Lecture 6: Solving Zero-Sum EFGs

17 Sept 2025

CS6208 Fall 2025: Computational Game Theory

Admin Matters

Quiz 1 has been released

- 2 weeks left to complete it, due 29 Sept
- To be completed **individually**

Admin Matters

Quiz 1 has been released

- 2 weeks left to complete it, due 29 Sept
- To be completed **individually**

Reminder about grading:

- Homework 1 (20%) + Quiz 1 (10%) = 30%
- Homework 2 (20%) + Quiz 2 (10%) = 30%
- Project = 40%
- Think about homework as having an individual / group component

Quiz 1 is due the same time as HW1+3 days

- A few of you have already submitted.
- You can submit unlimited times, last score is counted

Please visit office hours if you need help

Project Topic Proposals

Due 1 week after HW1

Upload on Canvas

- Make sure team members' names are visible

Check Canvas for more instructions

Email me or go to office hours if you need help generating ideas!

Review

EFGs

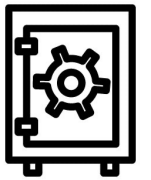
- Information sets (important!)
- Perfect Recall
- Timeability

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

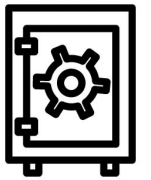
Finder

Hider



Review: Guess-the-number

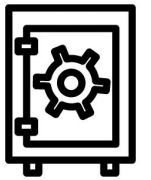
Dixit and Nalebuff, The Art of Strategy

Finder  Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

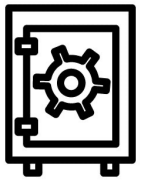
Finder  Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Finder choose a number x_1

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

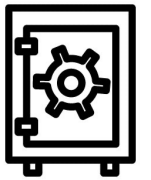
Finder  Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Finder choose a number x_1 → $x_1 = y$? → Finder wins v_1 dollars

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

Finder  Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Finder choose a number x_1

$x_1 = y$? → Finder wins v_1 dollars

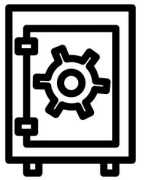
$x_1 \neq y$?

Finder observes if $x_1 < y$ OR $x_1 > y$

Finder chooses a new number x_2

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

Finder  Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Finder choose a number x_1

$x_1 = y$? → Finder wins v_1 dollars

$x_1 \neq y$?

Finder observes if $x_1 < y$ OR $x_1 > y$

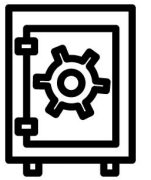
Finder chooses a new number x_2

$x_2 = y$? → Finder wins v_2 dollars

Review: Guess-the-number

Dixit and Nalebuff, The Art of Strategy

Finder 

Hider 

Hider picks an integer $y \in \{1, \dots, N\}$. This is kept a secret.

Repeat T times

Finder choose a number x_1

$x_1 = y$? → Finder wins v_1 dollars

$x_1 \neq y$?

Finder observes if $x_1 < y$ OR $x_1 > y$

Finder chooses a new number x_2

$x_2 = y$? → Finder wins v_2 dollars

$x_2 \neq y$?

Finder observes if $x_{t-1} < y$ OR $x_{t-1} > y$

Finder chooses a new number x_t

$x_t = y$? → Finder wins v_t dollars

…, …

Guess-the-number game as an EFG

Suppose $N = 50, T = 4, v = [80,60,40,20]$

What is the "perfect-information" version of the game?

- Transitions?
- Payoffs?

What are the infosets of the Hider?

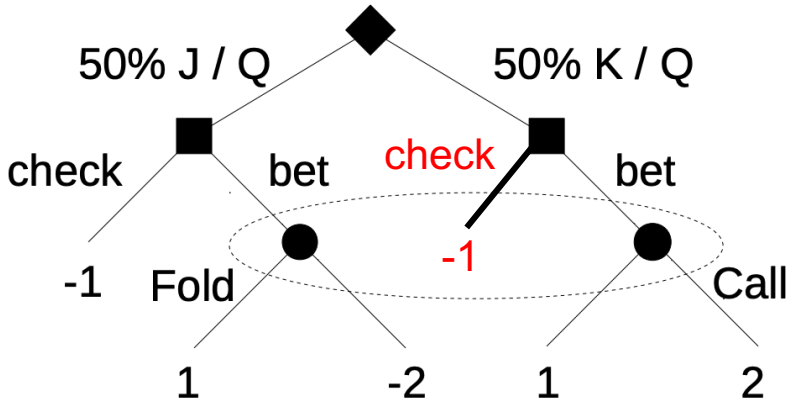
What are the infosets of the Finder?

Strategy Representation in EFGs

Method 1: conversion to normal form

Cartesian product of all actions at each infoset

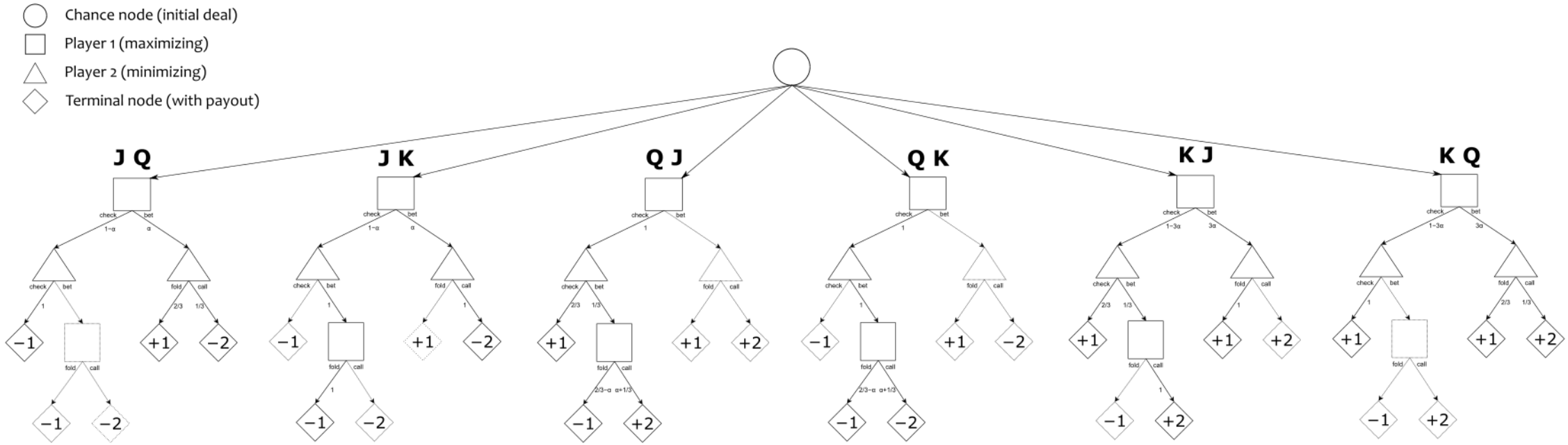
Recall example from Lecture 3



	J→Check K→Check	J→Check K→Bet	J→Bet K→Check	J→Bet K→Bet
Fold	-1	0	0	1
Call	-1	0.5	-1.5	0

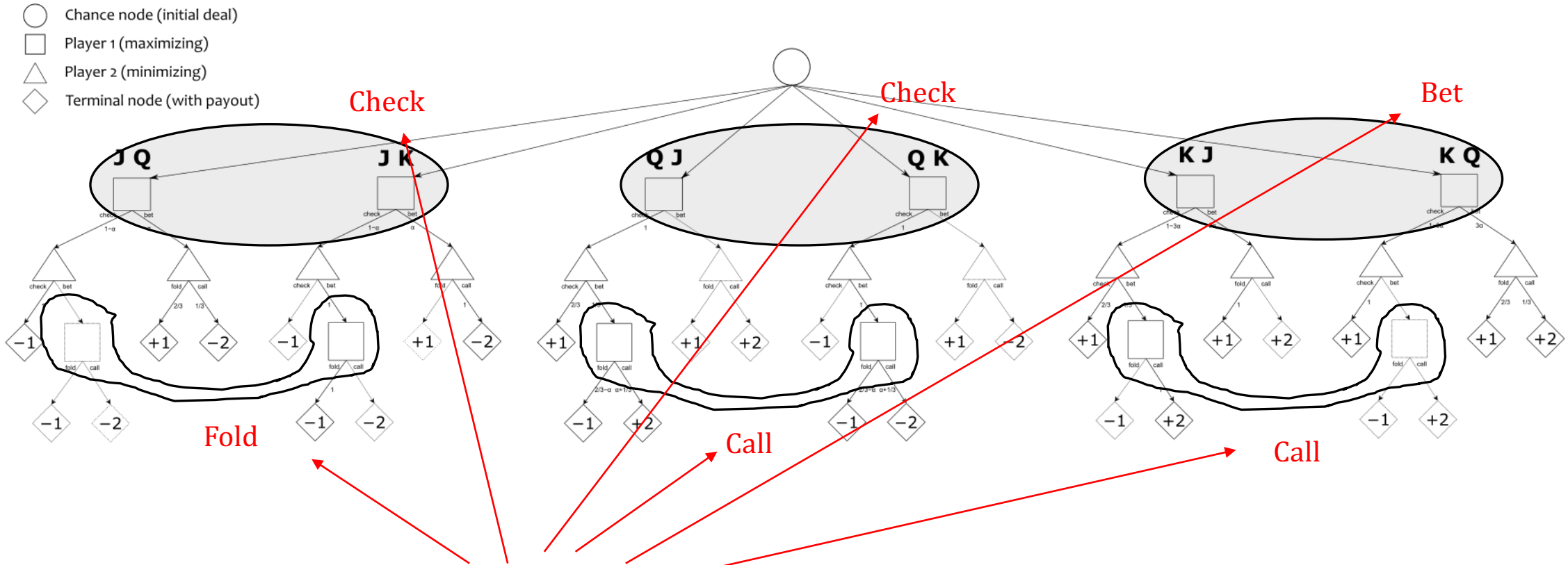
Kuhn Poker revisited

What are the normal form strategies for Player 1?



Kuhn Poker revisited (II)

6 infosets, each with 2 actions →2^6=64 actions

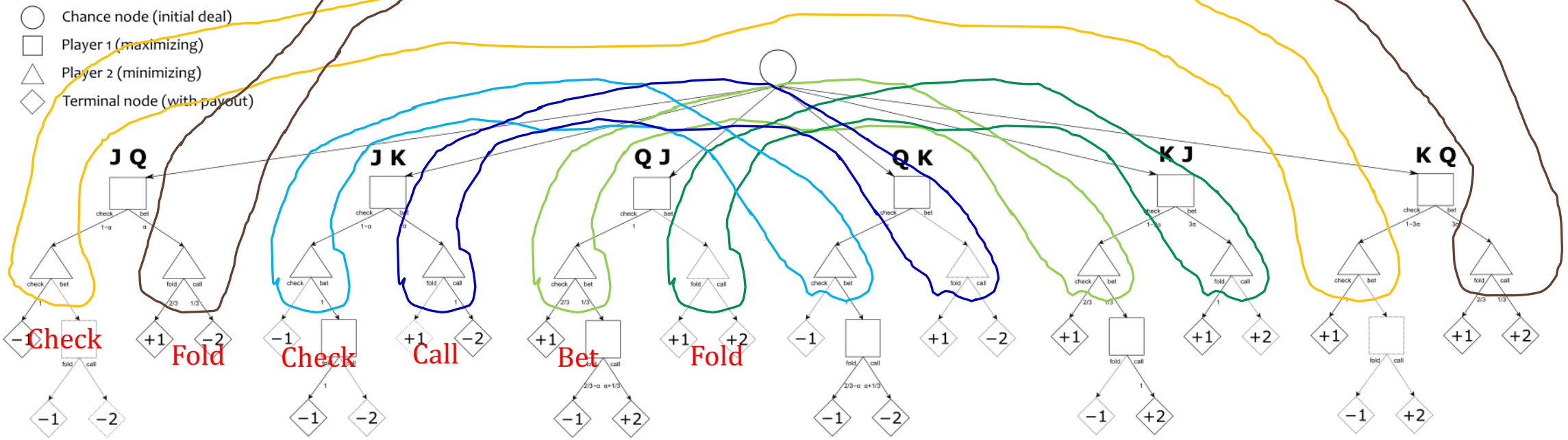


One action
(dictionary which tells player what to do at <u>every</u> infoset)

Kuhn Poker revisited (III)

What are the infosets for Player 2?

What are the normal form strategies for Player 2?



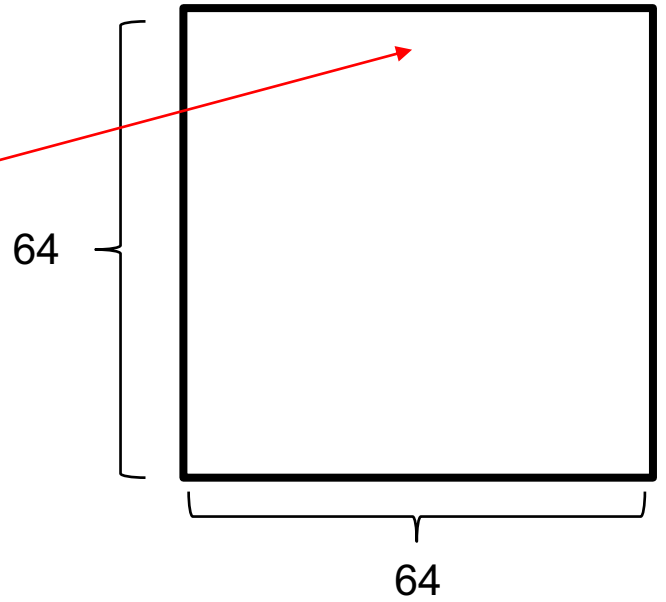
Payoffs under normal form games

Player vertices now become deterministic.

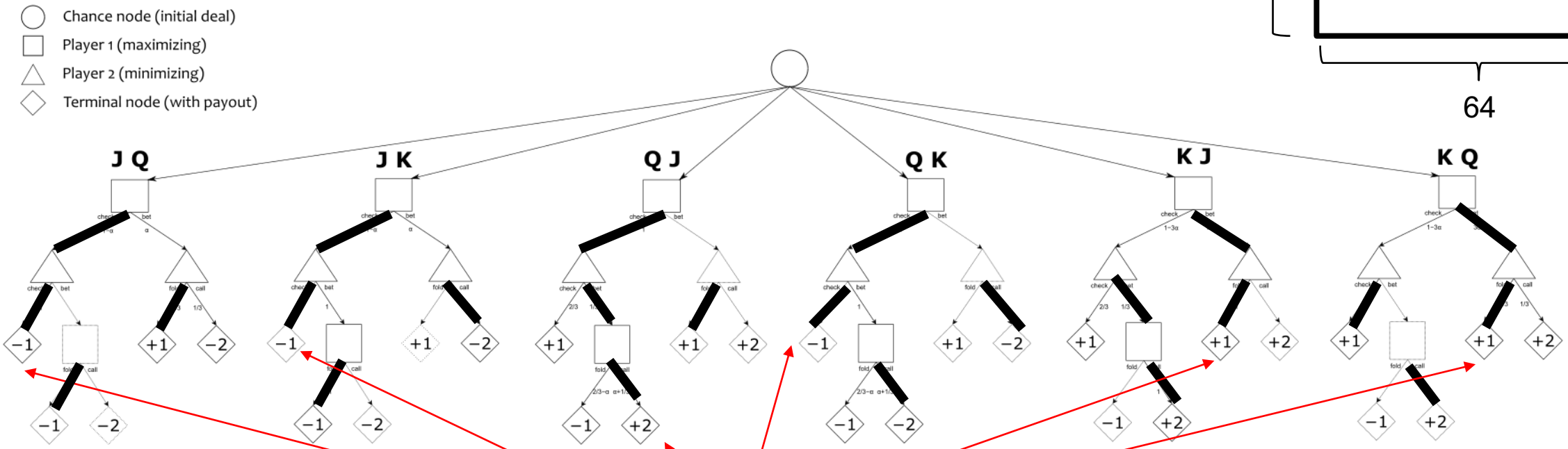
- Traverse according to chance vertices

Example: CCBFCC v.s. CFCCBF

- (-1-1+2-1+1+1)/6=1/6



One of the 64*64 entries in the payoff matrix



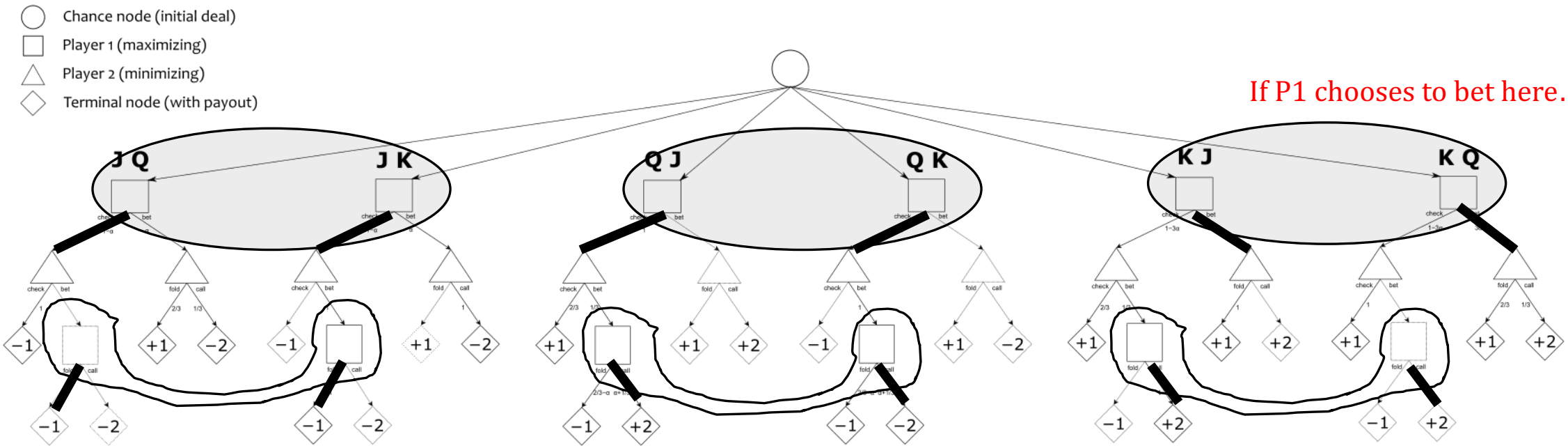
Outcomes, equally weighted (based on the root chance vertex)

The reduced normal form

Performing some actions earlier → some infosets no longer important

Example with Kuhn Poker, Player 1

- CCBFCC and CCBFCF → [**J**: Check-Fold, **Q**: Check-Fold, **K**: Bet]. $3^3 = 27$ actions!
- In literature, will be written as CCBFC*, * denotes any action



More on the reduced normal form

Will trim off more when game tree is very deep

Extreme case, only one player, no chance

- Player simply chooses which leaf it wants

Always applicable, no assumptions on perfect recall yet

But #actions can **still be exponential**

- When many parallel information sets, still need cartesian product, e.g., Player 2
- E.g., what if there were 100 cards?

