Dynamic Games

- Markov Games
- 2 Player zero sum turn taking games

Markov Games

Markov Game Definition

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: $T(s' \mid s, a)$ 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

Goofspiel (or Game of Pure Strategy)

Goofspiel(5) Optimal Strategy

Table 1. Optimal strategy at the first move, $N = 5$.

	upcard				
	1	**2**	**3**	**4**	**5**
1	0.0470	0.1855	0.1182	0.1226	0.1123
2	0.8327	0	0.1188	0.07347	0.0241
3	0.1203	0.7375	0	0.1915	0
4	0	0.0770	0.7630	0.2043	0
5	0	0	0	0.4081	0.8636

Calculated with Dynamic Programming
[Rhoads and Bartholdi, 2012]

This is not a game payoff matrix!

Nash equilibrium at every state

$$\begin{aligned}
\underset{\boldsymbol{\pi}, \mathcal{U}}{\text{minimize}} \quad & \sum_{i \in \mathcal{I}} \sum_{s} \Big(\mathcal{U}^i(s) - Q^i(s, \boldsymbol{\pi}(s)) \Big) \\
\text{subject to} \quad & \mathcal{U}^i(s) \geq Q^i(s, a^i, \boldsymbol{\pi}^{-i}(s)) \text{ for all } i, s, a^i \\
& \sum_{a^i} \pi^i(a^i \mid s) = 1 \text{ for all } i, s \\
& \pi^i(a^i \mid s) \geq 0 \text{ for all } i, s, a^i
\end{aligned}$$

