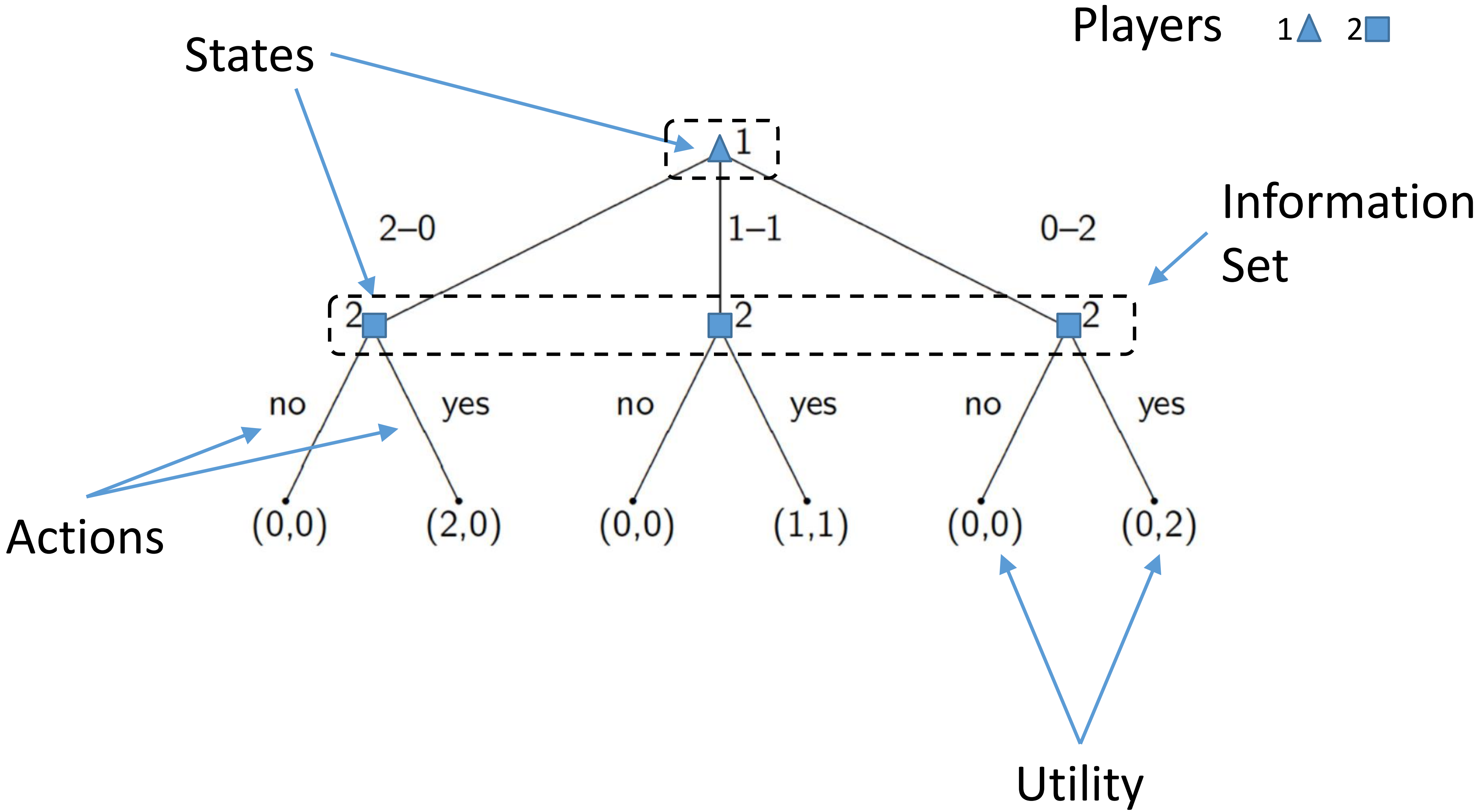
OTEVŘENÁ INFORMATIKA

Solving Zero-Sum Extensive-Form Games

Branislav Bošanský

AE4M36MAS, Fall 2013, Lecture 6

Imperfect Information EFGs