More on the reduced normal form

Will trim off more when game tree is very deep

Extreme case, only one player, no chance

- Player simply chooses which leaf it wants

Always applicable, no assumptions on perfect recall yet

But #actions can **still be exponential**

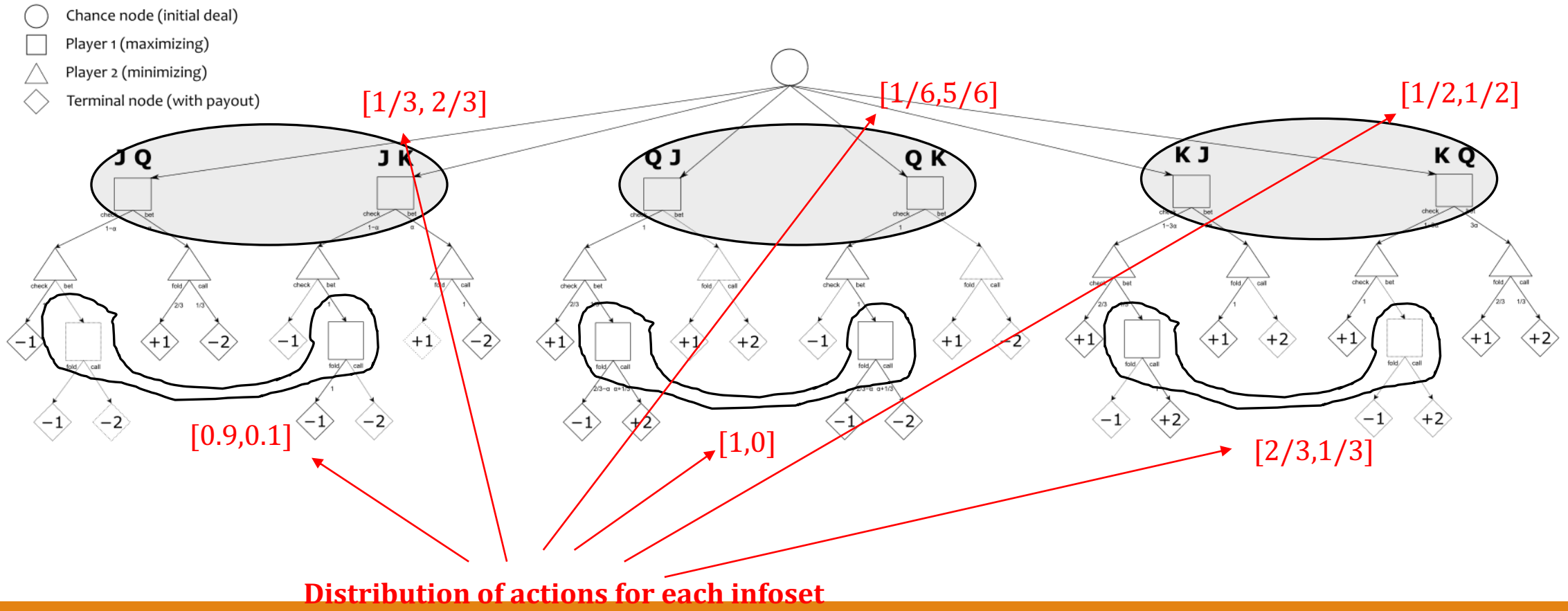
- When many parallel information sets, still need cartesian product, e.g., Player 2
- E.g., what if there were 100 cards?

Warning: we can remove "duplicated actions" since those were payoff equivalent and our choice of equilibrium concept was "nice"

- Under other equilibrium concepts (especially those with bounded rationality, e.g., Quantal response equilibrium), this will change the set of equilibrium
- QRE of the normal form game will favour actions in deeper branches of tree as compared to reduced normal form

Method 2: Behavioral Strategies

- Normal form: randomization is done ex-ante, draw from a distribution of "dictionaries", but after that, just follow dictionary blindly
- Behavioral strategy is more natural: distribution over actions **locally** for each infoset



Kuhn's Theorem

Under perfect recall, the space of behavioral strategies and simplex over normal-form strategies is payoff (strategically) equivalent

- Only need to consider behavioral strategies
- Much smaller in dimensions! If $|a|$ actions per infoset and $|I|$ infosets, strategy is a vector of length $|a| \cdot |I|$ rather than $|a|^{|I|}$

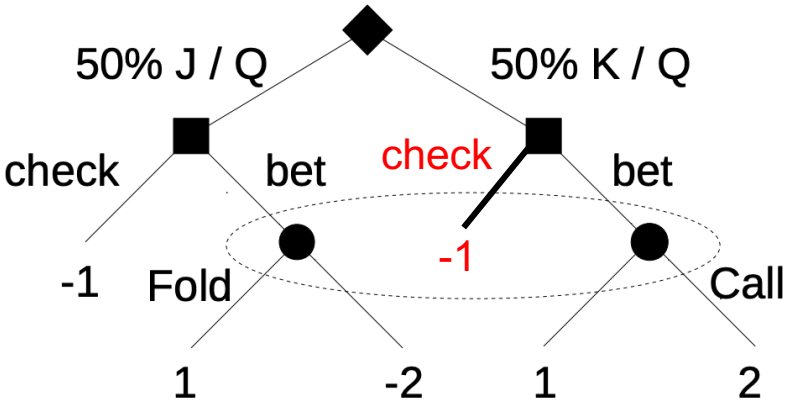
Also much easier to interpret, like MDPs

- Randomly select action when we reach an infostate, rather than sample ex-ante when game starts

Example of Kuhn's Theorem

Player 1 Normal form strategies:

- 4 strategies, CC, CB, BC, BB
  - J: Check-K: Check
  - J: Check-K: Bet
  - J: Bet-K: Check
  - J: Bet-K: Bet
- Strategy is of the form [P(CC), P(CB), P(BC), P(BB



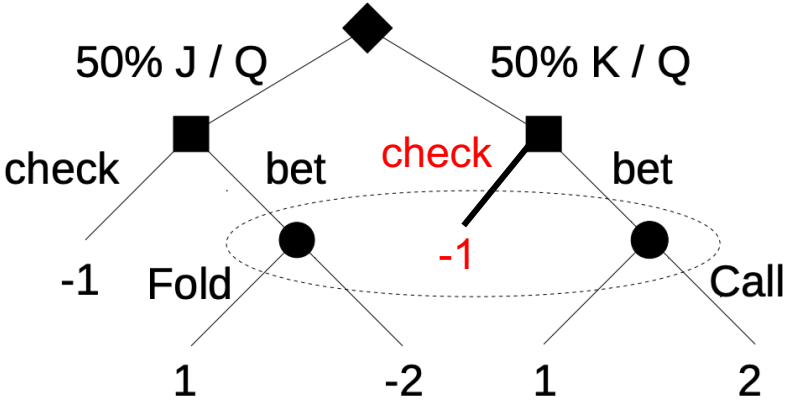
Player 1 Behavioral strategies are

- [P(C|Jack drawn), P(B|Jack drawn), P(C|King drawn), P(B|King drawn)]

Example of Kuhn's Theorem

Player 1 Normal form strategies:

- 4 strategies, CC, CB, BC, BB
  - J: Check-K: Check
  - J: Check-K: Bet
  - J: Bet-K: Check
  - J: Bet-K: Bet
- Strategy is of the form [P(CC), P(CB), P(BC), P(BB



Player 1 Behavioral strategies are

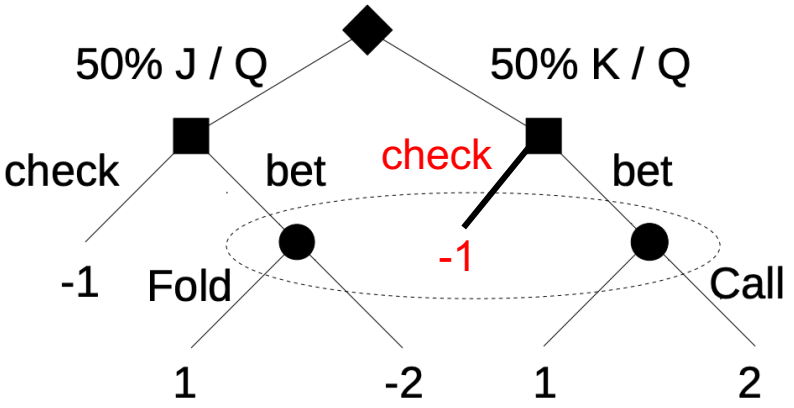
- [P(C|Jack drawn), P(B|Jack drawn), P(C|King drawn), P(B|King drawn)]

How do we go from Normal Form → Behavioral Strategy?

Example of Kuhn's Theorem

Player 1 Normal form strategies:

- 4 strategies, CC, CB, BC, BB
  - J: Check-K: Check
  - J: Check-K: Bet
  - J: Bet-K: Check
  - J: Bet-K: Bet
- Strategy is of the form [P(CC), P(CB), P(BC), P(BB



Player 1 Behavioral strategies are

- [P(C|Jack drawn), P(B|Jack drawn), P(C|King drawn), P(B|King drawn)]

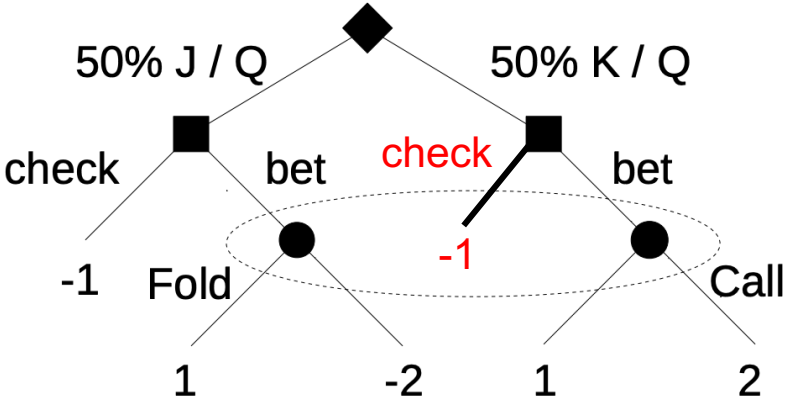
How do we go from Normal Form → Behavioral Strategy?

How do we go from Behavioral Form → Normal Form Strategy?

Example of Kuhn's Theorem

Player 1 Normal form strategies:

- 4 strategies, CC, CB, BC, BB
  - J: Check-K: Check
  - J: Check-K: Bet
  - J: Bet-K: Check
  - J: Bet-K: Bet
- Strategy is of the form [P(CC), P(CB), P(BC), P(BB



Player 1 Behavioral strategies are

- [P(C|Jack drawn), P(B|Jack drawn), P(C|King drawn), P(B|King drawn)]

How do we go from Normal Form → Behavioral Strategy?

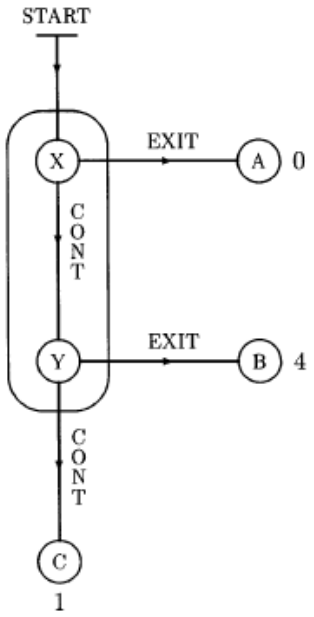
How do we go from Behavioral Form → Normal Form Strategy?

Not a 1-1 mapping, NF strategies can map to same behavioral one

Example 1 of why PR is important

In imperfect recall games with absentmindedness, i.e., the case where paths go through some infoset twice

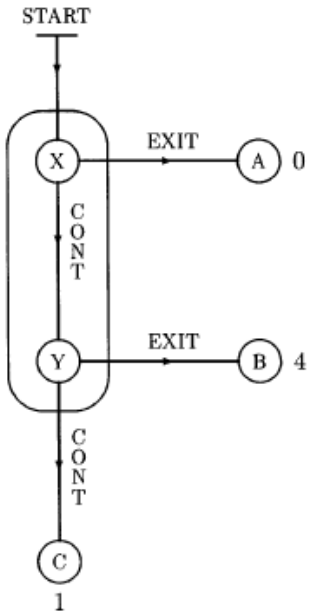
- Exiting at B is impossible for normal form strategies
  - If action is to exit, then we will end up at A. If action is to continue, then end up at C.
- For behavioral strategies, there are at least two interpretations
  - Sample each action at every infoset based on behavioral strategy at the start of game
  - OR, sample an action at infoset “online”, each time we reach it



Example 1 of why PR is important

In imperfect recall games with absentmindedness, i.e., the case where paths go through some infoset twice

- Exiting at B is impossible for normal form strategies
  - If action is to exit, then we will end up at A. If action is to continue, then end up at C.
- For behavioral strategies, there are at least two interpretations
  - Sample each action at every infoset based on behavioral strategy at the start of game
  - OR, sample an action at infoset "online", each time we reach it
- The first interpretation can never end up at B.
- The second one does so with probability p(1-p)
- There is no "right" or "wrong" interpretation here
  - Matter of defining what a "strategy" is
  - Most people coming from AI choose the second interpretation

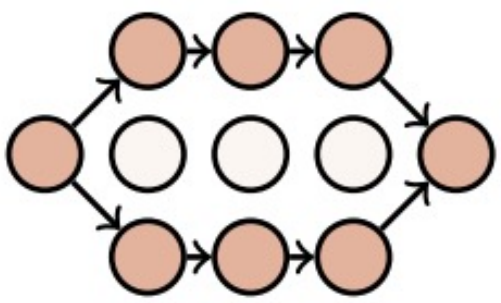


Example 2 of why PR is important

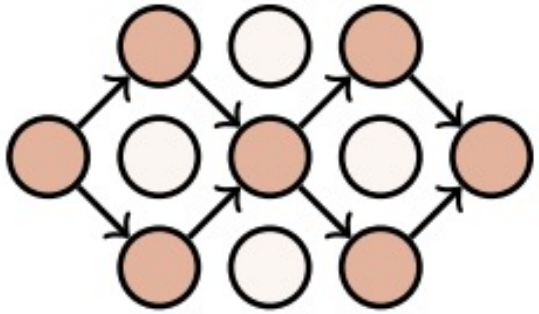
IR makes it such that there is some “low-rank” constraint

From Lecture 3: Professor pursuing a student over T steps

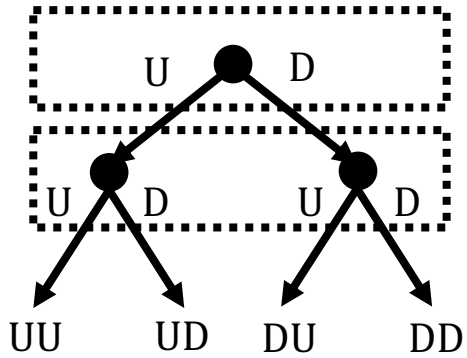
- If professor meets student, student will be assigned work
- Number of times met doesn’t matter, just binary



Professor



Student



Student’s perspective

Example 2 of why PR is important

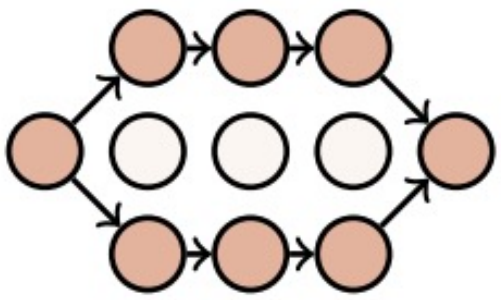
IR makes it such that there is some "low-rank" constraint

From Lecture 3: Professor pursuing a student over T steps

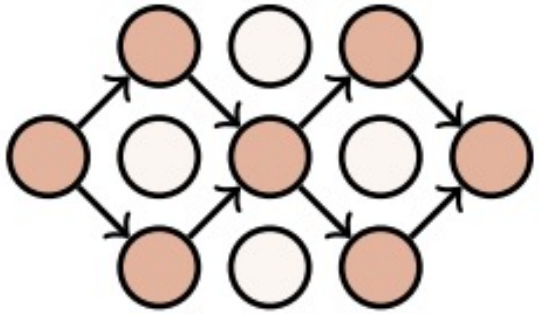
- If professor meets student, student will be assigned work
- Number of times met doesn't matter, just binary

NE under perfect recall is for student is to go UU, DD w.p. 0.5

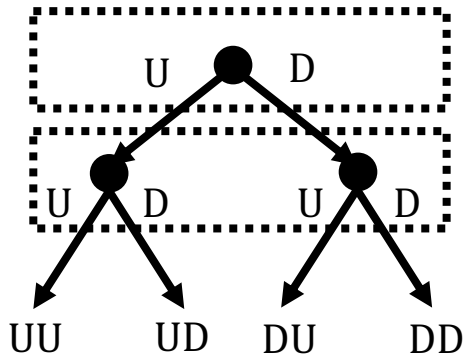
- Can be obtained by normal form strategies
- **Cannot** be obtained by behavioral strategies!



Professor



Student



Student's perspective

The Limitation of Behavioral Strategies

Probability of reaching leaf =

- Product of player 1 action probabilities along path to leaf ×
- Product of player 2 action probabilities along path to leaf ×
- Product of chance probabilities along path to leaf

NOT bilinear, cannot write utilities in the form $x^T Ay$ where x, y are behavioral strategies

Nonconvex in this form, not as useful for computation

Summary:

- Normal form is useful for game solving, but too big
- Behavioral form is small, but not as useful for game solving

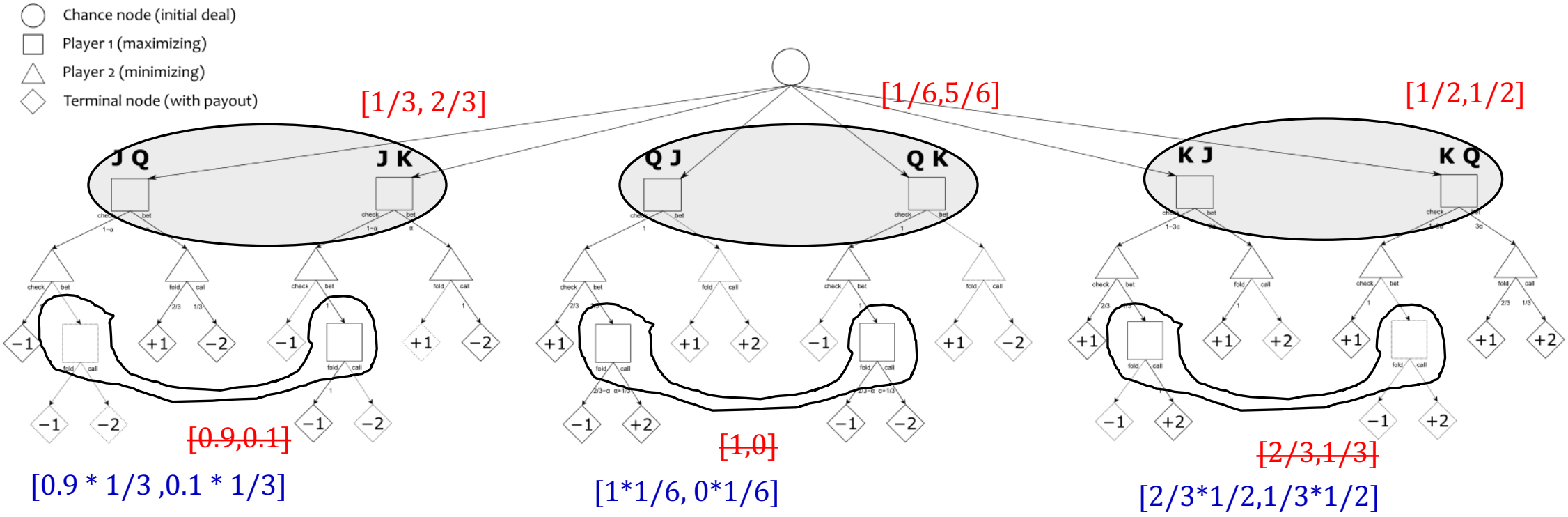
Sequence form: try to get the best of both worlds