where

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

Reduction to Simple Game

Best response in a Markov Game

Fictitious Play in a Markov Game

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

Turn-taking Games

Terminology

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

Minimax Trees

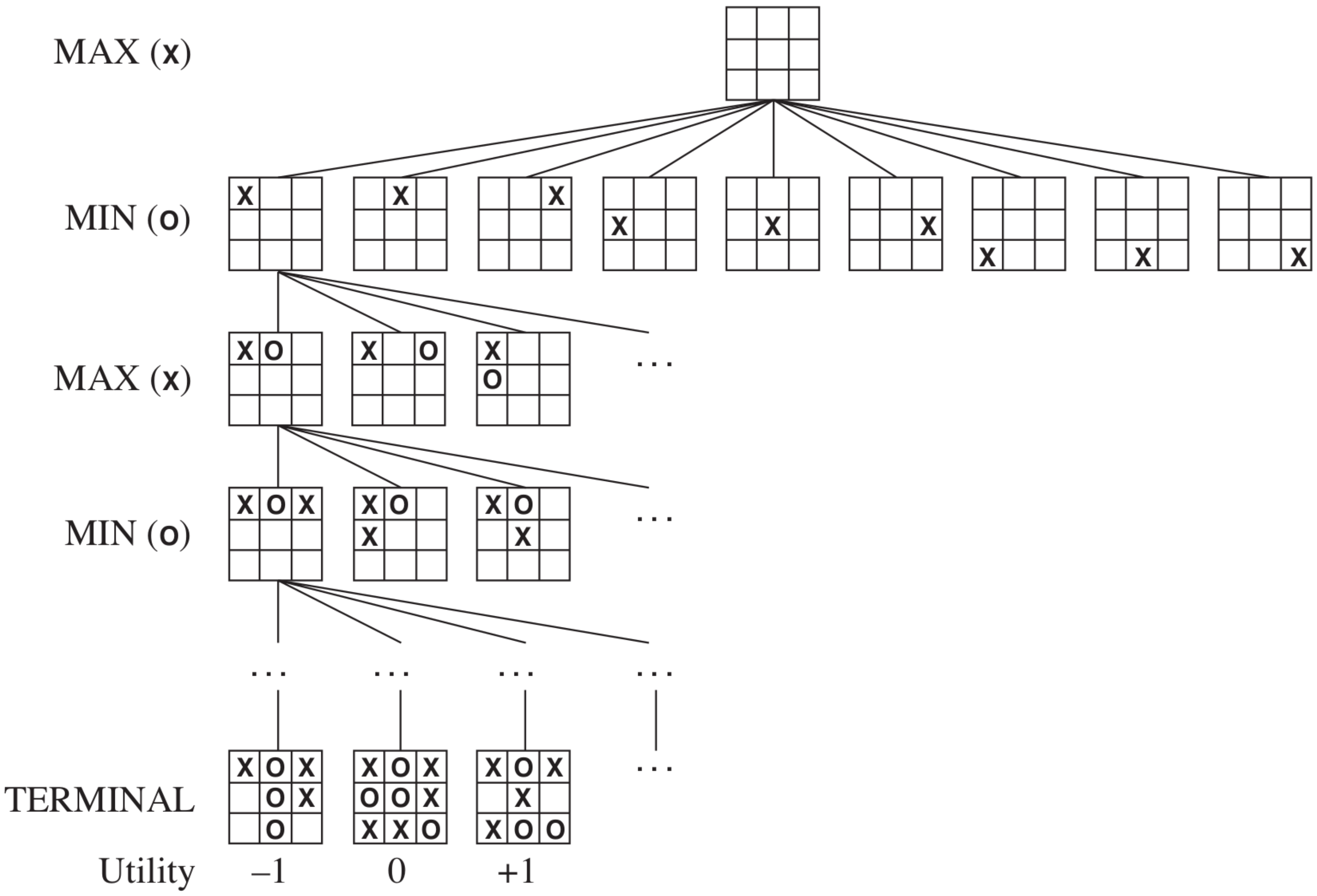
MDP Expectimax Tree

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

Minimax Tree

$$V(s) = \max_{a \in \mathcal{A}_1} \left(R(s,a) + \min_{a' \in A_2}(R(s',a') + V(s''))\right)$$

Tic-Tac-Toe Example



Tree Backup Example

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

---

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**
    $v \leftarrow$ MAX(v, MIN-VALUE(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**
    $v \leftarrow$ MIN(v, MAX-VALUE(RESULT(s, a)))
  return v

3 12 8 2 4 6 14 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

---

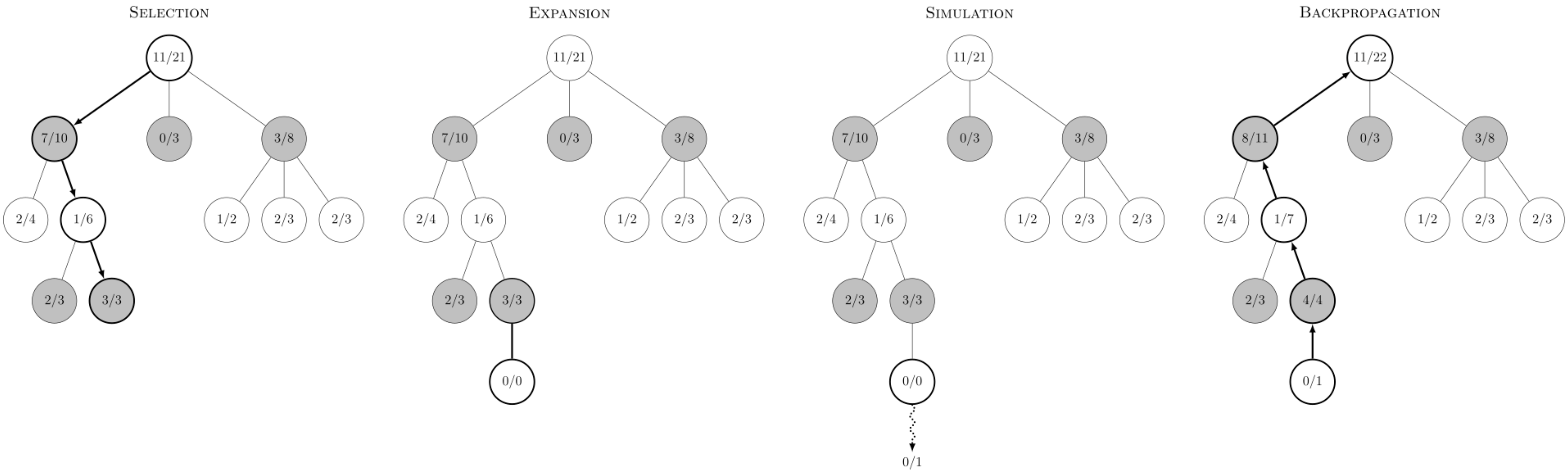
function MAX-VALUE($state, \alpha, \beta$) **returns** a $utility$ $value$
  if TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  for each a **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if $v \geq \beta$ **then return** v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

3 12 8 2 4 6 14 5 2

MCTS for Games



Note: the above example does not follow UCB exploration

Incomplete Information

Partially Observable Markov Game

Belief updates?