Solving II Zero-Sum EFG with perfect recall

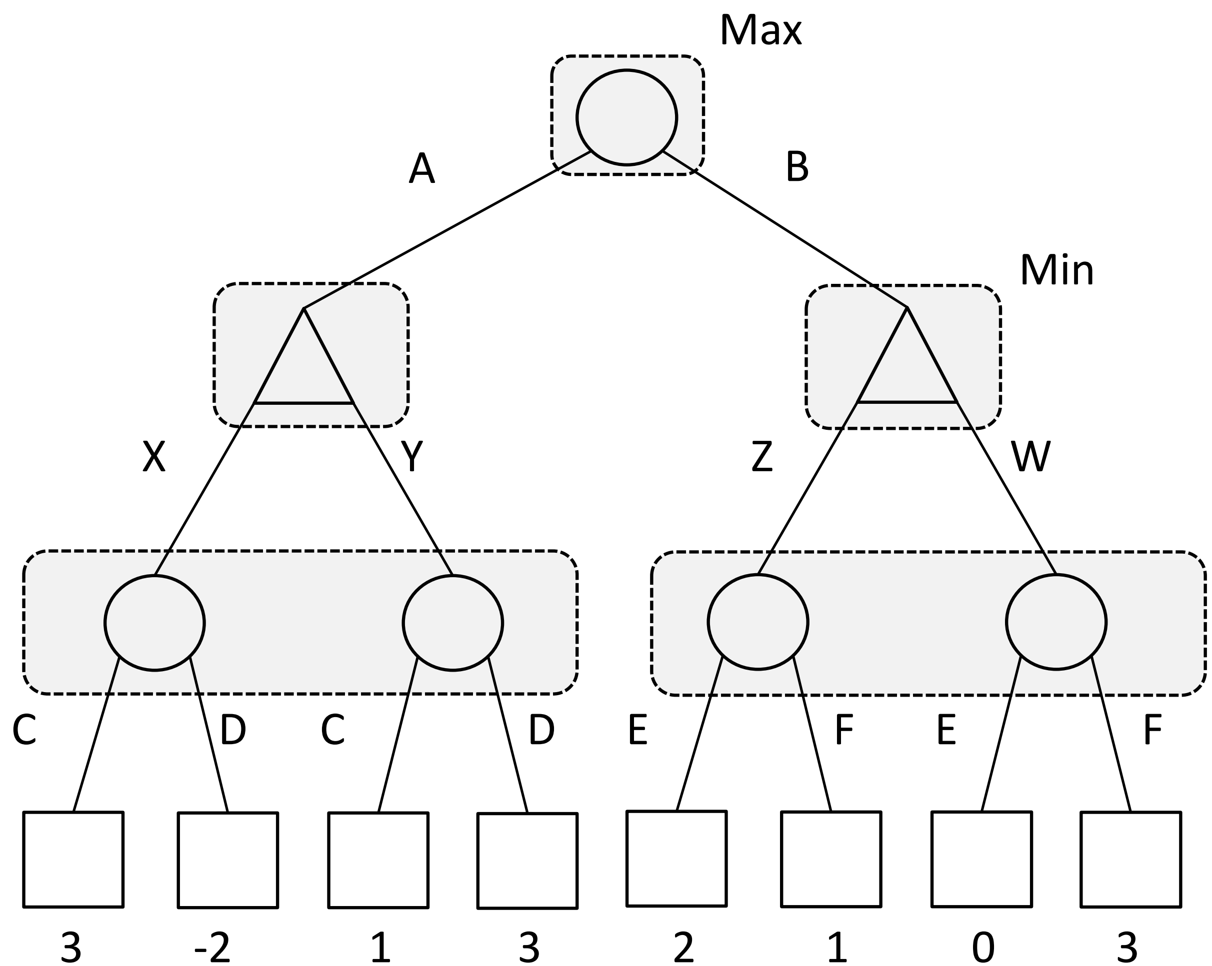
Exact algorithms:

- Transformation to the normal form
- Using the sequence form
- (Iterative improvements of the sequence form)