Sequence Form and Treeplexes

Method 3: Sequence Form

Instead of probabilities of actions, use probability of **sequences**

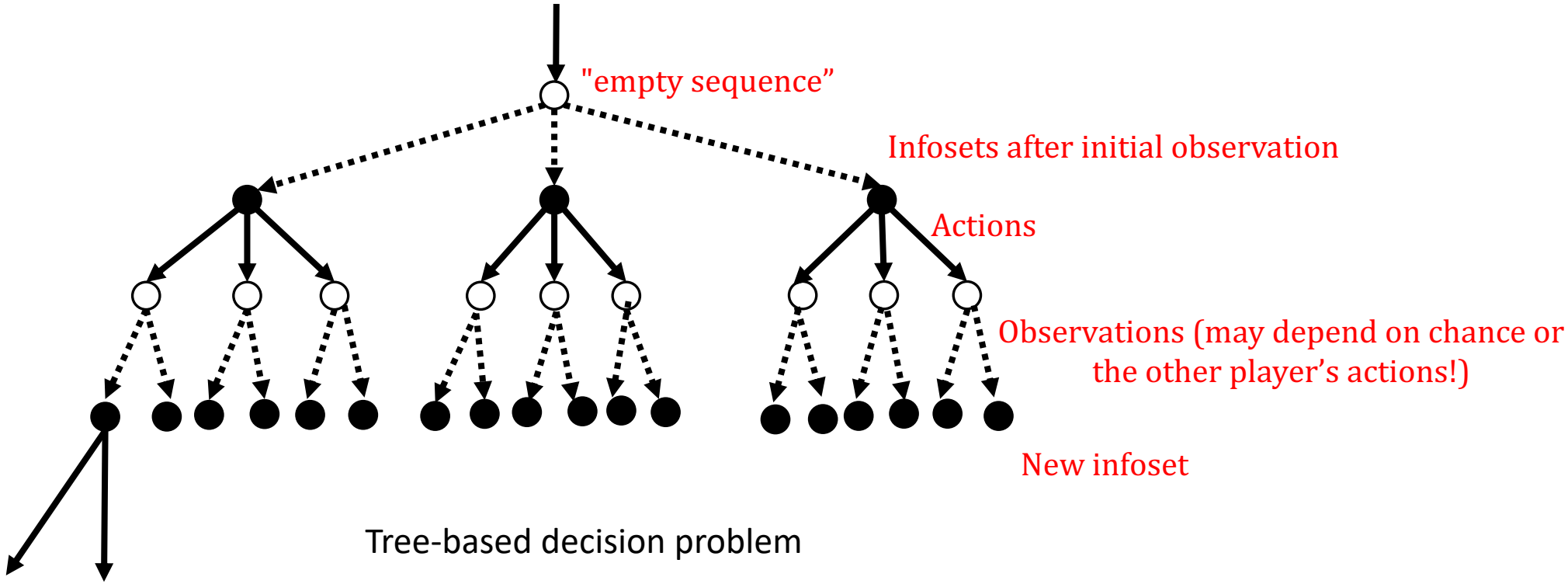
- Sequences already account for probabilities in parent sequences (past actions) taken
- Converting between the 2 is simply a matter of traversing the tree



Treeplexes

Natural strategy space for tree-based decision problems

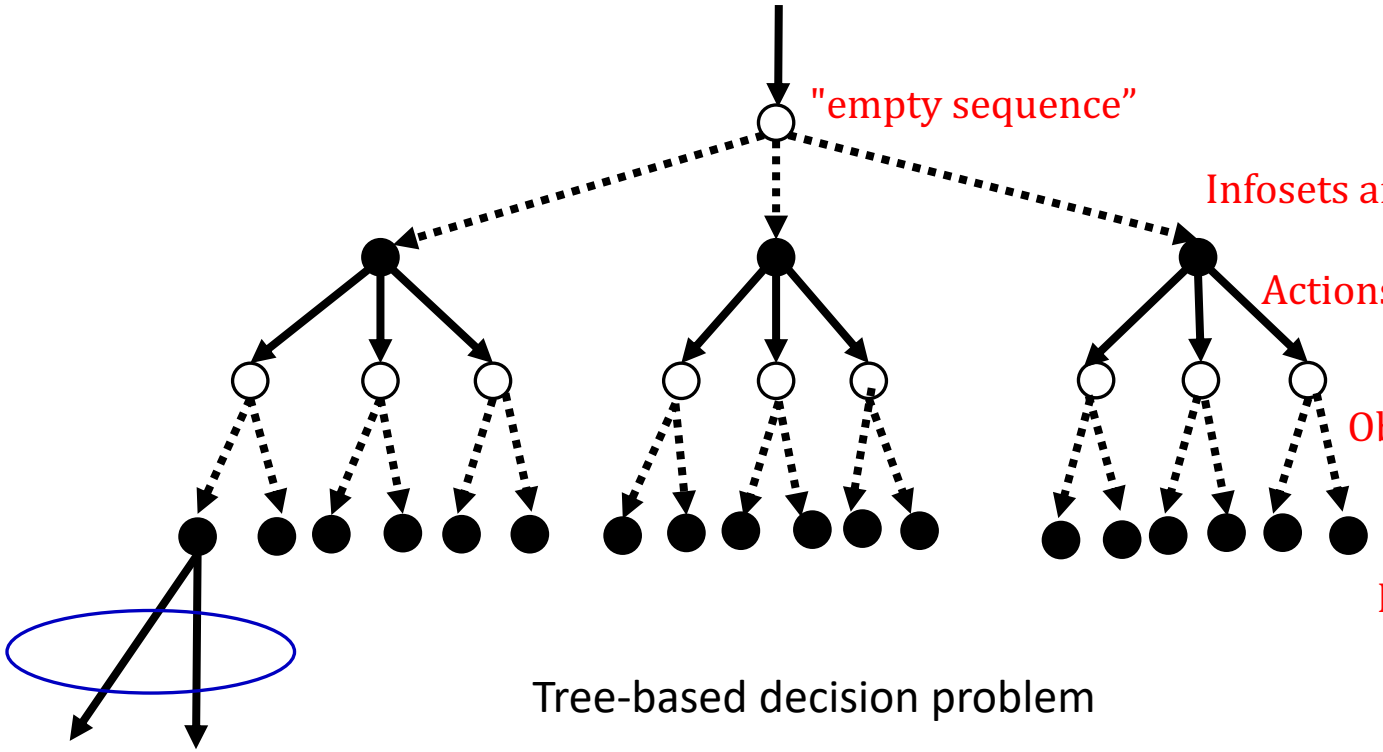
- Recall that assuming PR we end up with a tree-like structure



Treeplexes

Natural strategy space for tree-based decision problems

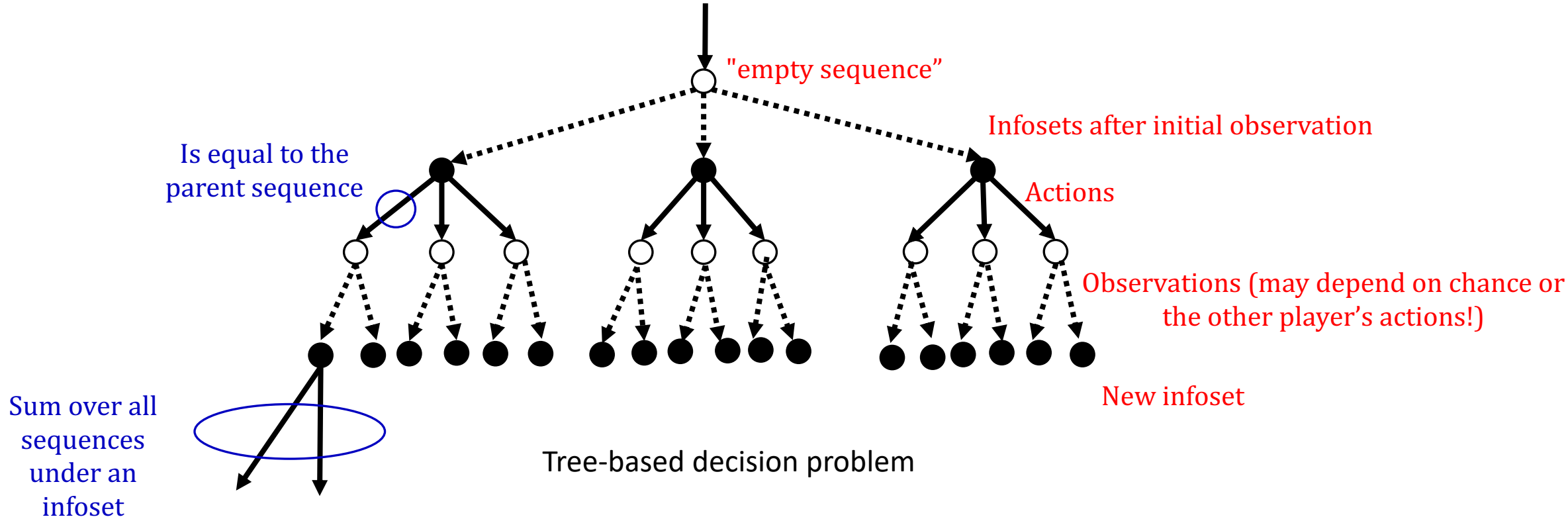
- Recall that assuming PR we end up with a tree-like structure



Treeplexes

Natural strategy space for tree-based decision problems

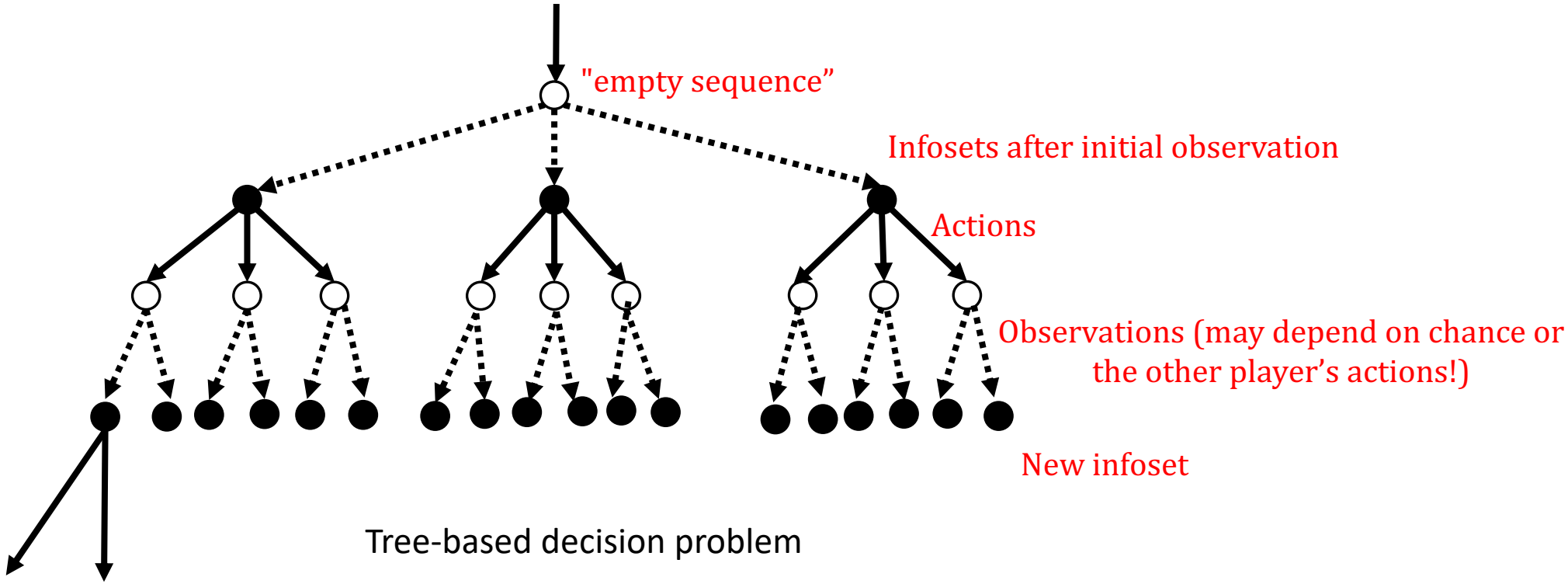
- Recall that assuming PR we end up with a tree-like structure



Treeplexes

Natural strategy space for tree-based decision problems

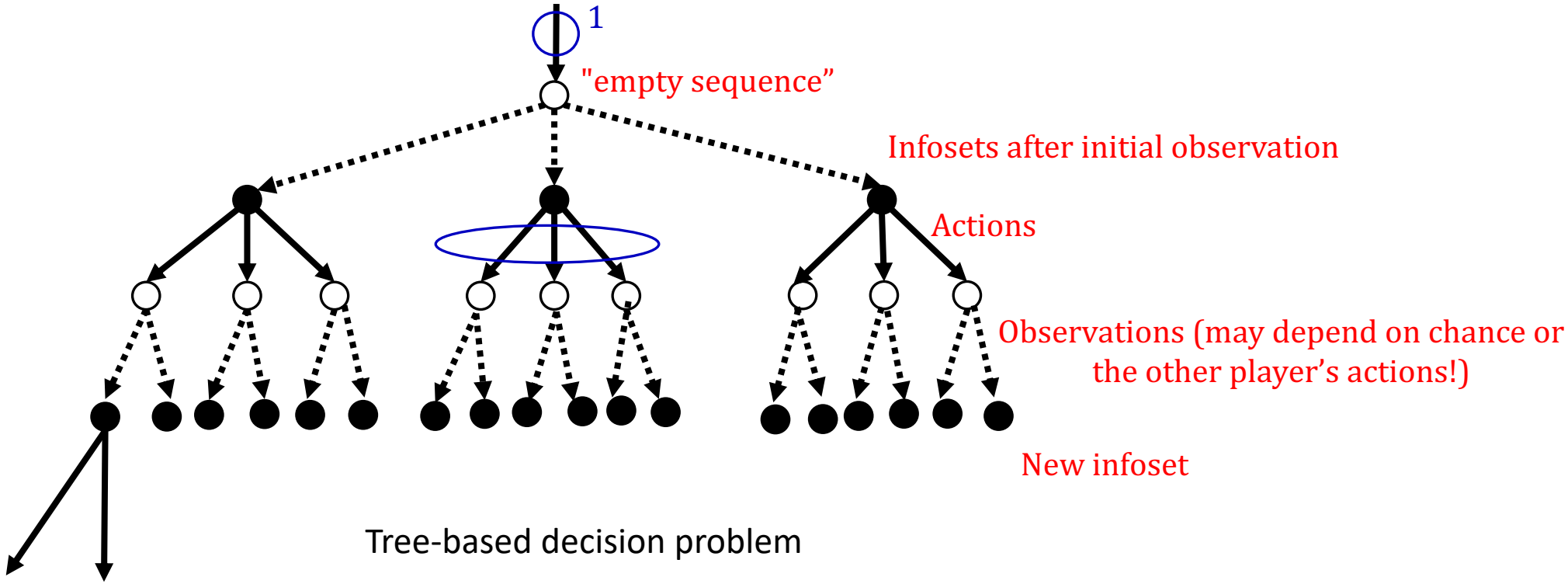
- Recall that assuming PR we end up with a tree-like structure



Treeplexes

Natural strategy space for tree-based decision problems

- Recall that assuming PR we end up with a tree-like structure



Representing a Treeplex as polytope

Instead of the simplex, we use the **treeplex** as domains

- n = number of sequences
- $Ex = e$ gives "these sum-of-children=parent" constraints
  - e is all 0's (for all the "non-root" constraints), except for one entry, where it sums to be parent sequence (which is by default 1)

$$\mathcal{X} = \left\{ x \in \mathbb{R}_+^n \middle| Ex = e \right\}$$

Representing a Treeplex as polytope

Instead of the simplex, we use the **treeplex** as domains

- n = number of sequences
- $Ex = e$ gives "these sum-of-children=parent" constraints
  - e is all 0's (for all the "non-root" constraints), except for one entry, where it sums to be parent sequence (which is by default 1)

$$\mathcal{X} = \left\{ x \in \mathbb{R}^n_+ \middle| Ex = e \right\}$$

Clearly, treeplex is a generalization of the simplex

- Treeplex with one infoset is a simplex

Treeplex is convex, compact

Vertices of Treeplex are pure/deterministic strategies

Solving zero-sum EFGs using LPs

Bilinear Saddle-Point Problem in Simplices

$$\min_{x \in \mathbb{R}^n} \max_{y \in \mathbb{R}^m} x^T A y$$

$$\text{such that } 1^T x = 1^T y = 1$$

$$x, y \geq 0$$

Solving zero-sum EFGs using LPs

Bilinear Saddle-Point Problem in Simplices

$$\min_{x \in \mathbb{R}^n} \max_{y \in \mathbb{R}^m} x^T A y$$

$$\text{such that } 1^T x = 1^T y = 1$$

$$x, y \geq 0$$

Bilinear Saddle-Point Problem in Treeplexes

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} x^T A y$$

*Sequence Form Polytopes

*Sequence Form Payoff Matrix

Solving zero-sum EFGs using LPs

Bilinear Saddle-Point Problem in Simplices

$$\min_{x \in \mathbb{R}^n} \max_{y \in \mathbb{R}^m} x^T A y$$

$$\text{such that } 1^T x = 1^T y = 1$$

$$x, y \geq 0$$

Bilinear Saddle-Point Problem in Treeplexes

$$\min_{x \in \mathcal{X}} \max_{y \in \mathcal{Y}} x^T A y$$

*Sequence Form Polytopes

*Sequence Form Payoff Matrix

Since vertices of treeplex are deterministic strategies, the saddle point is a NE

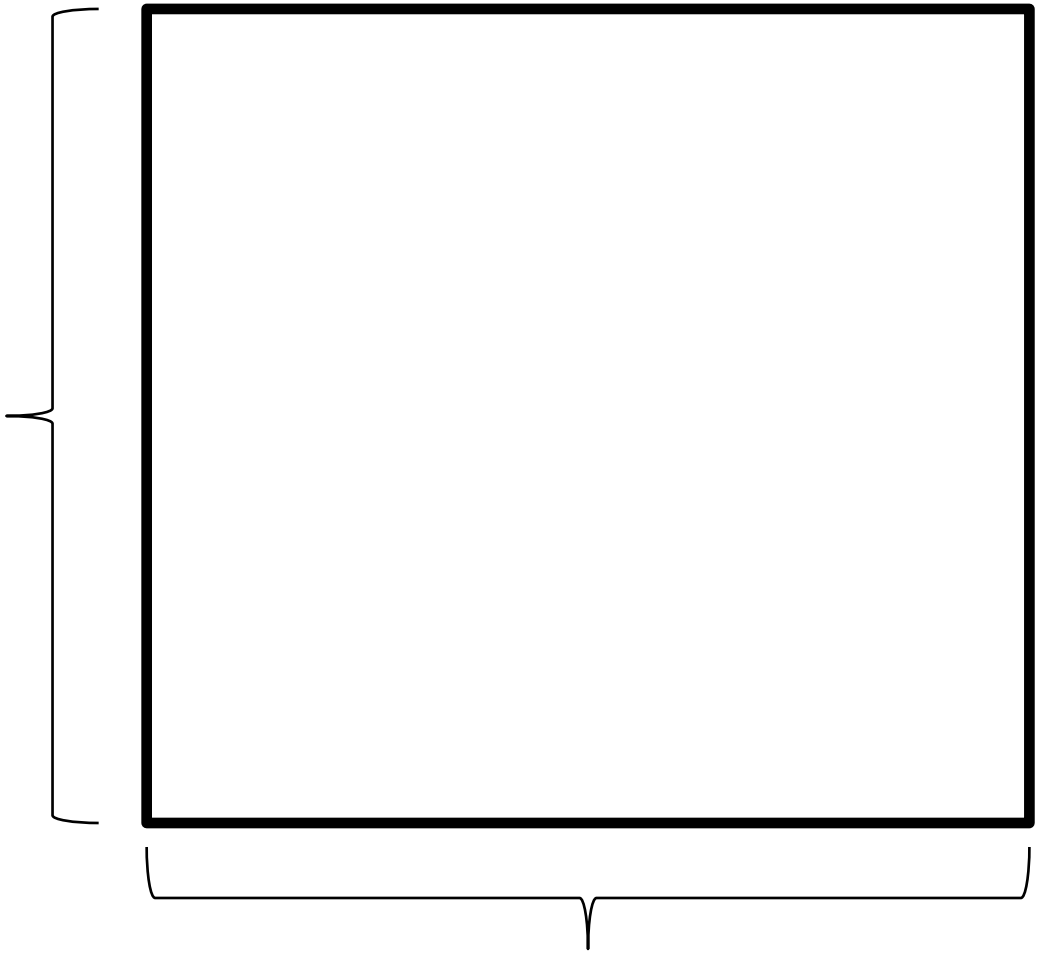
Domains of x, y are themselves polytopes, convex, compact

