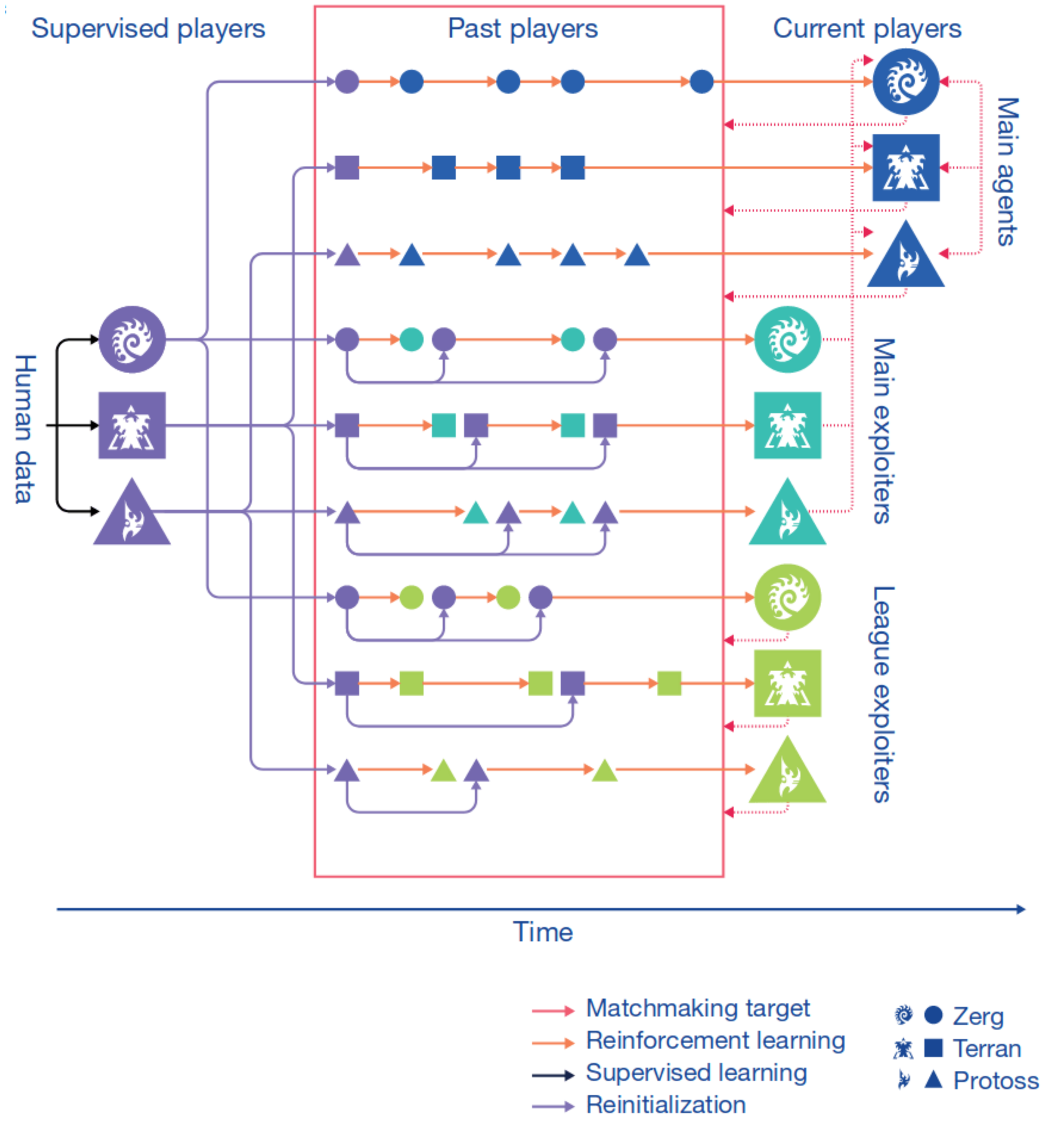
# Alpha Star



Monitoring layer
Actions limit ~22 per 5 s
Requested delay ~200 ms
Action
Move
Attack
Build
What?
Who?
Where?
When next action?
Real-time processing delay 80 ms
Real-time processing delay 30 ms
Move
Own units
Camera vision
Outside camera
Opponents units
Camera vision
?
Outside camera
Minimap

AlphaStar Supervised
AlphaStar Mid
AlphaStar Final
Main agents
League exploiters
Main exploiters
Supervised agent
Elite built-in bot
Training Elo
Training days
1,600
1,400
1,200
1,000
800
600
400
200
0
0
10
20
30
40

Supervised players
Past players
Current players
Main agents
Main exploiters
League exploiters
Human data
Time
Matchmaking target
Reinforcement learning
Supervised learning
Reinitialization
Zerg
Terran
Protoss

# Deep Nash

| | | Player 2 | |
|---|---|---|---|
| | | Head: $H$ | Tail: $T$ |
| Player 1 | Head: $H$ | 1 | $-1$ |
| | Tail: $T$ | $-1$ | 1 |