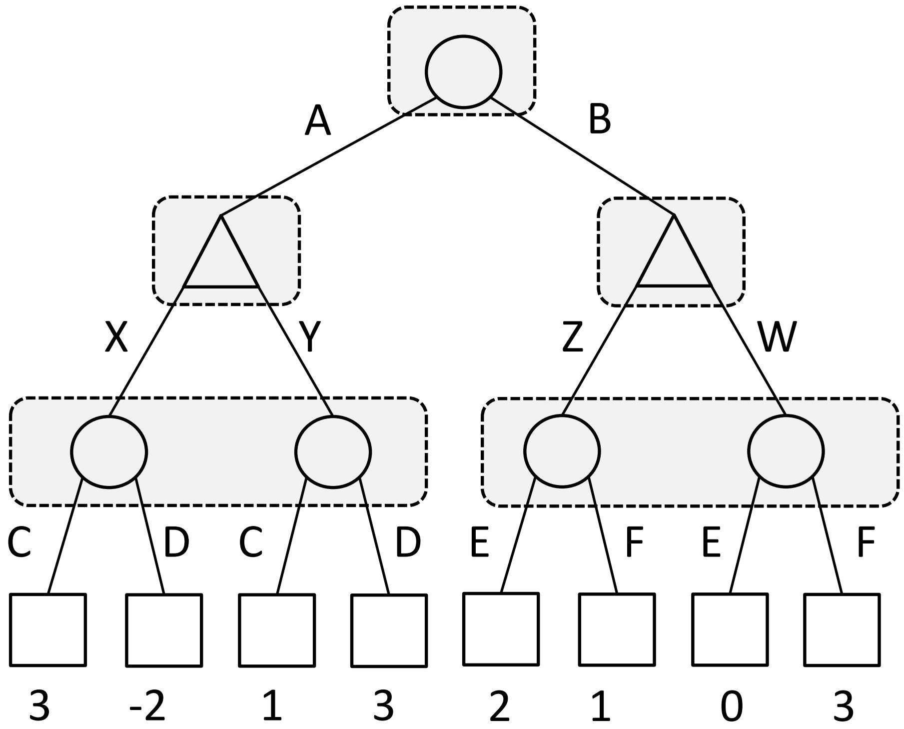
Approximate algorithms:

- Counterfactual Regret Minimization
- Excessive Gap Technique
- (variants of Monte-Carlo Tree Search)