- Minimax theorem holds

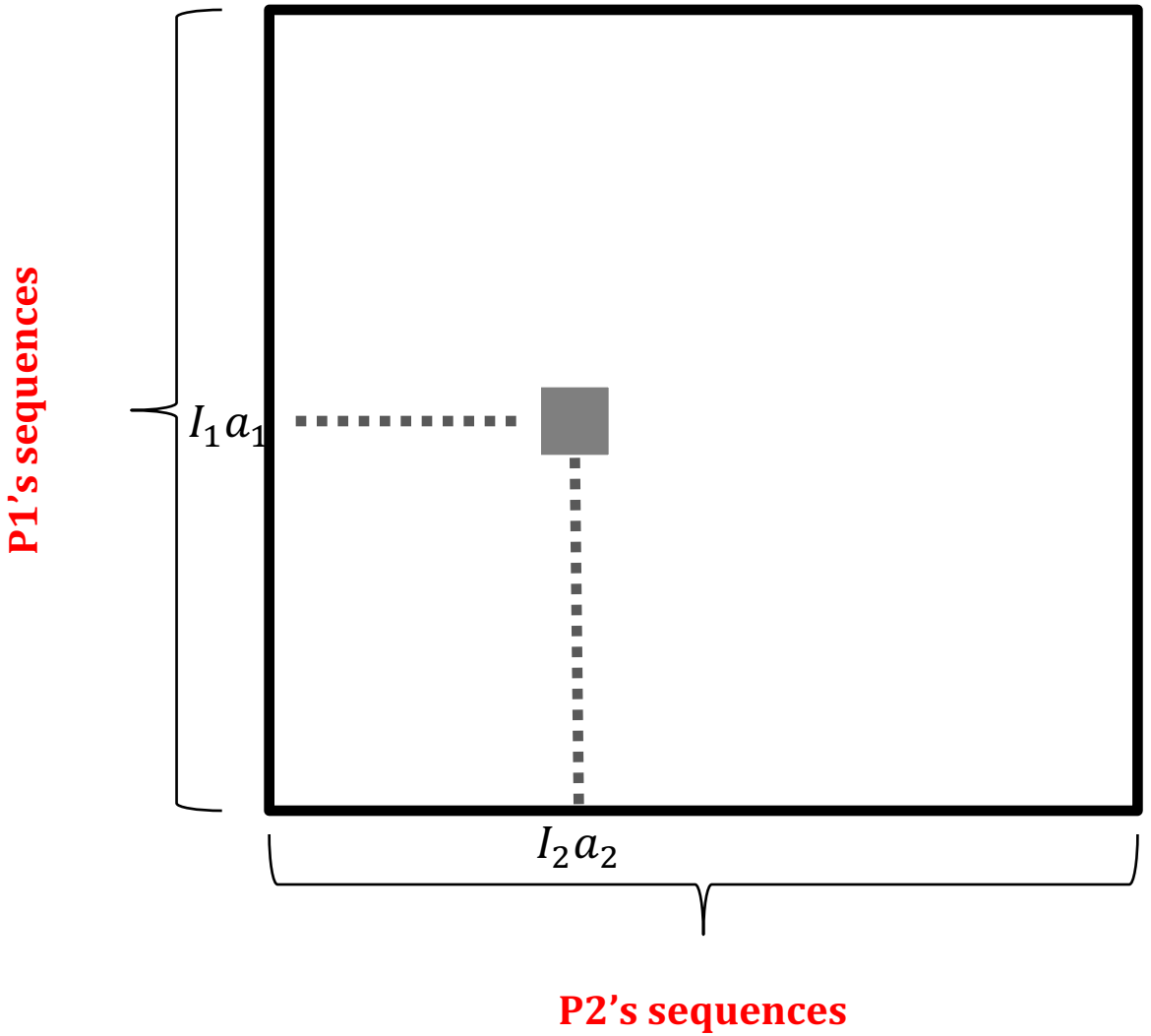
Can find the saddle point the usual way

- Dualize the inner max problem (can be more complicated) to give a min-min problem

The Sequence Form Payoff Matrix

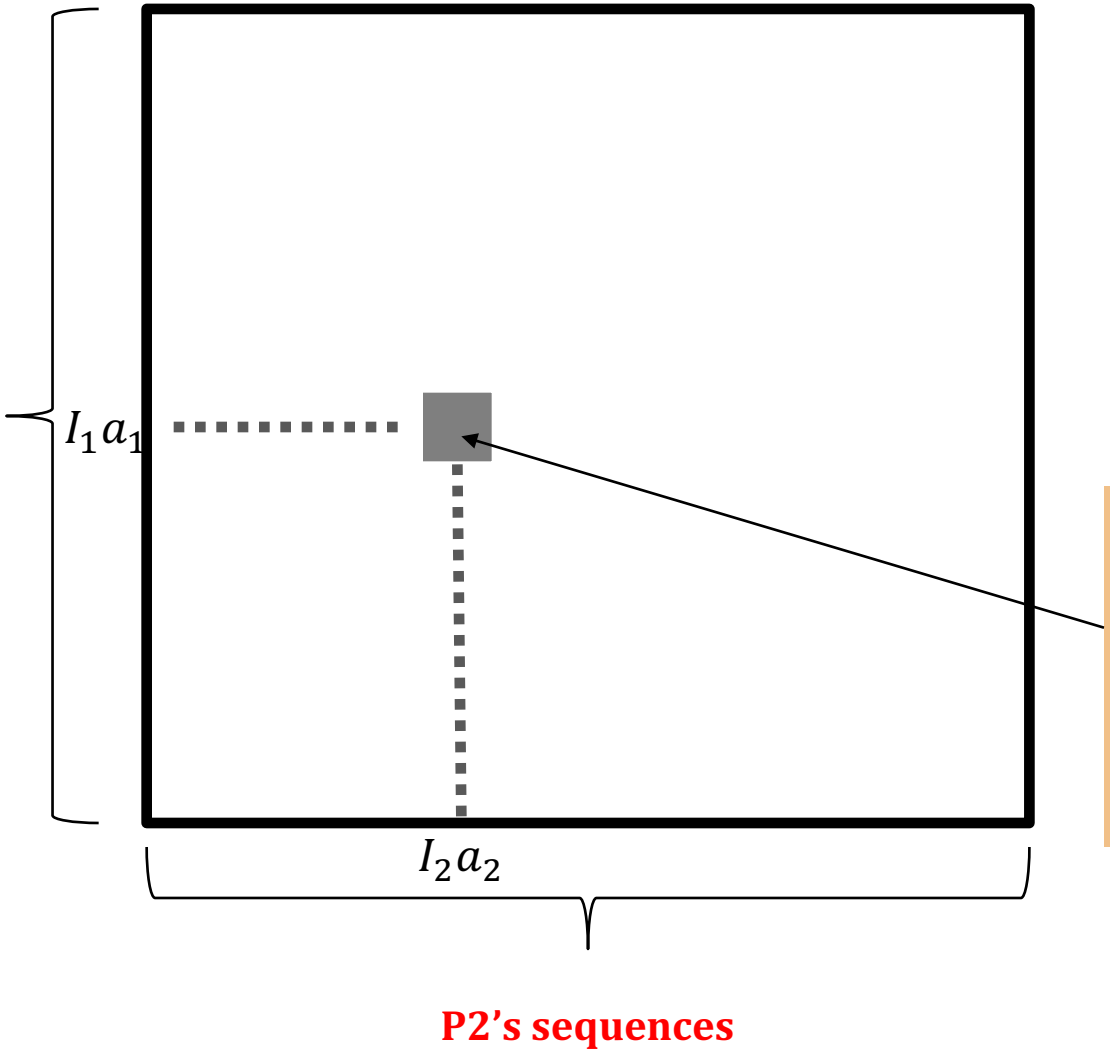


The Sequence Form Payoff Matrix



Recall that the probability of reaching each leaf can be decomposed into P1's, P2's, and nature probabilities

The Sequence Form Payoff Matrix

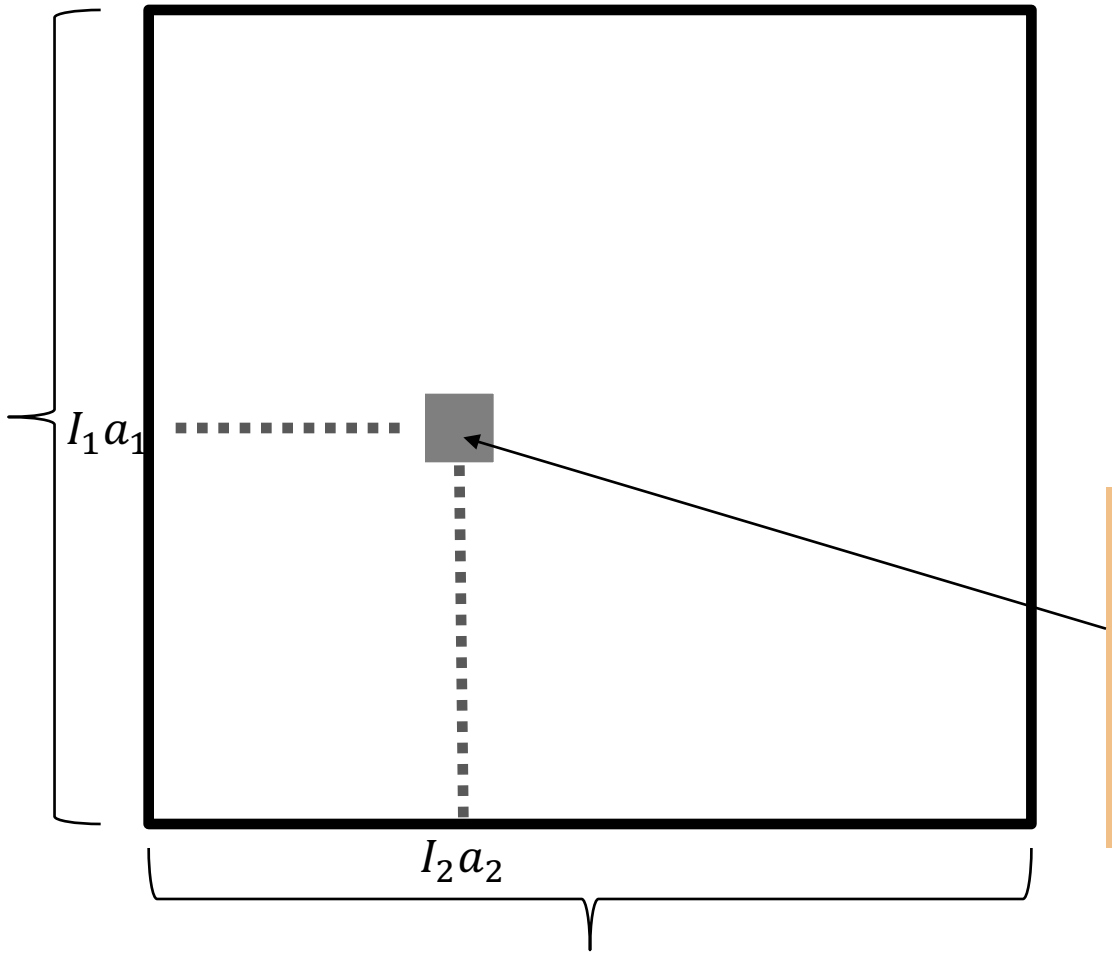


Recall that the probability of reaching each leaf can be decomposed into P1's, P2's, and nature probabilities

Utility of leaf × nature probabilities

Sum over *all* leaves terminating with sequences I_1a_1, I_2a_2.

The Sequence Form Payoff Matrix



Recall that the probability of reaching each leaf can be decomposed into P1's, P2's, and nature probabilities

Utility of leaf × nature probabilities

Sum over *all* leaves terminating with sequences I_1a_1, I_2a_2.

*Sequence Form Payoff Matrix is MUCH smaller and sparser than normal form payoff matrix!

What does the LP look like?

Bilinear Saddle point problem

$$\min_x \max_y x^T A y$$

$$Ex = e, x \geq 0$$

$$Fy = f, y \geq 0$$

What does the LP look like?

Bilinear Saddle point problem

$$\min_x \max_y x^T A y$$
$$Ex = e, x \geq 0$$
$$Fy = f, y \geq 0$$

Take duals of inner max problem

$$\max_y x^T A y$$
$$Fy = f, y \geq 0$$

$$\min_y f^T v$$
$$A^T x \leq F^T v$$

What does the LP look like?

Bilinear Saddle point problem

$$\min_x \max_y x^T A y$$
$$Ex = e, x \geq 0$$
$$Fy = f, y \geq 0$$

Take duals of inner max problem

$$\max_y x^T A y$$
$$Fy = f, y \geq 0$$

$$\min_y f^T v$$
$$A^T x \leq F^T v$$

LP Formulation

$$\min_{x,v} f^T v$$
$$Ex = e, x \geq 0$$
$$A^T x \leq F^T v$$

What does the LP look like?

Bilinear Saddle point problem

$$\min_x \max_y x^T A y$$
$$Ex = e, x \geq 0$$
$$Fy = f, y \geq 0$$

Take duals of inner max problem

$$\max_y x^T A y$$
$$Fy = f, y \geq 0$$

$$\min_y f^T v$$
$$A^T x \leq F^T v$$

LP Formulation

$$\min_{x,v} f^T v$$
$$Ex = e, x \geq 0$$
$$A^T x \leq F^T v$$

What does this mean physically? Hint: compare to the simplex case

What does the LP look like?

Bilinear Saddle point problem

$$\min_x \max_y x^T A y$$
$$Ex = e, x \geq 0$$
$$Fy = f, y \geq 0$$

Take duals of inner max problem

$$\max_y x^T A y$$
$$Fy = f, y \geq 0$$

$$\min_y f^T v$$
$$A^T x \leq F^T v$$

LP Formulation

$$\min_{x,v} f^T v$$
$$Ex = e, x \geq 0$$
$$A^T x \leq F^T v$$

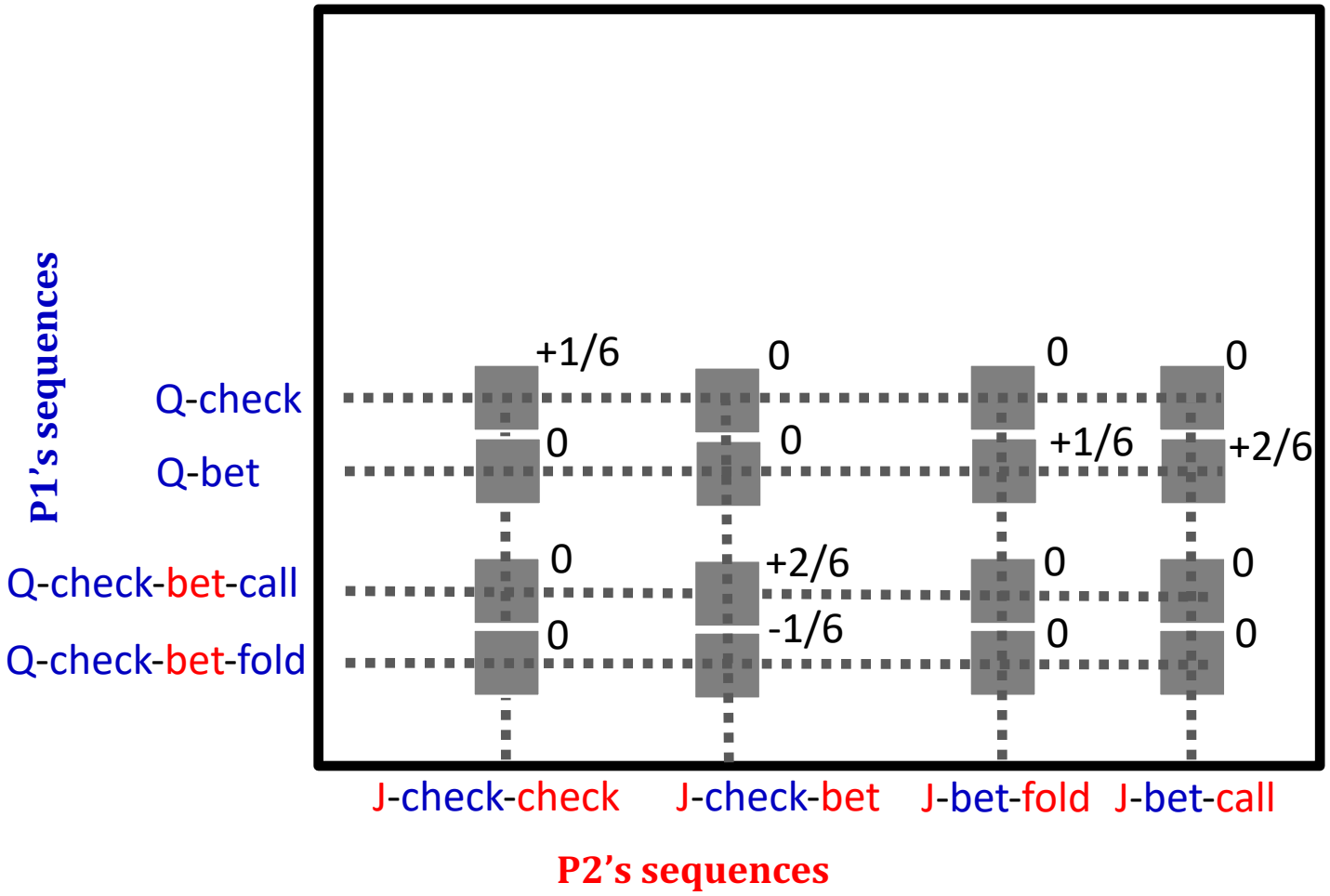
What does this mean physically? Hint: compare to the simplex case

$$\min_{x \in \mathbb{R}^n, V \in \mathbb{R}} V$$
$$\text{such that } 1^T x = 1$$
$$A^T x \leq V$$
$$x \geq 0$$

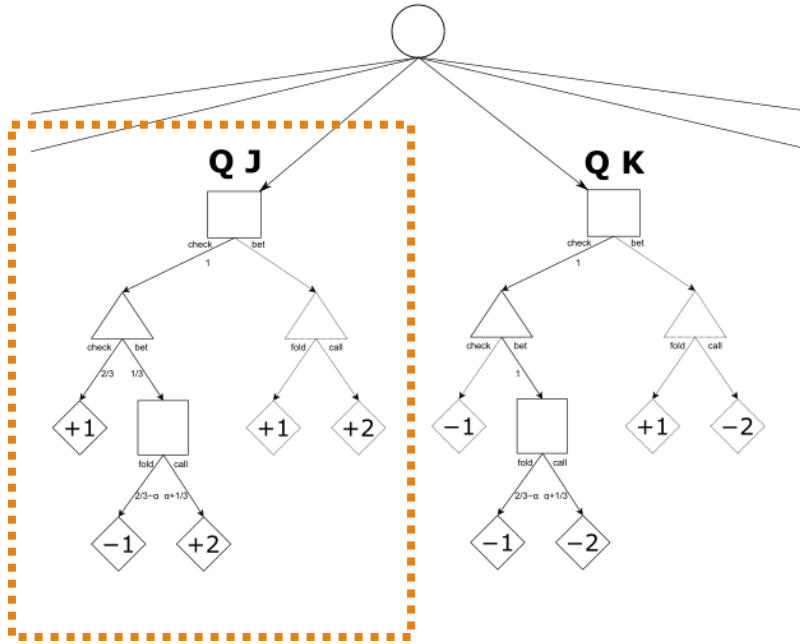
*No matter what max-player does, min player cannot do worse than *V*