Reduction to Simple Game

Extensive Form Game

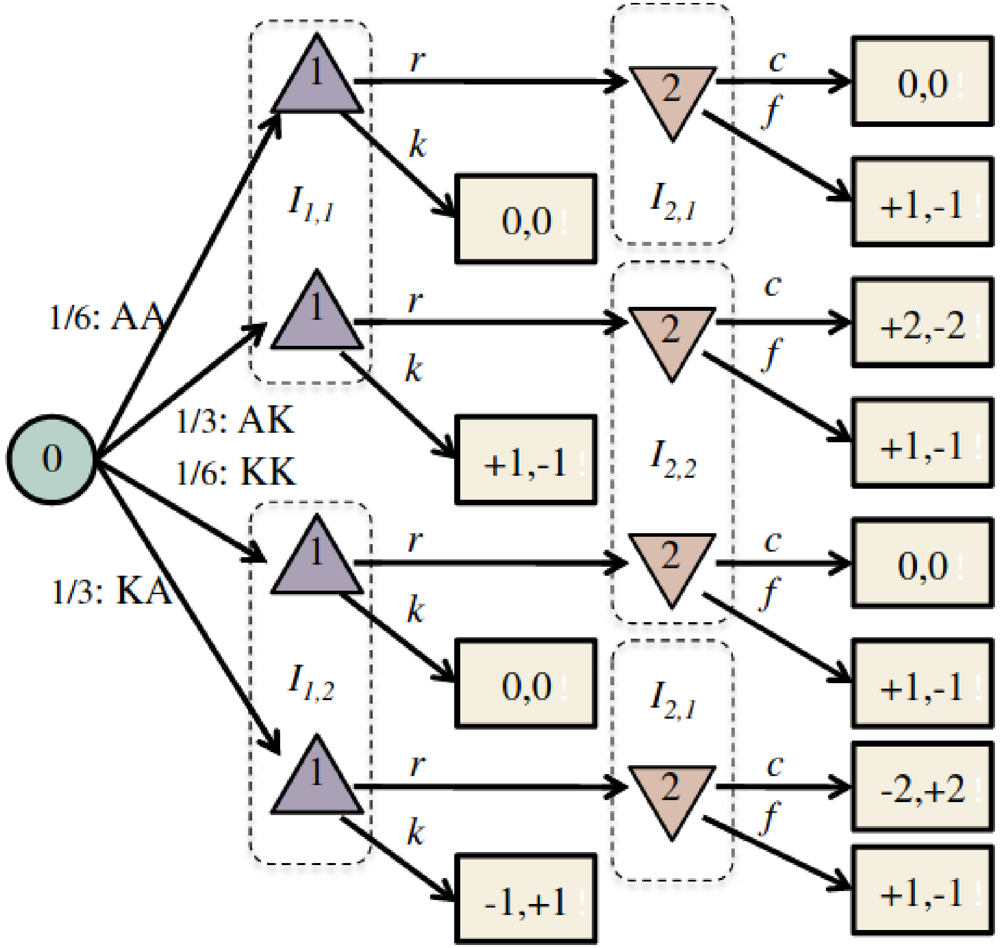
(Alternative to POMGs that is more common in the literature)