Imperfect Information Zero-Sum EFG

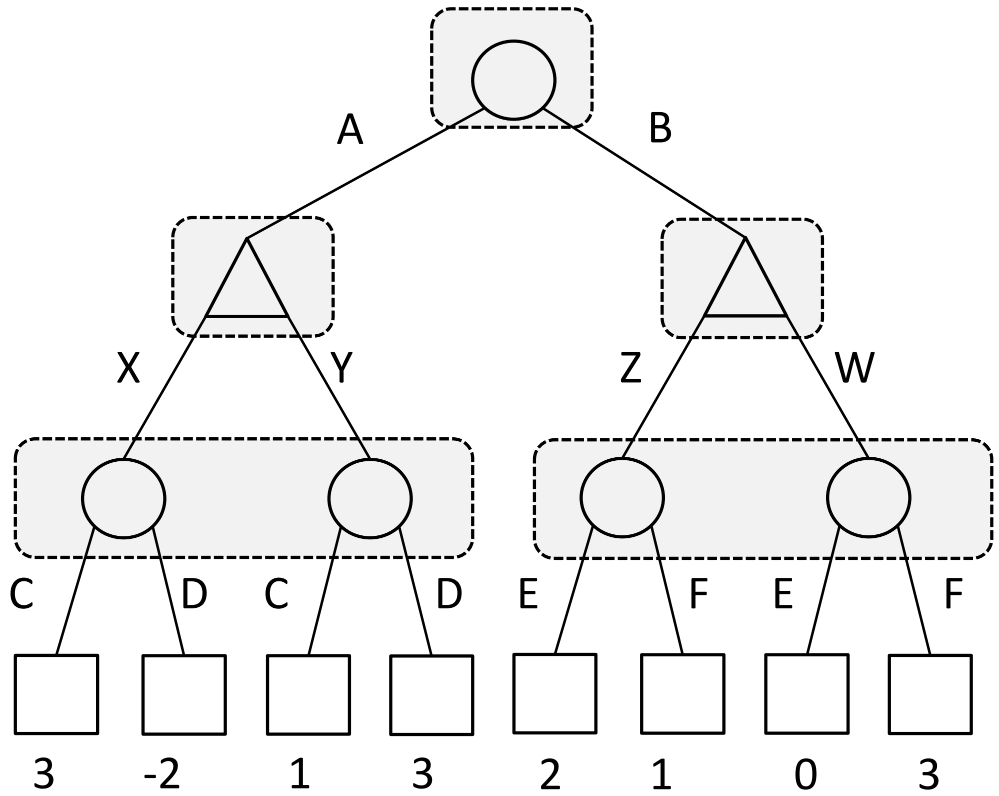


Imperfect Information Zero-Sum EFG



	XZ	XW	YZ	YW
ACE	3	3	1	1
ACF	3	3	1	1
ADE	-2	-2	3	3
ADF	-2	-2	3	3
BCE	2	0	2	0
BCF	1	3	1	3
BDE	2	0	2	0
BDF	1	3	1	3

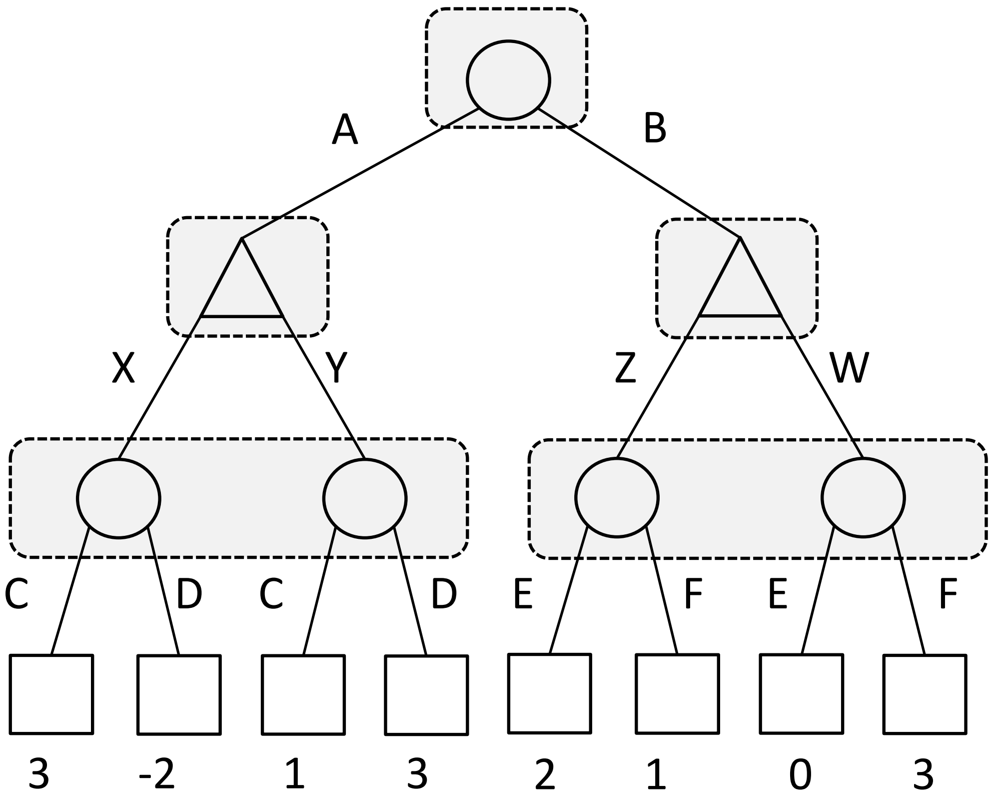
II EFGs - Sequences



Circle (Σ_1)	Triangle (Σ_2)
∅	∅
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- alternative representation of strategies
- $\sigma_i \in \Sigma_i$
- we use $\sigma_i a$ to denote executing an action a after the sequence σ_i