Exercise: Sequence form Payoff Matrix for Kuhn Poker

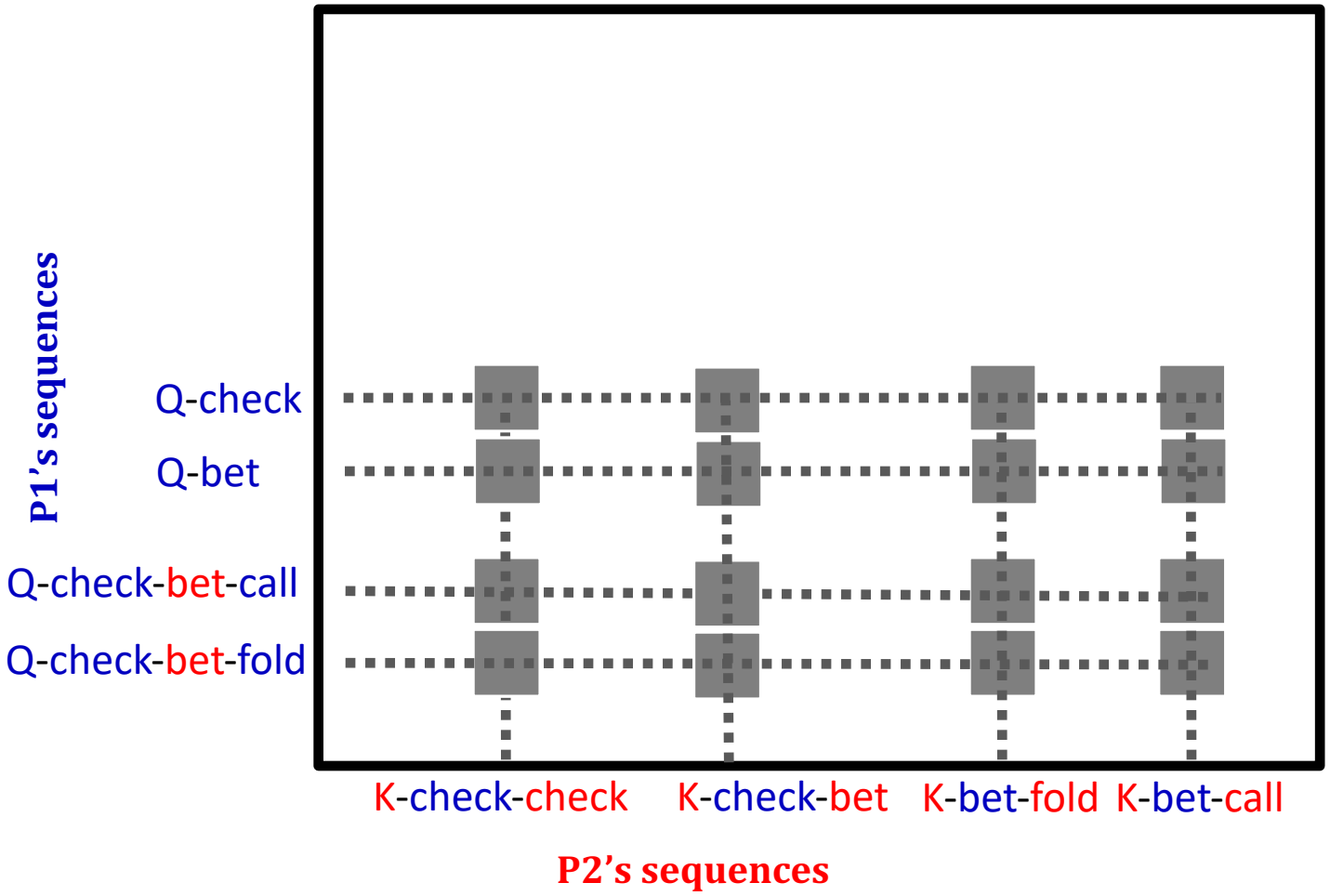
Example: Writing the *A* matrix explicitly

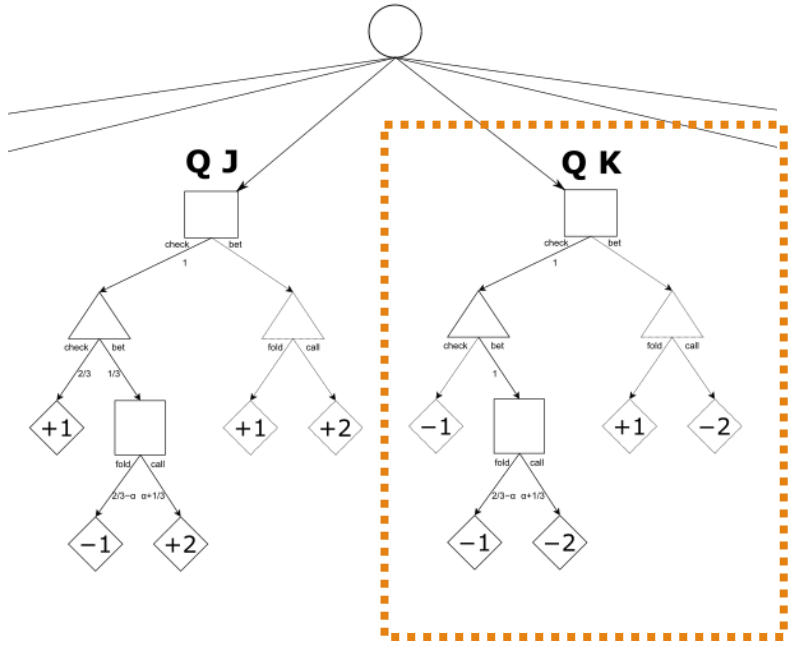


*We typically don't write it explicitly

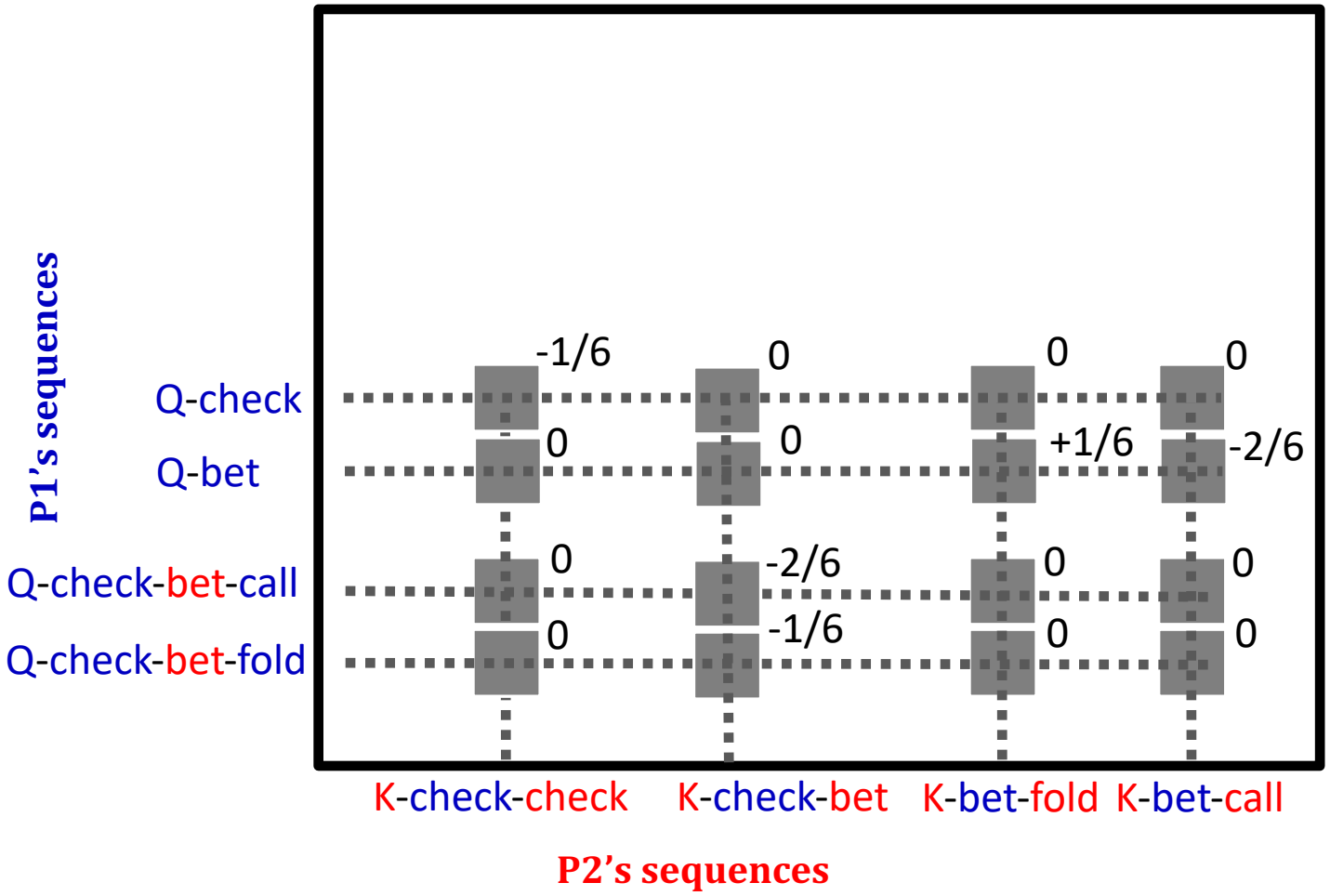


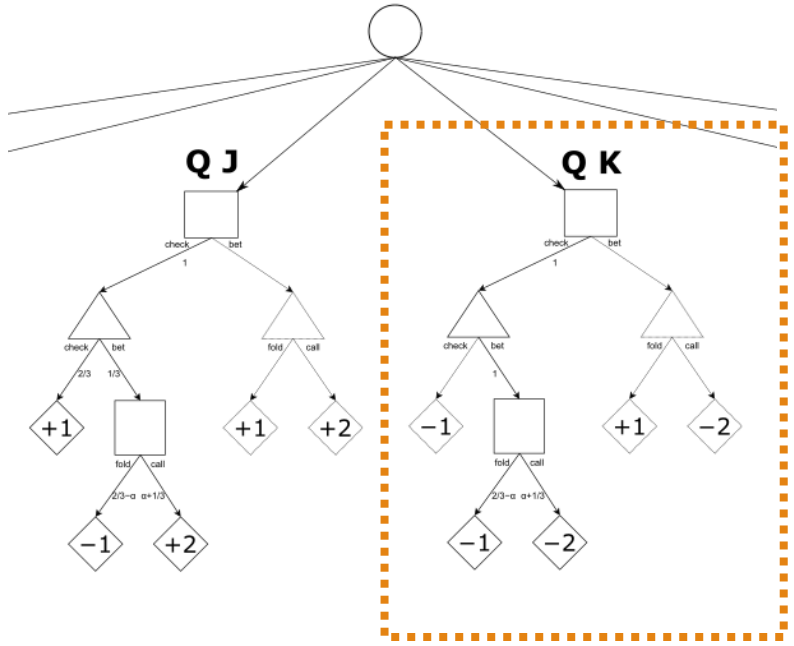
Example: Writing the A matrix explicitly





Example: Writing the *A* matrix explicitly





Does this mean we have poly-time solvers for 2p0s games?

Counterfactual Regret Minimization

Solving large games via regret minimization

LP solvers today are established and robust, but is slow and requires a lot of memory (usually superlinear)

Solving large games via regret minimization

LP solvers today are established and robust, but is slow and requires a lot of memory (usually superlinear)

Can we solve EFGs using self-play and regret minimization

- Just a matter of changing domains from simplex to Treeplex!

Solving large games via regret minimization

LP solvers today are established and robust, but is slow and requires a lot of memory (usually superlinear)

Can we solve EFGs using self-play and regret minimization

- Just a matter of changing domains from simplex to Treeplex!

There are off-the-shelf methods for constructing no-regret learners on general polytopes, but they are very inefficient

- Usually requires some kind of projection
- Recall that we did *not* need to project on the simplex for Hedge/RM
- Hedge can be used to solve generally LPs (technically)
- Treeplex has much more structure

Solving large games via regret minimization

LP solvers today are established and robust, but is slow and requires a lot of memory (usually superlinear)

Can we solve EFGs using self-play and regret minimization

- Just a matter of changing domains from simplex to Treeplex!

There are off-the-shelf methods for constructing no-regret learners on general polytopes, but they are very inefficient

- Usually requires some kind of projection
- Recall that we did *not* need to project on the simplex for Hedge/RM
- Hedge can be used to solve generally LPs (technically)
- Treeplex has much more structure

Can we construct a regret minimizer on a Treeplex?

Naïve method

It is trivially possible to construct a no-regret learner over any polytope if we can enumerate its extreme points (vertices)

Just run some minimizer on the (higher dimensional) simplex and project back onto this primal space

Not very useful… number of vertices is typically much larger than the dimension of the sequence form

- In fact, doing this is the same as converting the game to normal form and solving it

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g., Online Convex Optimization

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g., Online Convex Optimization

- Introduce “local” regret minimizers at each infoset (which have a decision space over the simplex)

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g., Online Convex Optimization

- Introduce “local” regret minimizers at each infoset (which have a decision space over the simplex)
- Show recursively that the regret at each sub-treeplex is bounded by the sum of the regret of each local regret minimizer in its descendant infosets.

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g.,
Online Convex Optimization

- Introduce “local” regret minimizers at each infoset (which have a decision space over the simplex)
- Show recursively that the regret at each sub-treeplex is bounded by the sum of the regret of each local regret minimizer in its descendant infosets.
- Thus, if each local regret minimizer has sublinear regret, the regret over the full treeplex is also sublinear

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g., Online Convex Optimization

- Introduce "local" regret minimizers at each infoset (which have a decision space over the simplex)
- Show recursively that the regret at each sub-treeplex is bounded by the sum of the regret of each local regret minimizer in its descendant infosets.
- Thus, if each local regret minimizer has sublinear regret, the regret over the full treeplex is also sublinear
- Can take average over **sequence form strategies** → converges to NE in zero-sum games

*NOT behavioral strategies. Super common mistake!!!

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Enter CFR (Zinkevich, 2007)

Key ideas

Also responsible for many other related topics, e.g., Online Convex Optimization

- Introduce "local" regret minimizers at each infoset (which have a decision space over the simplex)
- Show recursively that the regret at each sub-treeplex is bounded by the sum of the regret of each local regret minimizer in its descendant infosets.
- Thus, if each local regret minimizer has sublinear regret, the regret over the full treeplex is also sublinear
- Can take average over **sequence form strategies** → converges to NE in zero-sum games

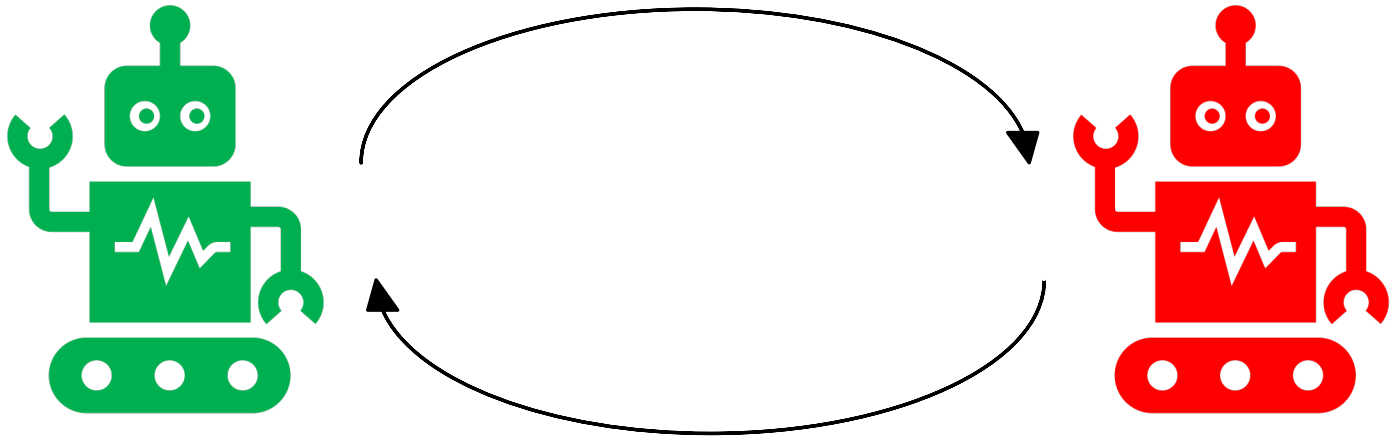
*NOT behavioral strategies. Super common mistake!!!

Warning: a ton of wrong implementations out there…

- The CFR paper itself has a number of typos that can be confusing

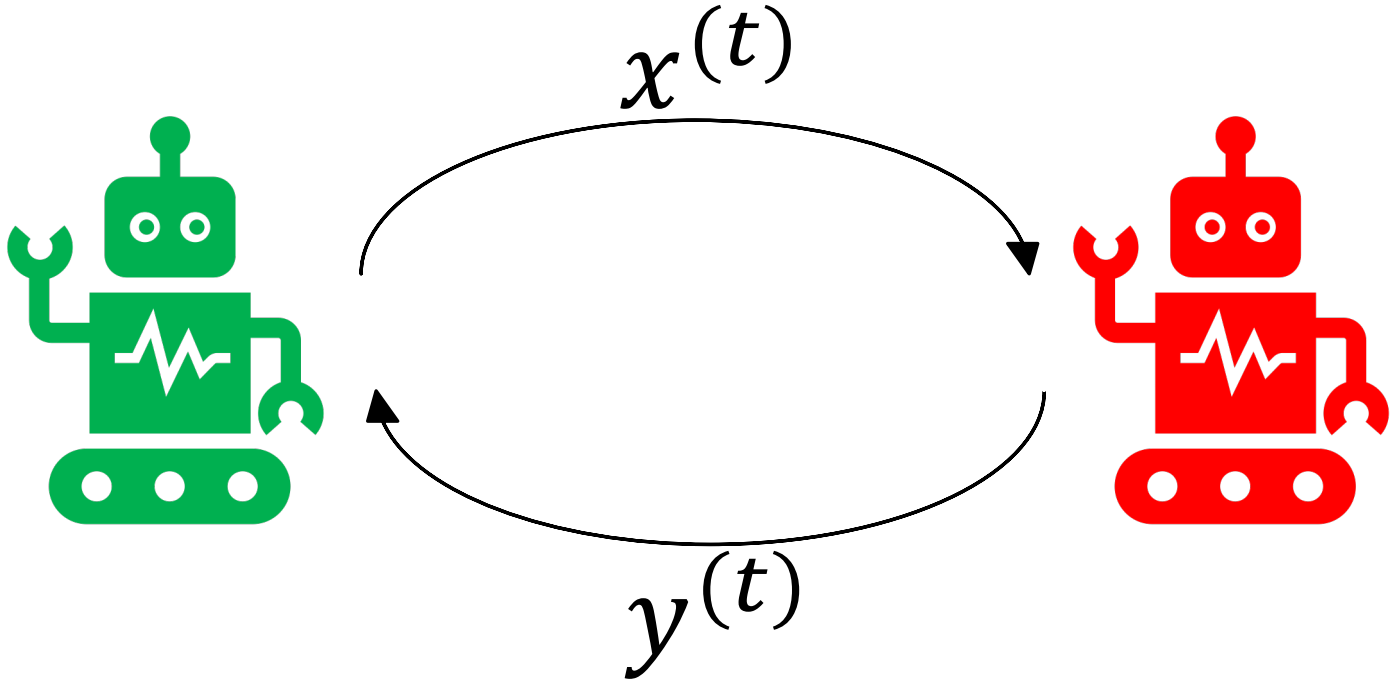
https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

Review: self-play



Review: self-play

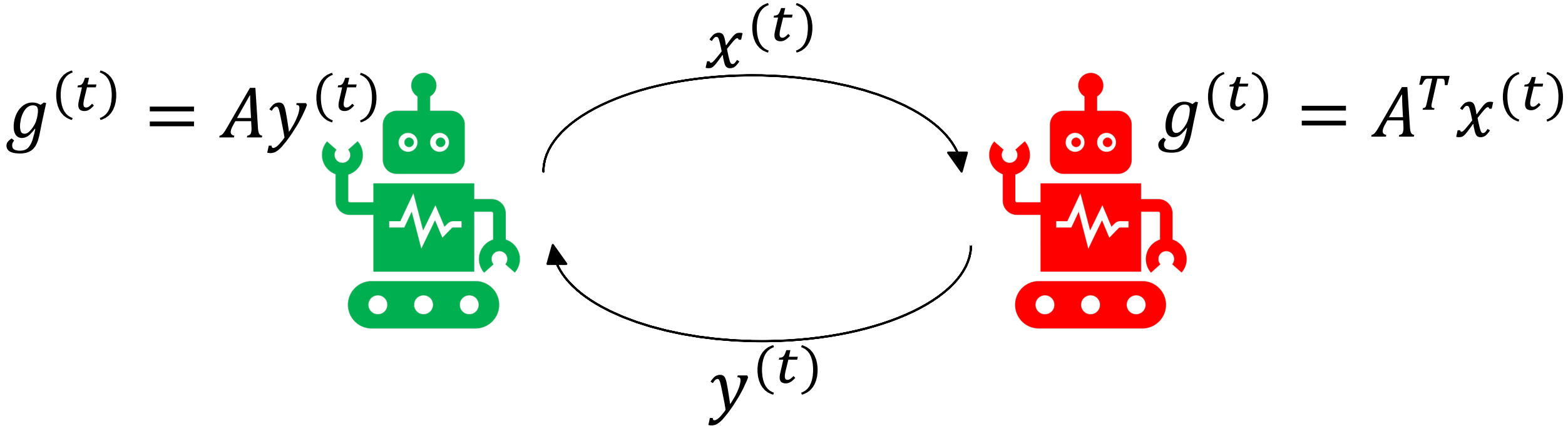
NextStrategy()