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

King-Ace Poker Example

- 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

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

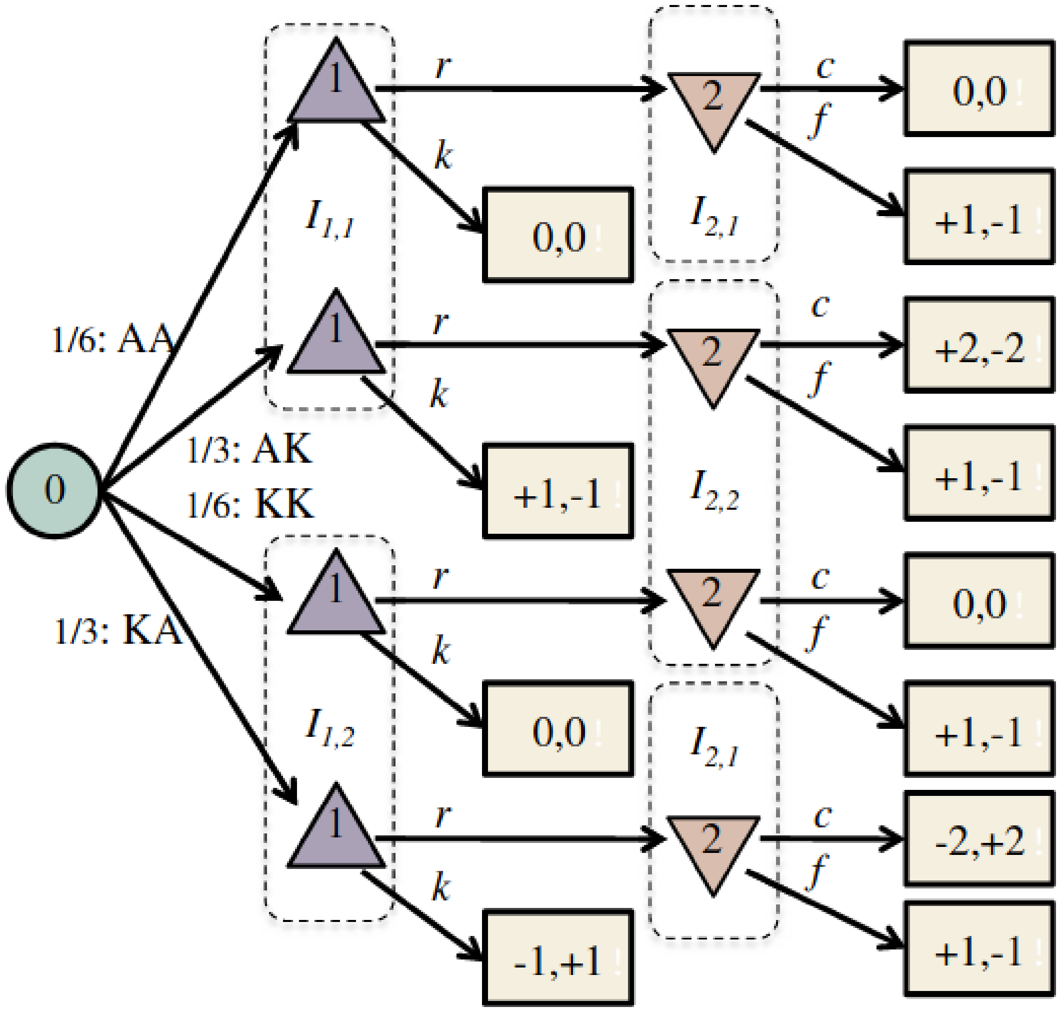
Exponential in number of info states!

Fictitious Play in Extensive Form Games

Algorithm 2 General Fictitious Self-Play

function $\textsc{FictitiousSelfPlay}(\Gamma, n, m)$
  Initialize completely mixed π_1
  $\beta_2 \leftarrow \pi_1$
  $j \leftarrow 2$
  while within computational budget **do**
    $\eta_j \leftarrow \textsc{MixingParameter}(j)$
    $\mathcal{D} \leftarrow \textsc{GenerateData}(\pi_{j-1}, \beta_j, n, m, \eta_j)$
    for each player $i \in \mathcal{N}$ **do**
      $\mathcal{M}^i_{RL} \leftarrow \textsc{UpdateRLMemory}(\mathcal{M}^i_{RL}, \mathcal{D}^i)$
      $\mathcal{M}^i_{SL} \leftarrow \textsc{UpdateSLMemory}(\mathcal{M}^i_{SL}, \mathcal{D}^i)$
      $\beta^i_{j+1} \leftarrow \textsc{ReinforcementLearning}(\mathcal{M}^i_{RL})$
      $\pi^i_j \leftarrow \textsc{SupervisedLearning}(\mathcal{M}^i_{SL})$
    end for
    $j \leftarrow j + 1$
  end while
  return π_{j-1}
end function

function $\textsc{GenerateData}(\pi, \beta, n, m, \eta)$
  $\sigma \leftarrow (1-\eta)\pi + \eta\beta$
  $\mathcal{D} \leftarrow n$ episodes $\{t_k\}_{1 \le k \le n}$, sampled from strategy profile σ
  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 (β^i, σ^{-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"

This slide not covered on exam