(a) Matching pennies

**R-NaD Iteration**

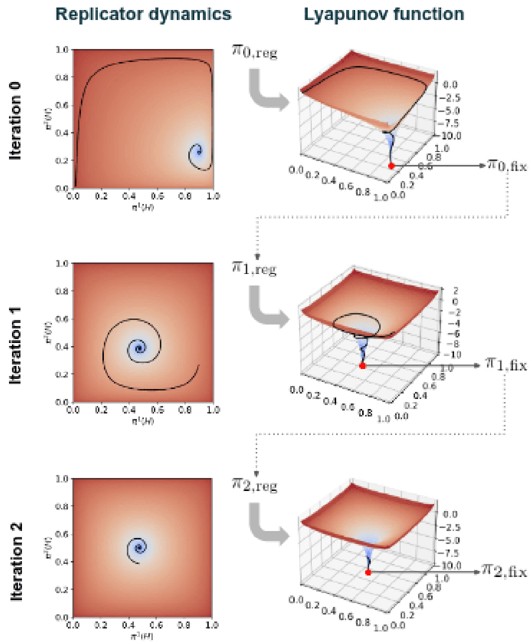
Start with an arbitrary regularization policy: $\pi_{0,\text{reg}}$

1. Reward transformation: Construct the transformed game with: $\pi_{n,\text{reg}}$
2. Dynamics: Run the replicator dynamics until convergence to: $\pi_{n,\text{fix}}$
3. Update: Set the regularization policy:

$$\pi_{n+1,\text{reg}} = \pi_{n,\text{fix}}$$

Repeat steps until convergence

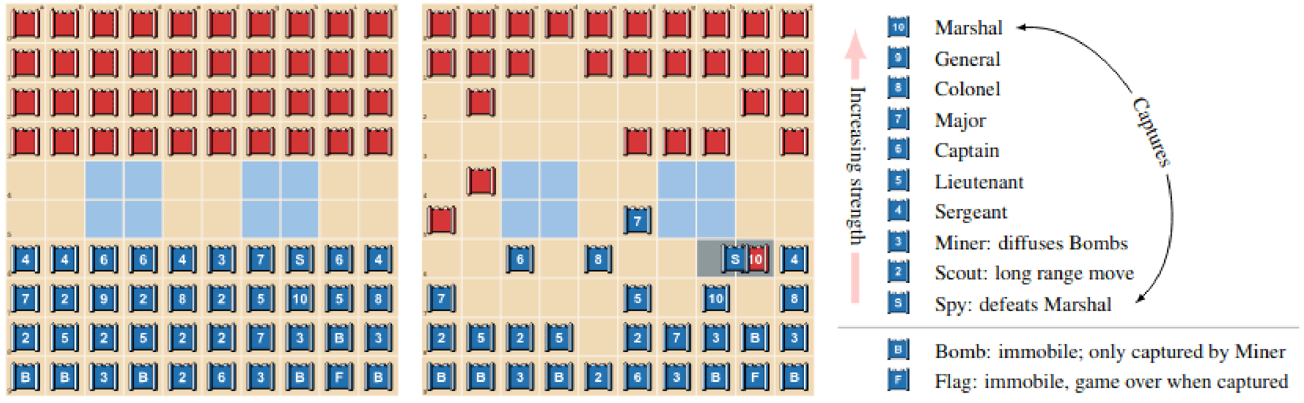
(b) Algorithmic steps




(c) Dynamics and Lyapunov function

Figure 2: The R-NaD learning algorithm illustrated with the matching pennies game

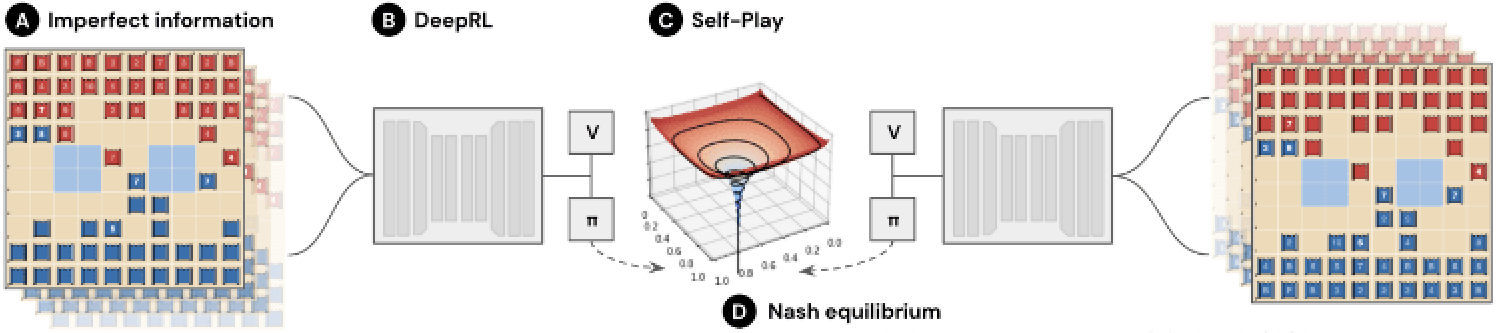
## Stratego




**Phase 1:** Private deployment

**Phase 2:** Game play

Piece types




**Replicator dynamics:** $\frac{d}{d\tau}\pi^i_\tau(a^i) = \pi^i_\tau(a^i)\left[Q^i_{\pi_\tau}(a^i) - \sum_{b^i}\pi^i_\tau(b^i)Q^i_{\pi_\tau}(b^i)\right]$

**Reward transformation:** $r^i(\pi^i, \pi^{-i}, a^i, a^{-i}) = r^i(a^i, a^{-i}) - \eta \log\left(\frac{\pi^i(a^i)}{\pi^i_{\text{reg}}(a^i)}\right) + \eta \log\left(\frac{\pi^{-i}(a^{-i})}{\pi^{-i}_{\text{reg}}(a^{-i})}\right)$