Review: self-play

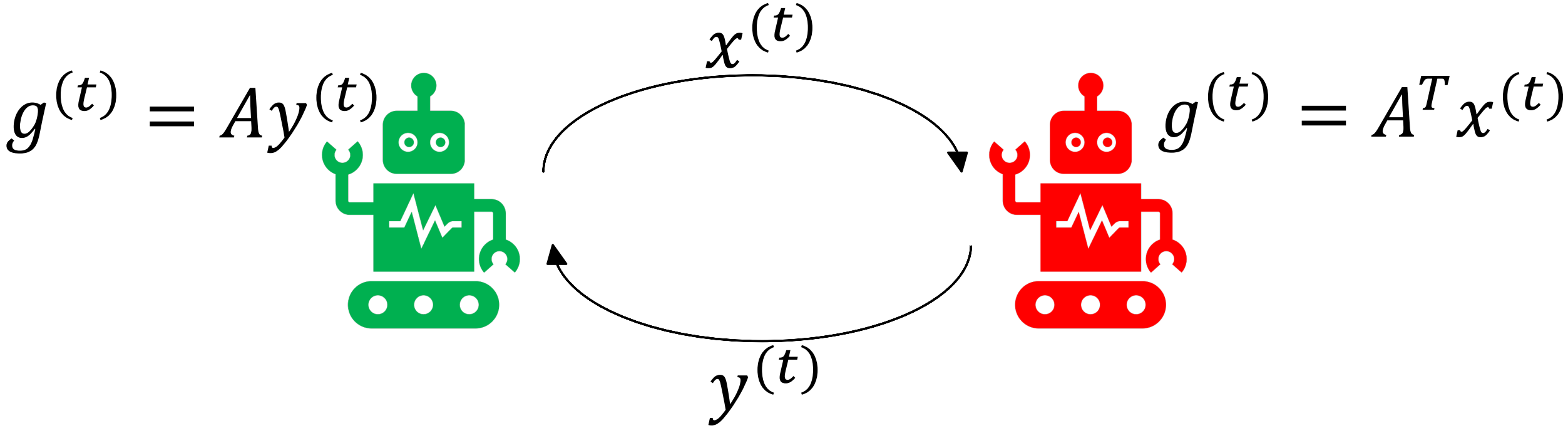
$\textsc{NextStrategy}()$

$\textsc{ObserveLoss}(g^{(t)})$



Review: self-play

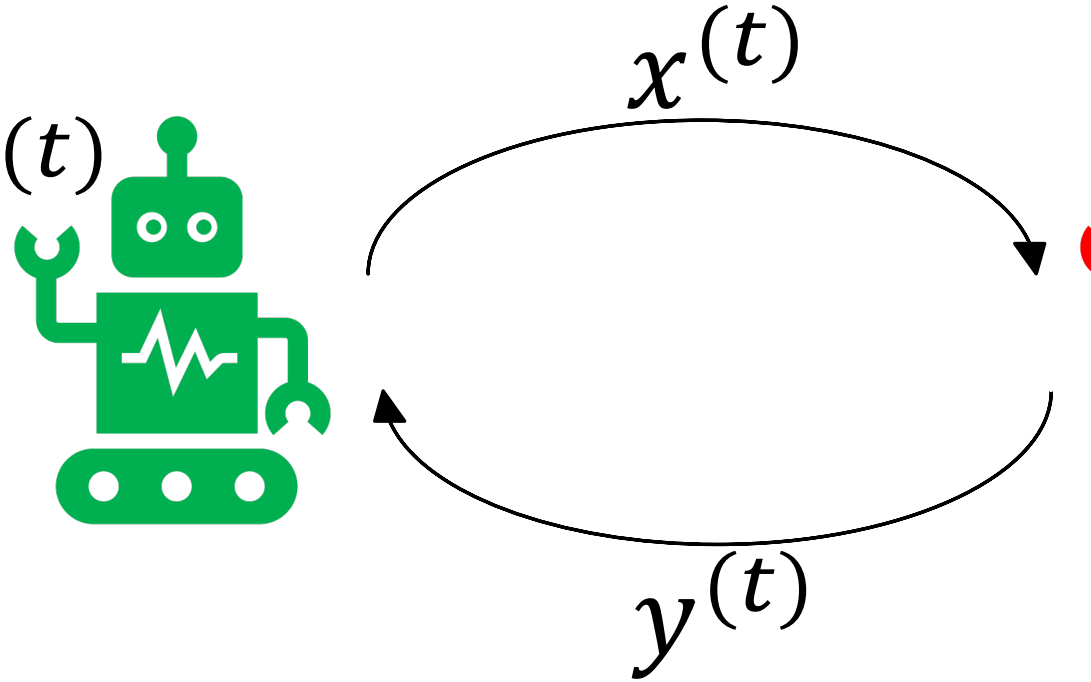
$\textsc{NextStrategy}()$
$\textsc{ObserveLoss}(g^{(t)})$
} Loop



Review: self-play

$\textsc{NextStrategy}()$
$\textsc{ObserveLoss}(g^{(t)})$
} Loop

$g^{(t)} = Ay^{(t)}$



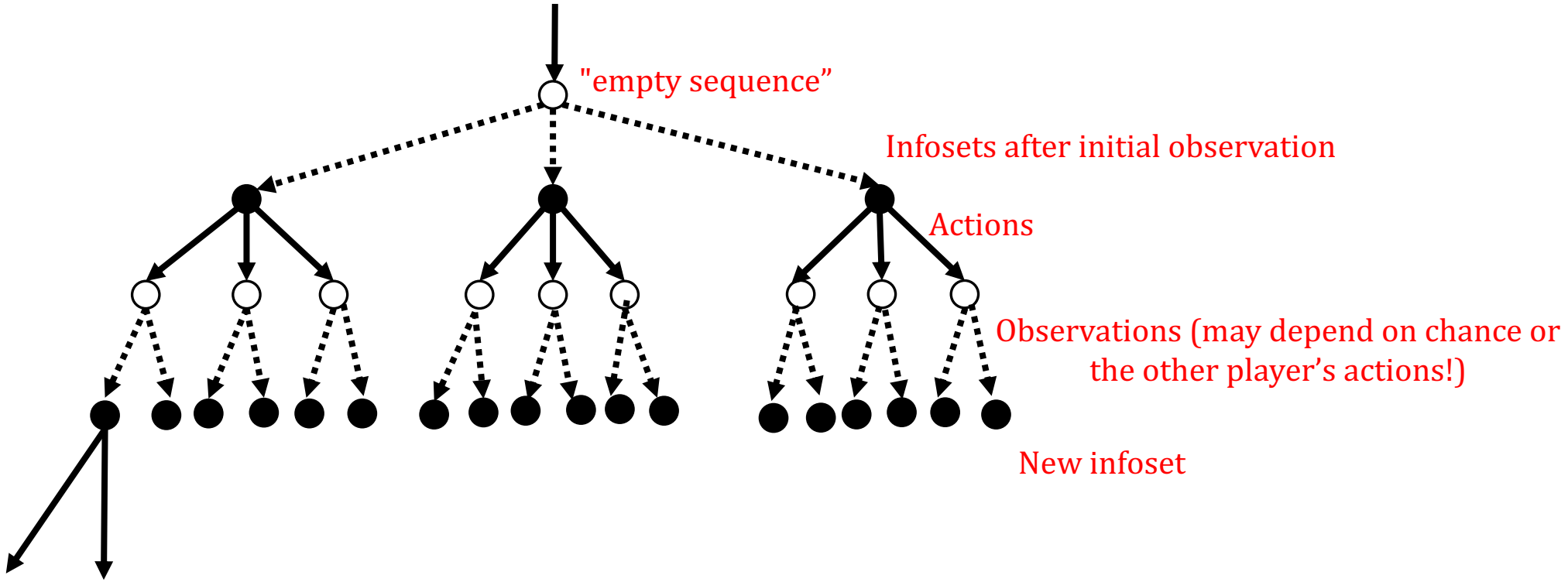
$g^{(t)} = A^T x^{(t)}$

We are going to do the exact same thing but $x^{(t)}, y^{(t)}$ are now in the treeplex

Average strategies converge to Nash (saddle point residual drops to 0)

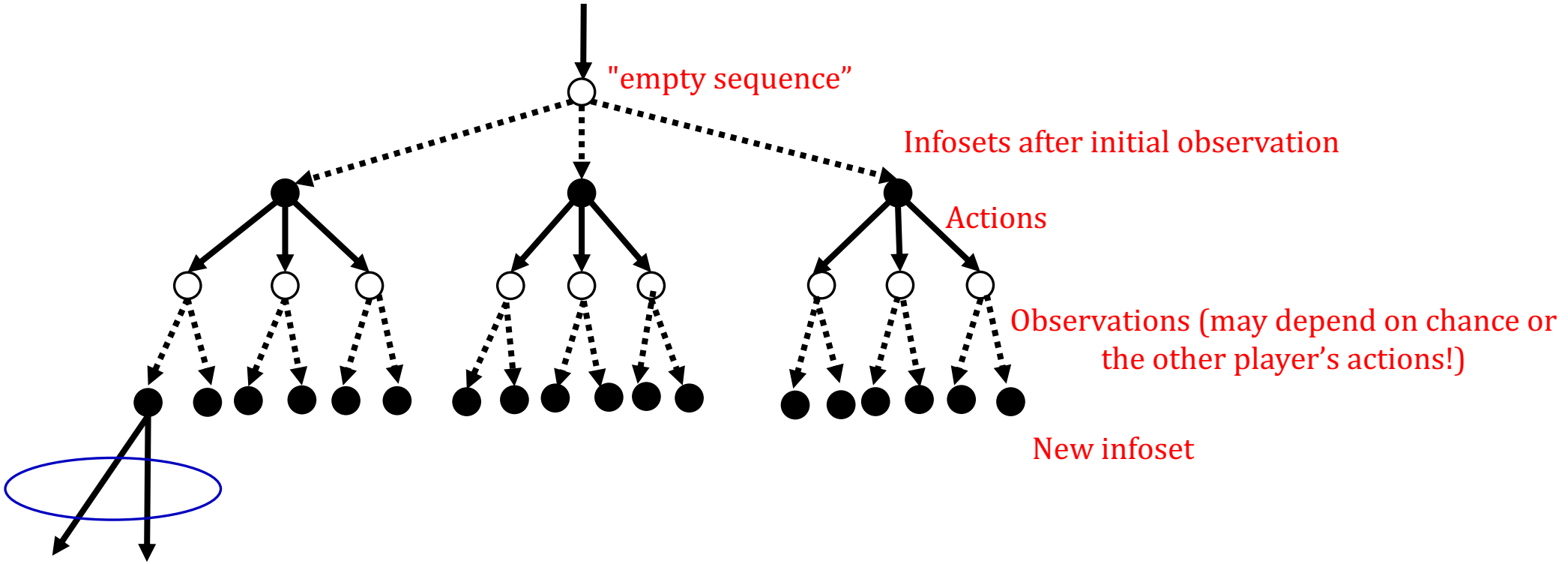
Immediate Counterfactual Regret (I)

Recall the treeplex structure



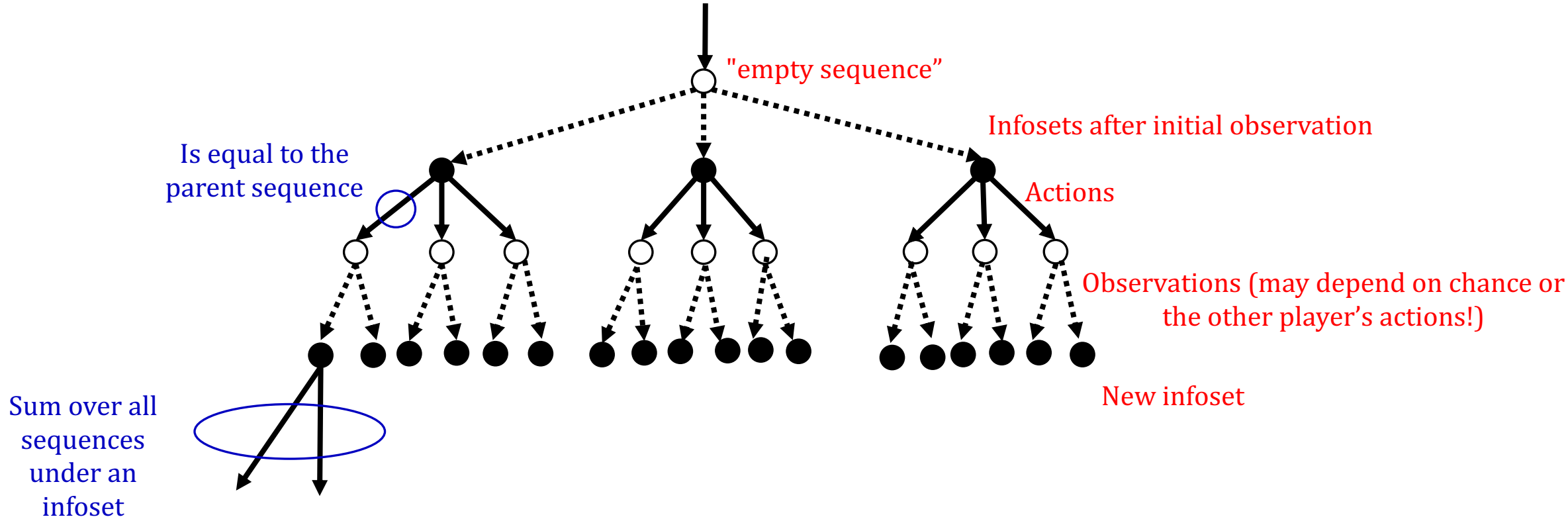
Immediate Counterfactual Regret (I)

Recall the treeplex structure



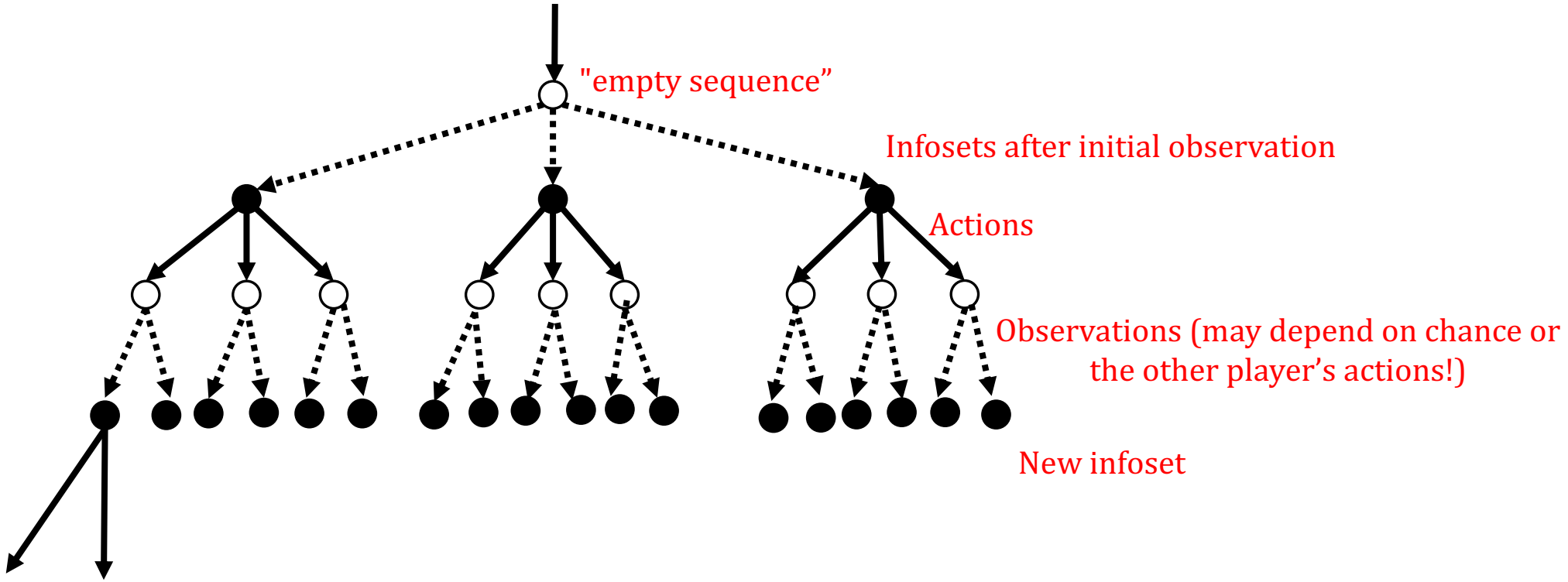
Immediate Counterfactual Regret (I)

Recall the treeplex structure



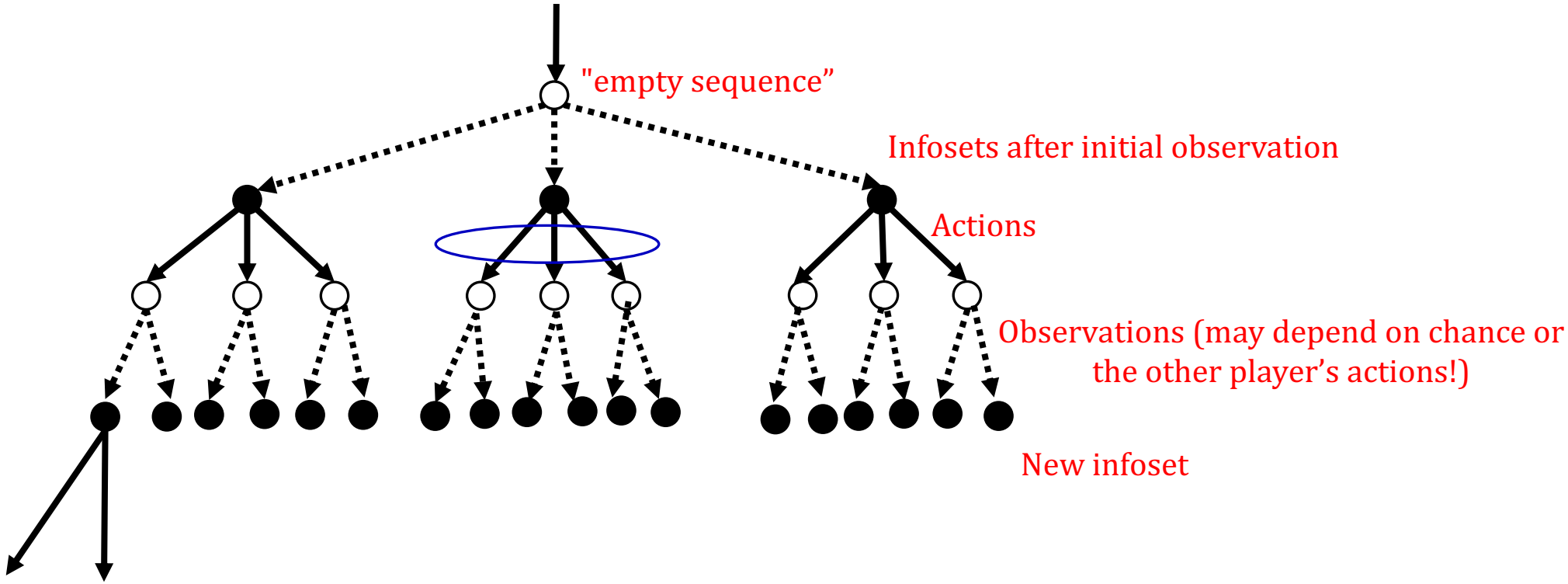
Immediate Counterfactual Regret (I)