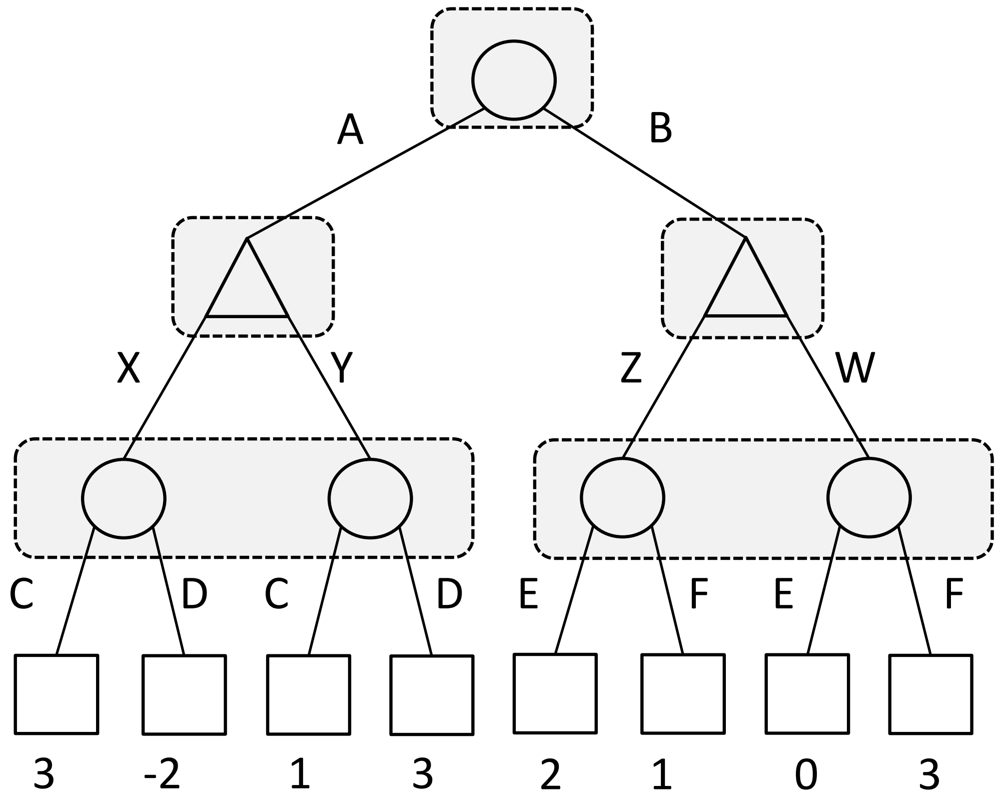
II EFGs - Sequences



Circle (Σ_1)	Triangle (Σ_2)
∅	∅
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- extension of the utility function *g*
  - $g_i\colon \Sigma_1 \times \Sigma_2 \rightarrow \mathbb{R}$
- sequentially execute actions of the players
  - stop at either:
    - leaf - $z \in Z$ $\quad g_i(\sigma_1, \sigma_2) = u_i(z)$
    - there is no applicable action $g_i(\sigma_1, \sigma_2) = 0$