Recall the treeplex structure



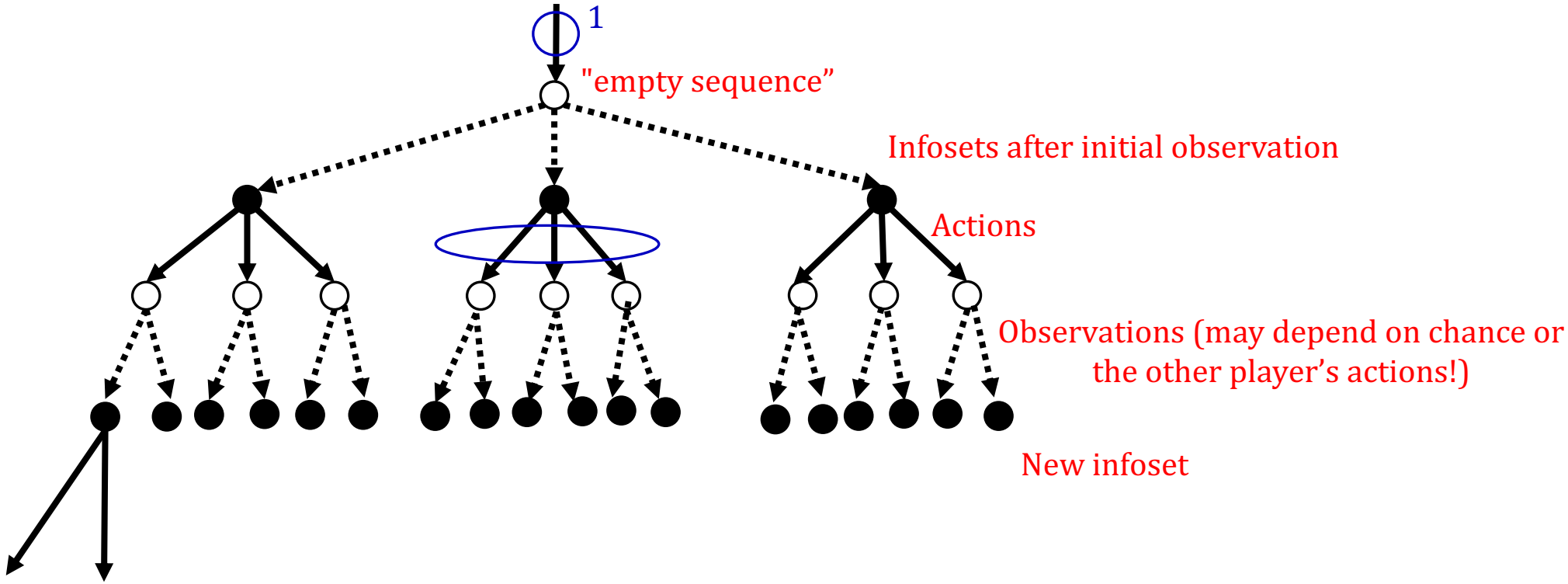
Immediate Counterfactual Regret (I)

Recall the treeplex structure



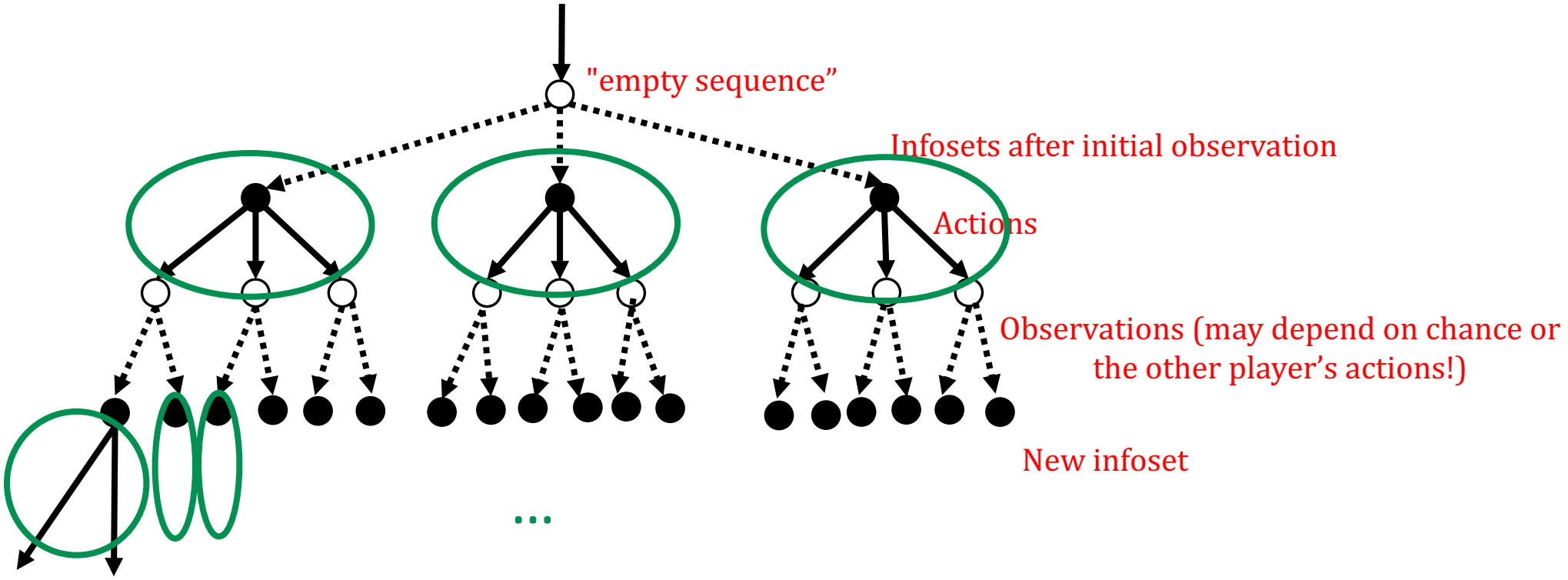
Immediate Counterfactual Regret (I)

Recall the treeplex structure



Immediate Counterfactual Regret (II)

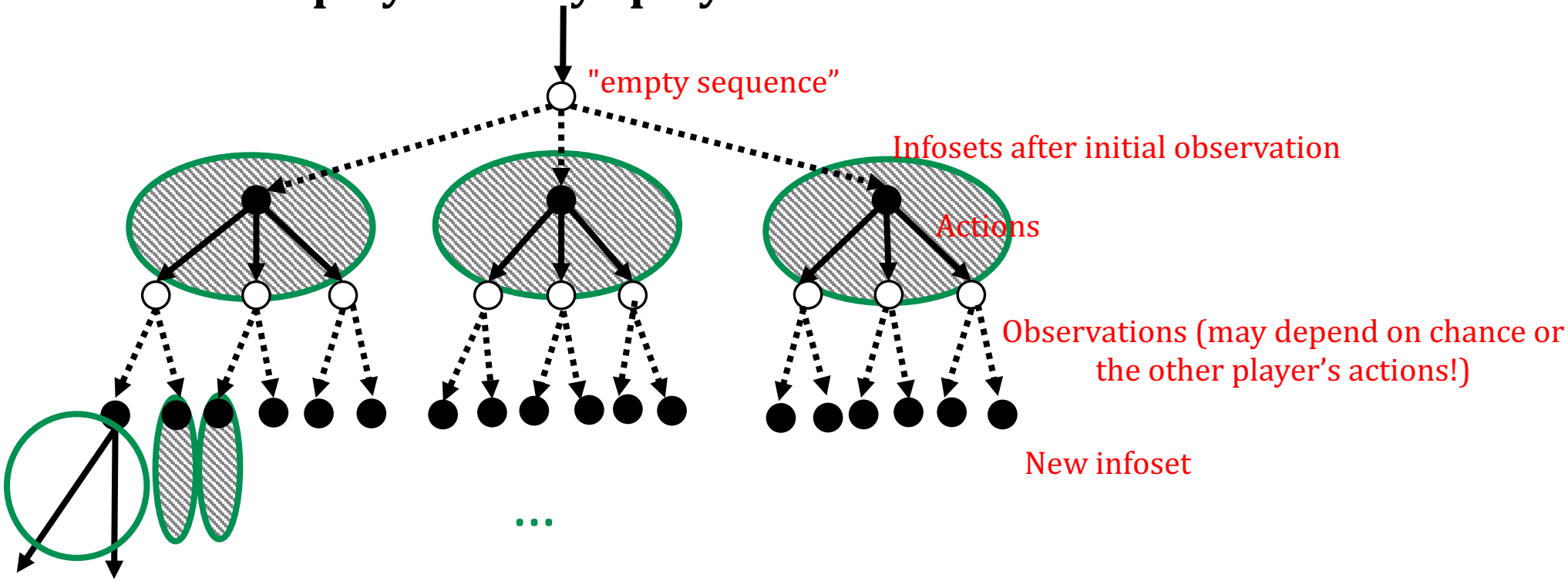
Place regret minimizers here



Immediate Counterfactual Regret (III)

Freeze behavioral strategy at every other decision point

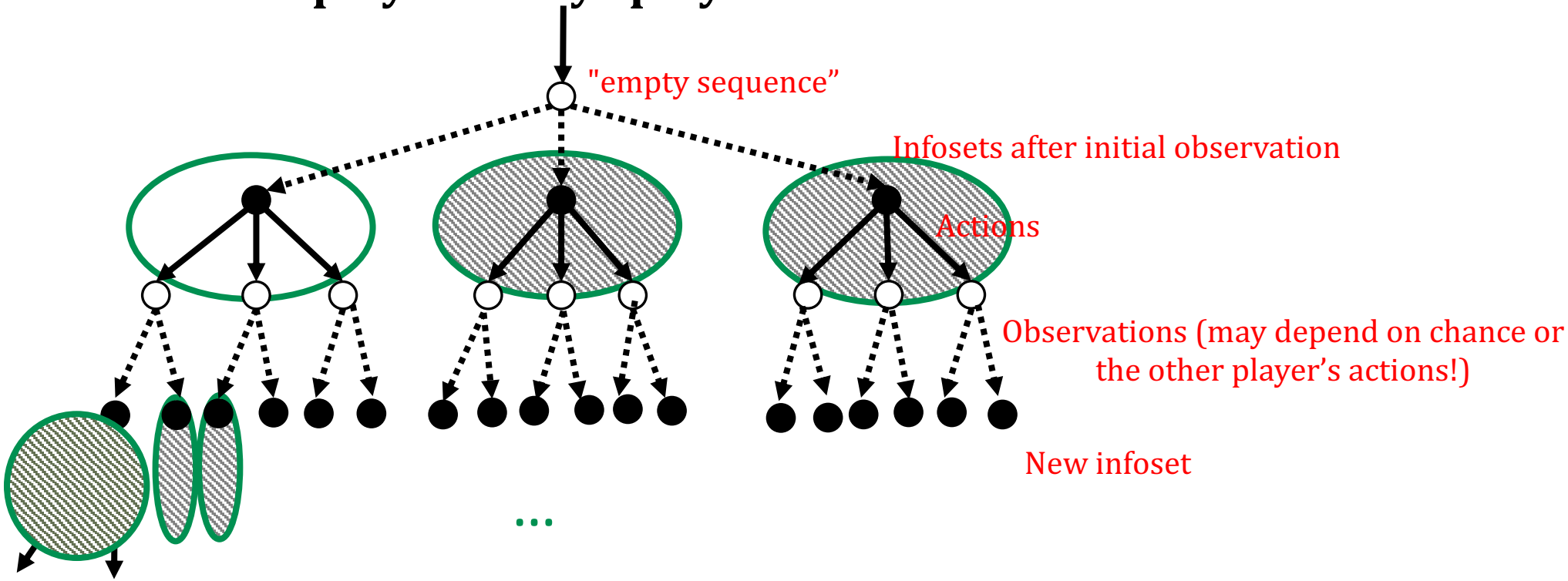
- Loss/Reward vector is what we would have obtained if all other decision makers played their recommended local strategy
- **Assume that the player always plays towards that infoset**



Immediate Counterfactual Regret (IV)

Freeze behavioral strategy at every other decision point

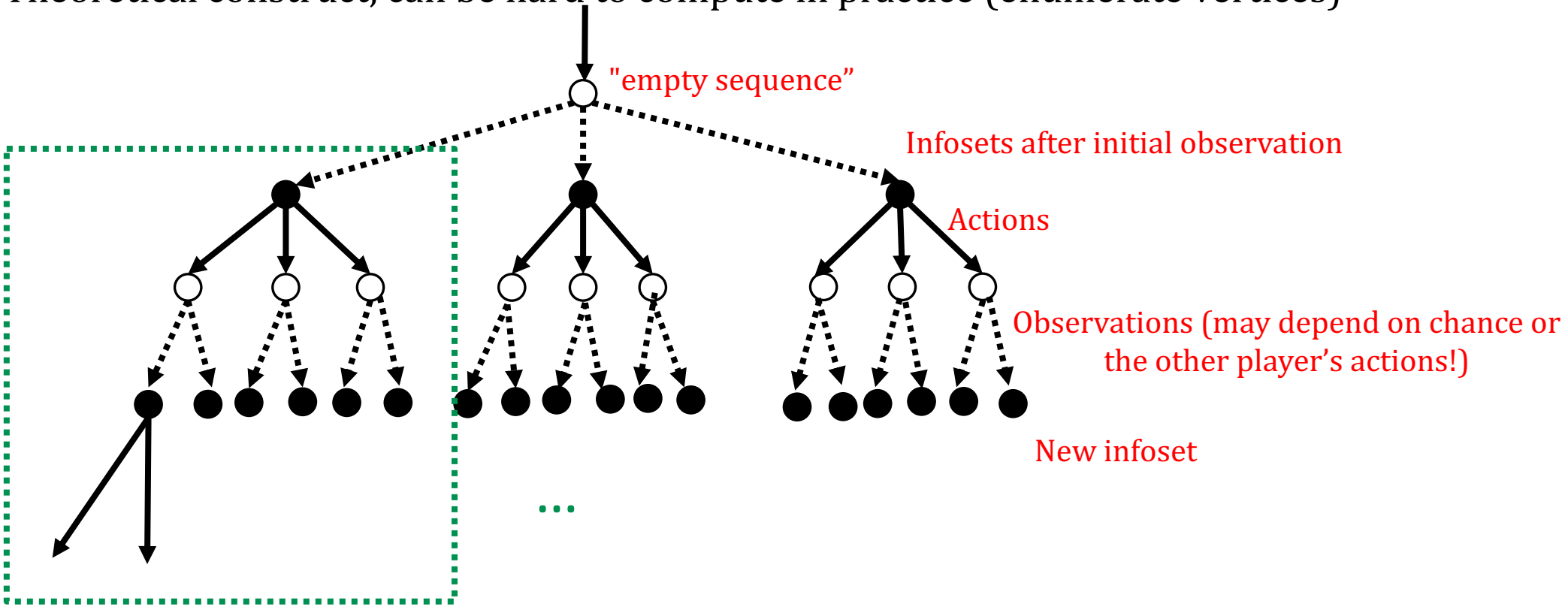
- Loss/Reward vector is what we would have obtained if all other decision makers played their recommended local strategy
- **Assume that the player always plays towards that infoset**



Full counterfactual regret

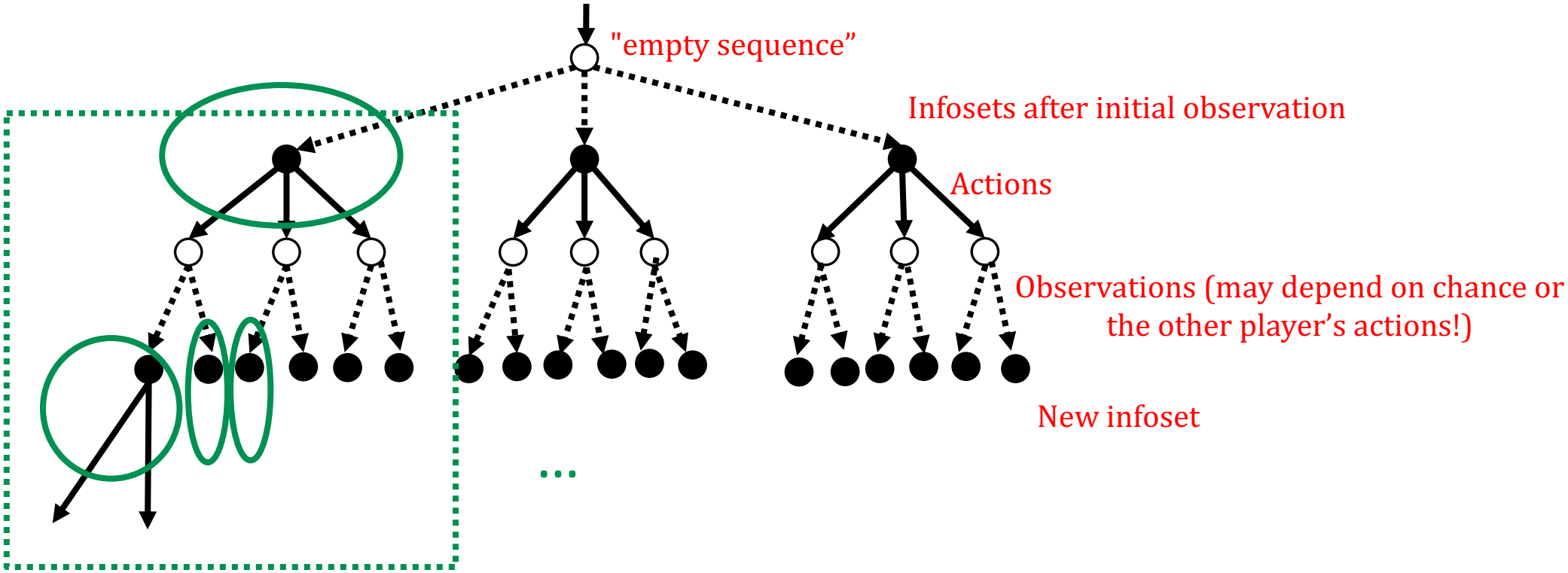
Take any infoset and consider the sub-treeplex **rooted there**

- Full counterfactual regret
  - What we got versus the case if we could change any actions from that infoset **and its descendants**
  - Theoretical construct, can be hard to compute in practice (enumerate vertices)



Sum of immediate regrets bounds full regret

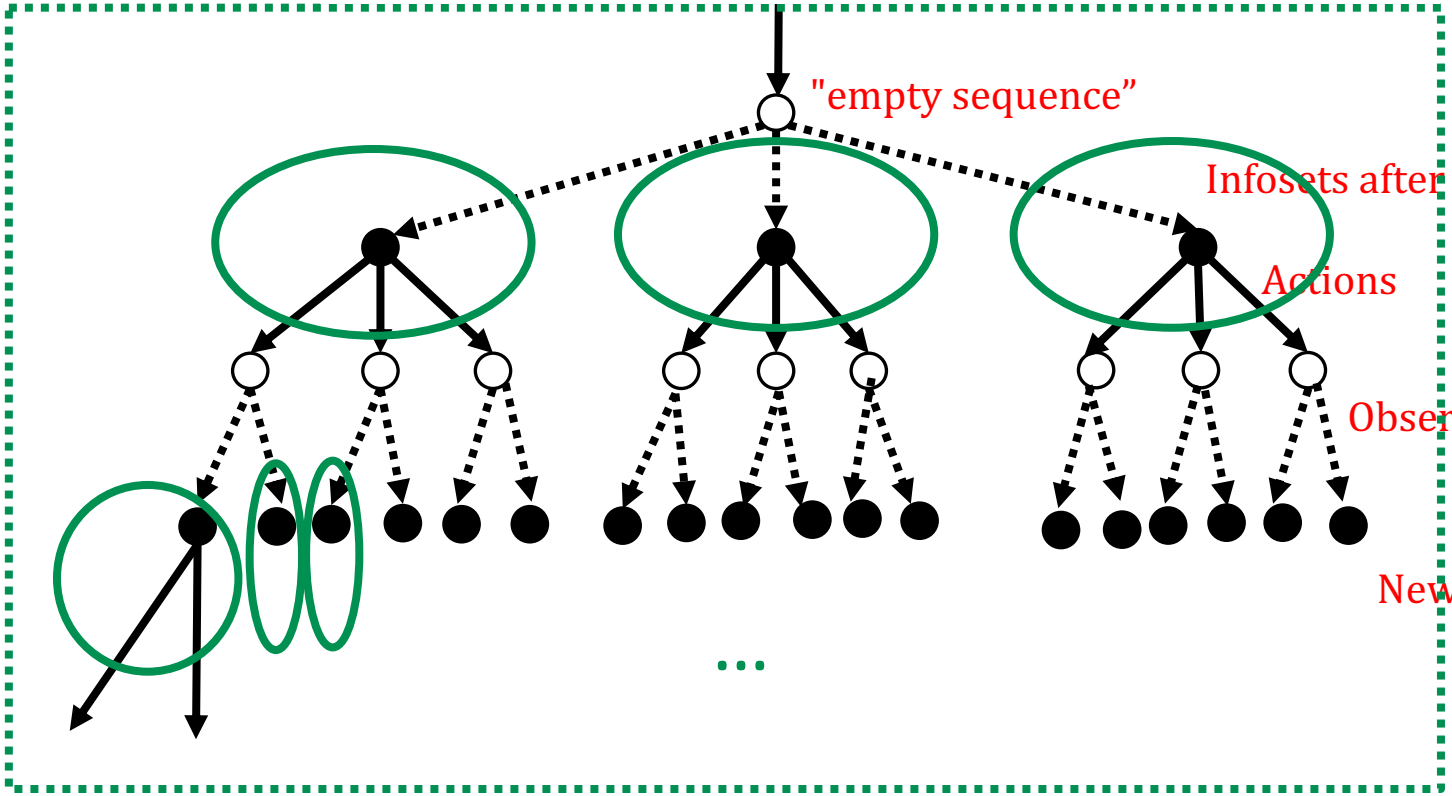
Lemma 7 $R^{T,+}_{i,\text{full}}(I) \leq \sum_{I' \in D(I)} R^{T,+}_{i,\text{imm}}(I').$



Full Regret( ) ≤ ∑Immediate Regret( )

At the root…

Theorem 3 $R_i^T \leq \sum_{I \in \mathcal{I}_i} R_{i,\text{imm}}^{T,+}(I)$



Full Regret( ) ≤ ∑Immediate Regret( )

Minimize immediate counterfactual regrets

Lemma 7 $R_{i,\text{full}}^{T,+}(I) \leq \underbrace{\sum_{I' \in D(I)} R_{i,\text{imm}}^{T,+}(I')}.$

Sublinear in T

Minimize immediate counterfactual regrets

Lemma 7 $R^{T,+}_{i,\text{full}}(I) \leq \sum_{I' \in D(I)} R^{T,+}_{i,\text{imm}}(I').$

Sublinear in T Sublinear in T

Minimize immediate counterfactual regrets

Lemma 7 $R^{T,+}_{i,\text{full}}(I) \leq \sum_{I' \in D(I)} R^{T,+}_{i,\text{imm}}(I').$

Sublinear in T Sublinear in T

Use any regret minimizer out there!
Most commonly RM+

Tying it up with Nash

$$\bar{\sigma}_i^t(I)(a) = \frac{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(I)(a)}{\sum_{t=1}^{T} \pi_i^{\sigma^t}(I)}. \quad (4)$$

There is a well-known connection between regret and the Nash equilibrium solution concept.