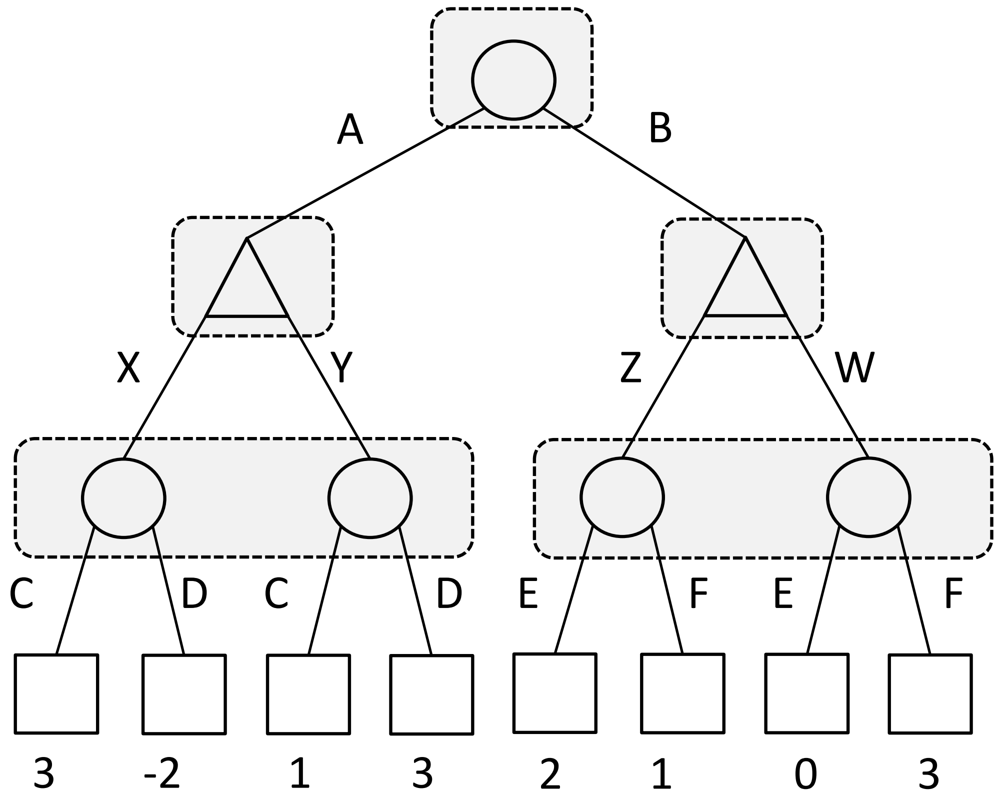
II EFGs - Sequences



Circle (Σ_1)	Triangle (Σ_2)
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- Examples
  - $g_1(\emptyset, W) = 0$
  - $g_1(AC, W) = 0$
  - $g_1(BF, W) = 3$
  - …

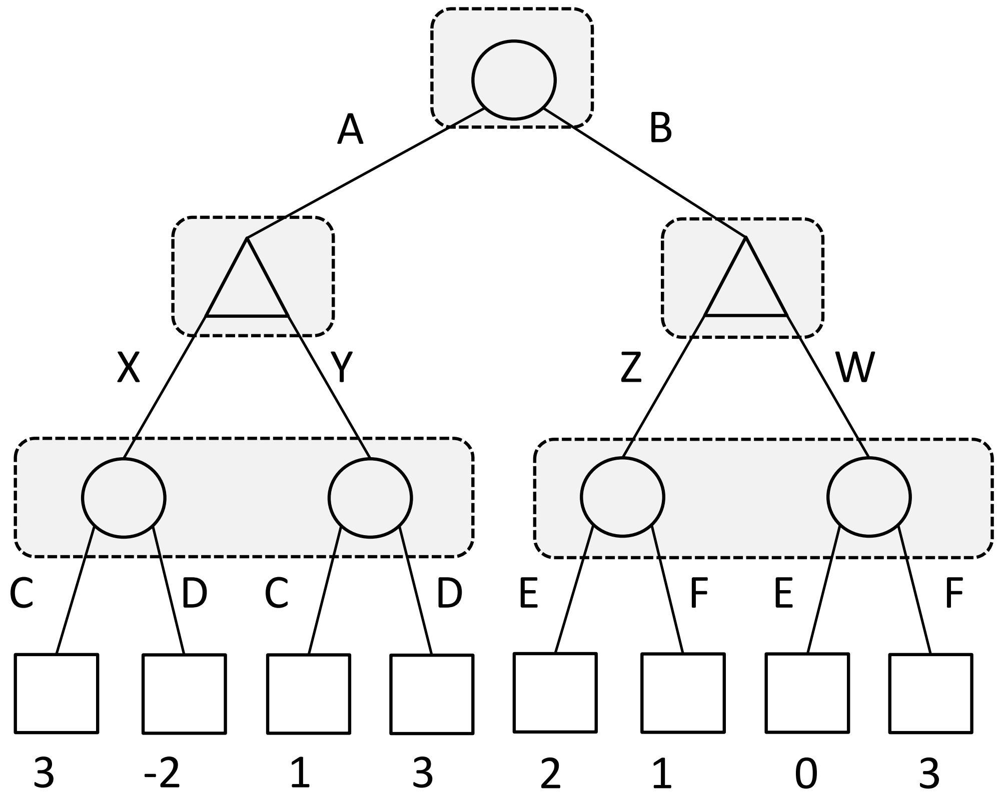
II EFGs – Realization Plans



Circle (Σ_1)	Triangle (Σ_2)
∅	∅
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- behavioral strategies represented as realization plans
  - probabilities for sequences
  - assume the opponent allows us to play the actions from the sequence
  - RP ~ probability that this sequence will be played in this strategy

II EFGs – Realization Plans



Circle (Σ_1)	Triangle (Σ_2)
∅	∅
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- $r_1(\emptyset) = 1$
- $r_1(A) + r_1(B) = r_1(\emptyset)$
- $r_1(AC) + r_1(AD) = r_1(A)$
- $r_1(BE) + r_1(BF) = r_1(B)$

- $r_2(\emptyset) = 1$
- $r_2(X) + r_2(Y) = r_2(\emptyset)$
- $r_2(Z) + r_2(W) = r_2(\emptyset)$

- network-flow perspective

II EFGs – Sequence Form LP

- calculate NE
  - optimization against a best-response of the opponent
  - $\mathcal{J}(\sigma)$ – information set, in which the last action was executed
  - $seq(I)$ – sequence leading to an information set I
  - v_{I_i} - expected utility in an information set

- $\max v_{\mathcal{J}(\emptyset)}$
- $r_1(\emptyset) = 1$
- $\forall I_{1,k} \in I_1, \sigma = seq(I_{1,k})$: $\quad r_1(\sigma) = \sum_{\{a \in A(I_{1,k})\}} r_1(\sigma a)$
- $\forall \sigma_2 \in \Sigma_2$: $\quad v_{\mathcal{J}(\sigma_2)} \leq \sum_{\{I_{2,j} : seq(I_{2,j}) = \sigma_2\}} v_{I_{2,j}} + \sum_{\sigma_1} g_1(\sigma_1, \sigma_2) r_1(\sigma_1)$

Alternative algorithms - learning

- instead of calculating the exact NE, we can learn the best strategy to play
- repeatedly play the game and adjust the strategy according to the observations
- hopefully, we should converge to an equilibrium

- the simplest learning rule in games:
  - fictitious play
    - assume the opponent is playing NE strategy
    - we should play the best response against it
    - we can do that at each iteration

Fictitious play

	L	R
U	3	1
D	-2	3

1. assume some a-priori strategy of the opponent (e.g., uniform)
2. calculate pure best-response strategy
3. play this strategy and observe the action of the opponent
4. update the belief about the mixed strategy of the opponent
5. repeat from step 2

Fictitious play and learning

	L	R
U	3	1
D	-2	3

- for many game models FP converges to a NE (e.g., zero-sum)
- the convergence is rather slow
- there are different variants …

- we are interested in **no-regret learning** algorithms

Reminder - regret

- Player i's regret for playing an action a_i if the other players adopt action profile a_{-i}

$$\left[\max_{a'_i \in A_i} u_i(a'_i, a_{-i})\right] - u_i(a_i, a_{-i}).$$

- we can use the concept of regret to learn the optimal strategy
  - to find such a strategy that would not yield less than playing any pure strategy

No-regret Learning: Regret Matching

- let's define α^t the average reward throughout iterations $1 \dots t$
- let's define $\alpha^t(s)$ the average reward throughout iterations $1 \dots t$ the player would have played s and all the opponents play as before
- regret at time t the agent experiences for not having played s equals $R^t(s) = \alpha^t(s) - \alpha^t$

- learning rule is "no-regret" if it guarantees that with high probability the agent will experience no positive regret

- Regret matching
  - start with an arbitrary strategy (e.g., uniform)
  - at each time step each action is chosen with probability proportional to its regret
  - $p^{t+1}(s) = \frac{R^t(s)}{\sum_{\{s' \in S\}} R^t(s')}$

Application of no-regret learning in EFGs

- Counterfactual Regret Minimization (CFR)
  - construct the complete game tree
  - in each iteration traverse through the game tree and adapt the strategy in each information set according to the learning rule
  - this learning rule minimizes the (counterfactual) regret
  - it was proven that the algorithm minimizes the overall regret in the game

Comparing the algorithms

- Sequence form
  - the leading exact algorithm
  - suffers from memory requirements
    - improved by iterative double-oracle construction of the game tree
- CFR
  - more memory-efficient
  - many incremental variants (MC-CFR,…)
  - leading algorithm for solving Poker
  - can solve some games with imperfect recall