Theorem 2 *In a zero-sum game at time T, if both player's average overall regret is less than ϵ, then $\bar{\sigma}^T$ is a 2ϵ equilibrium.*

https://poker.cs.ualberta.ca/publications/NIPS07-cfr.pdf

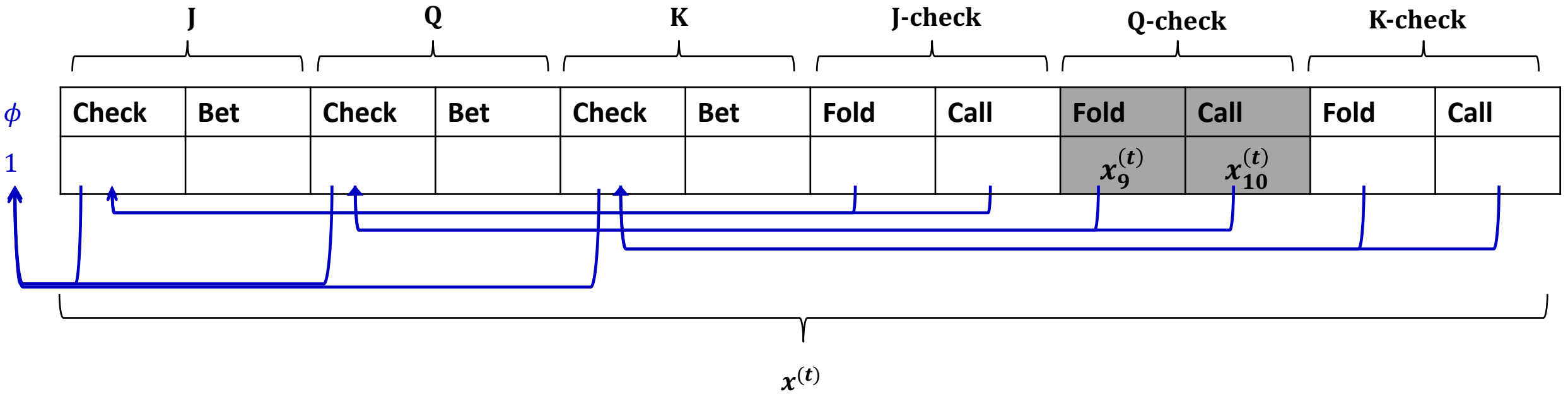
Very important: we are averaging over sequence form strategies, NOT behavioral strategies

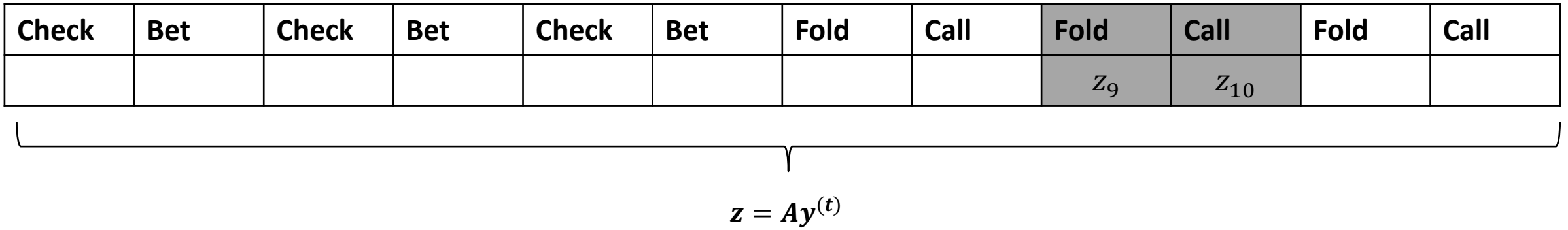
- Sometimes the latter would "work" approximately, but in general no

*Another perspective: regret circuits

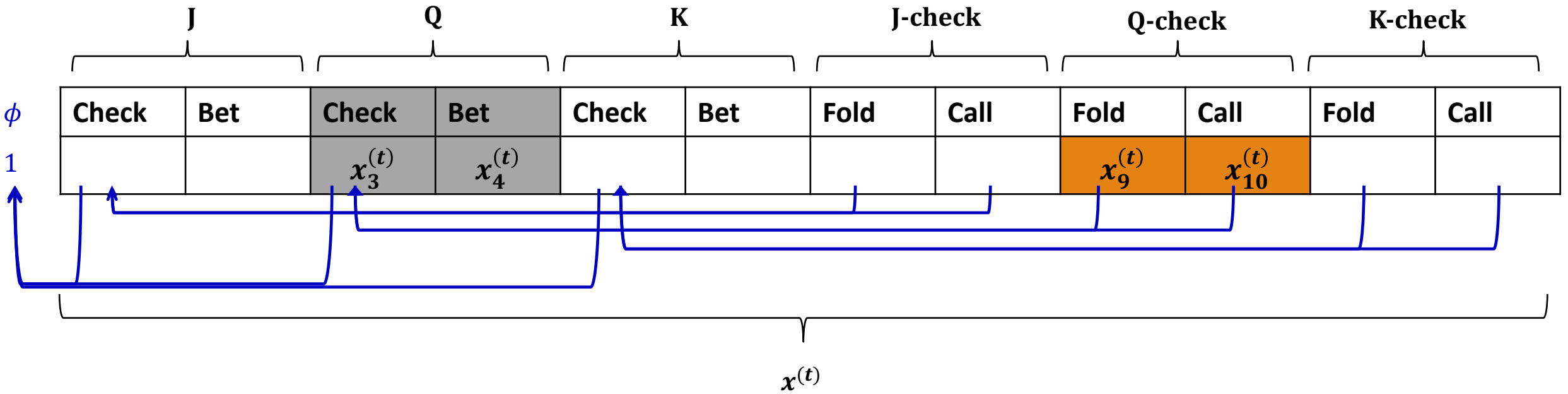
Some implementation issues…

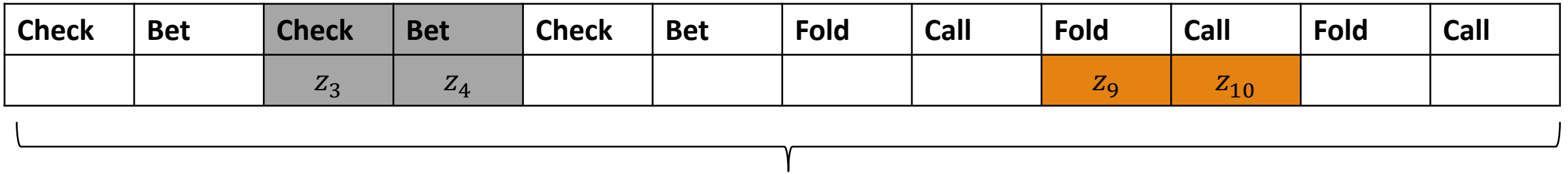
What is the $g_I^{(t)}$ that an infoset I sees?





What is the $g_I^{(t)}$ that an infoset I sees?

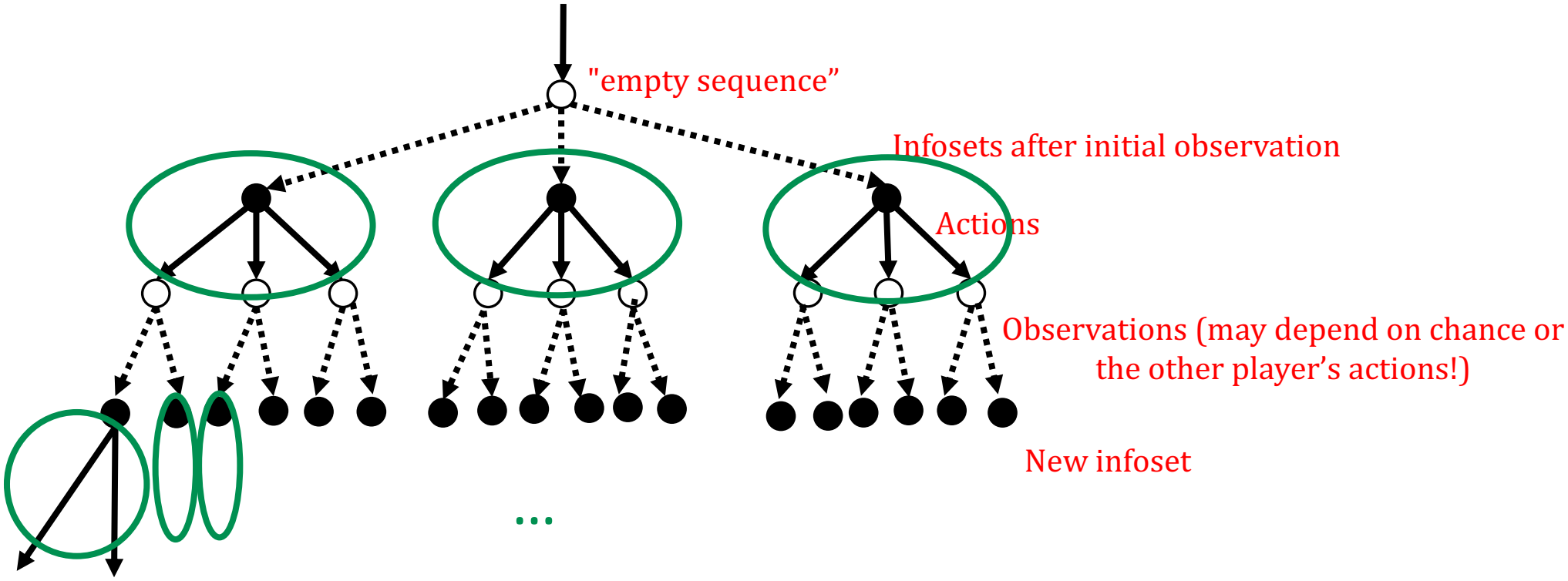




*Have to add the contribution from children infoset!!!

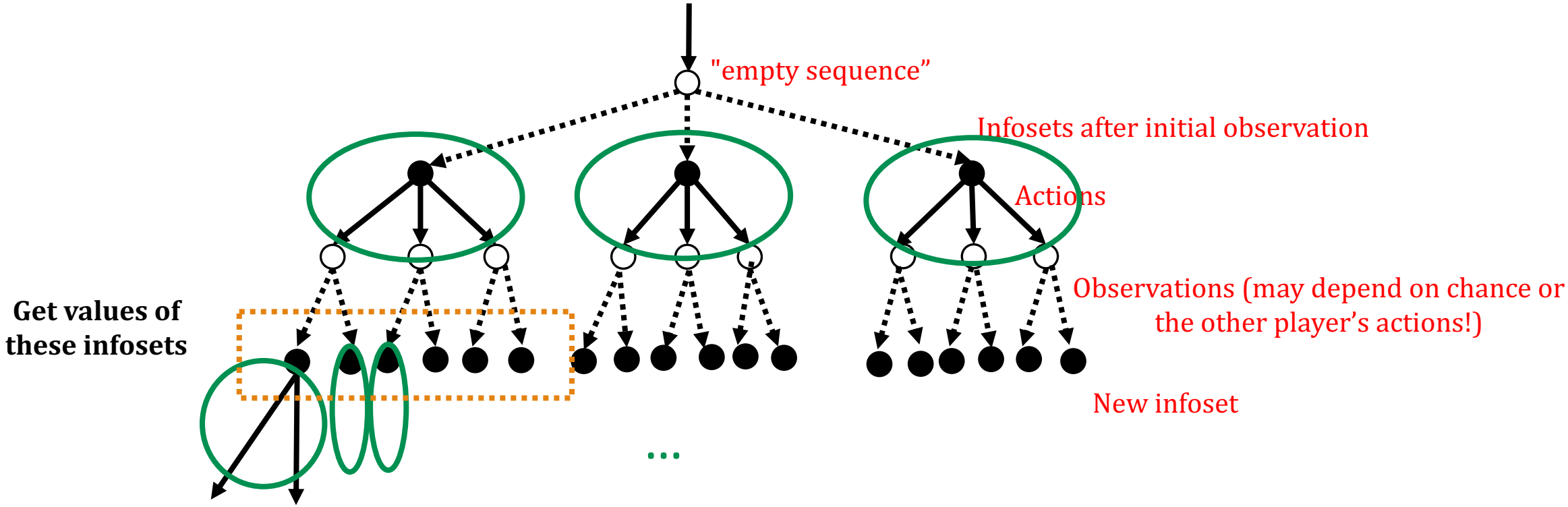
How do we do this efficiently?

Work in behavioral strategies and propagate loss (or reward) vector $g^{(t)}$ upwards the tree. Compute value of each infoset v_I



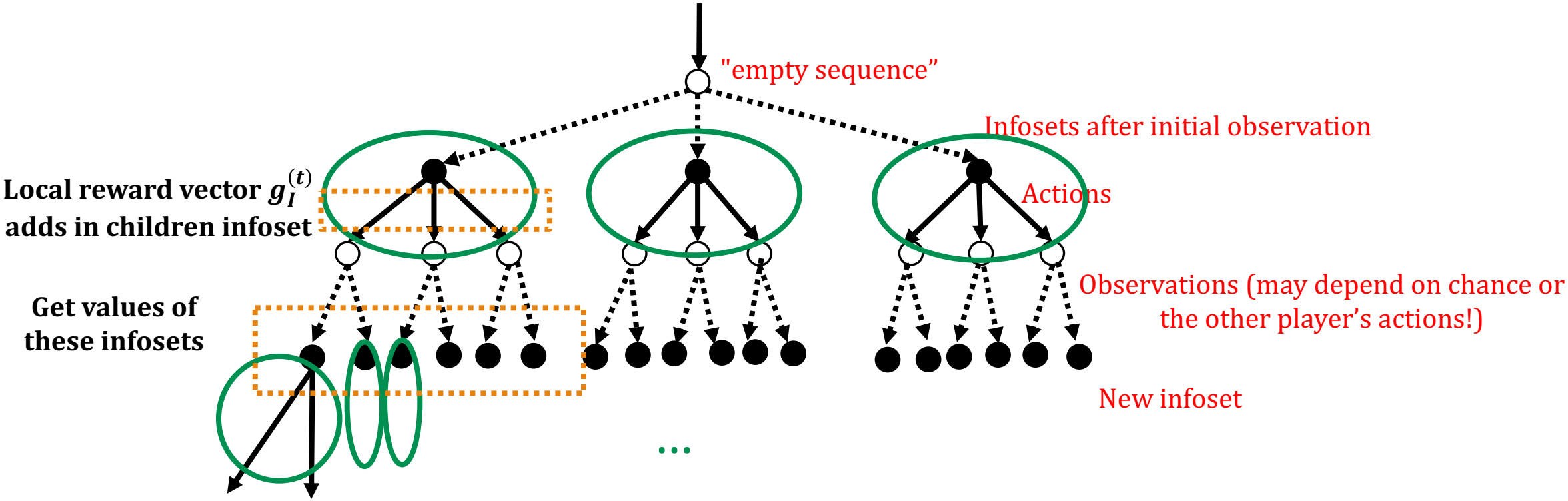
How do we do this efficiently?

Work in behavioral strategies and propagate loss (or reward) vector $g^{(t)}$ upwards the tree. Compute value of each infoset v_I



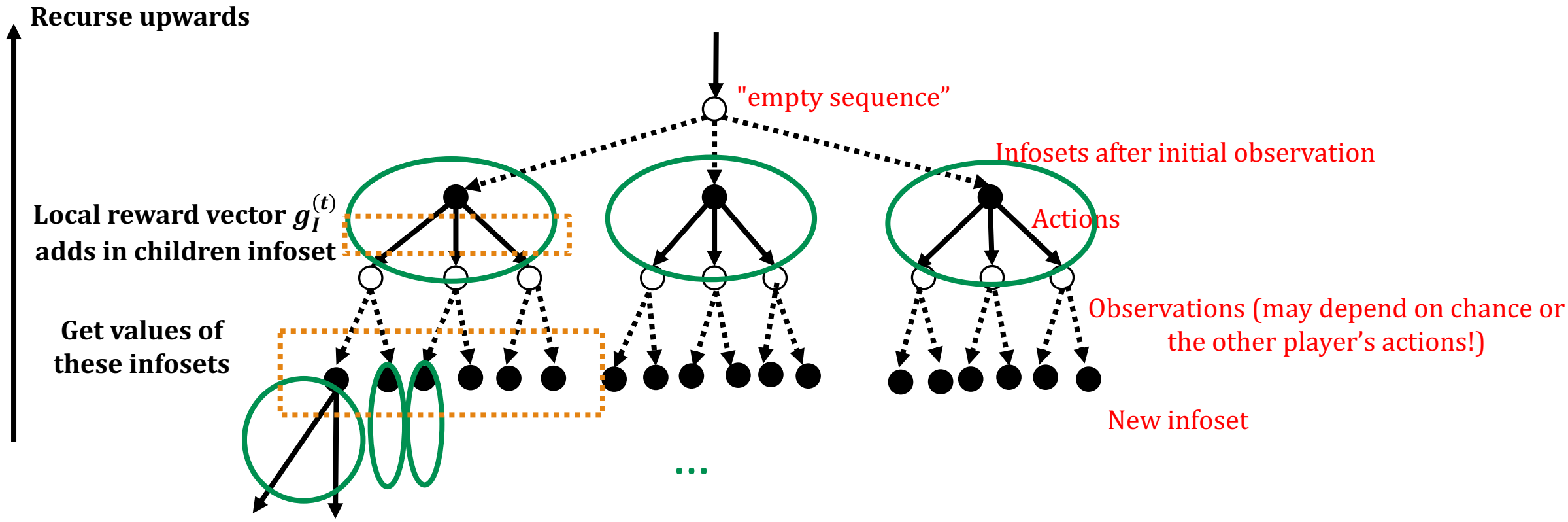
How do we do this efficiently?

Work in behavioral strategies and propagate loss (or reward) vector $g^{(t)}$ upwards the tree. Compute value of each infoset v_I



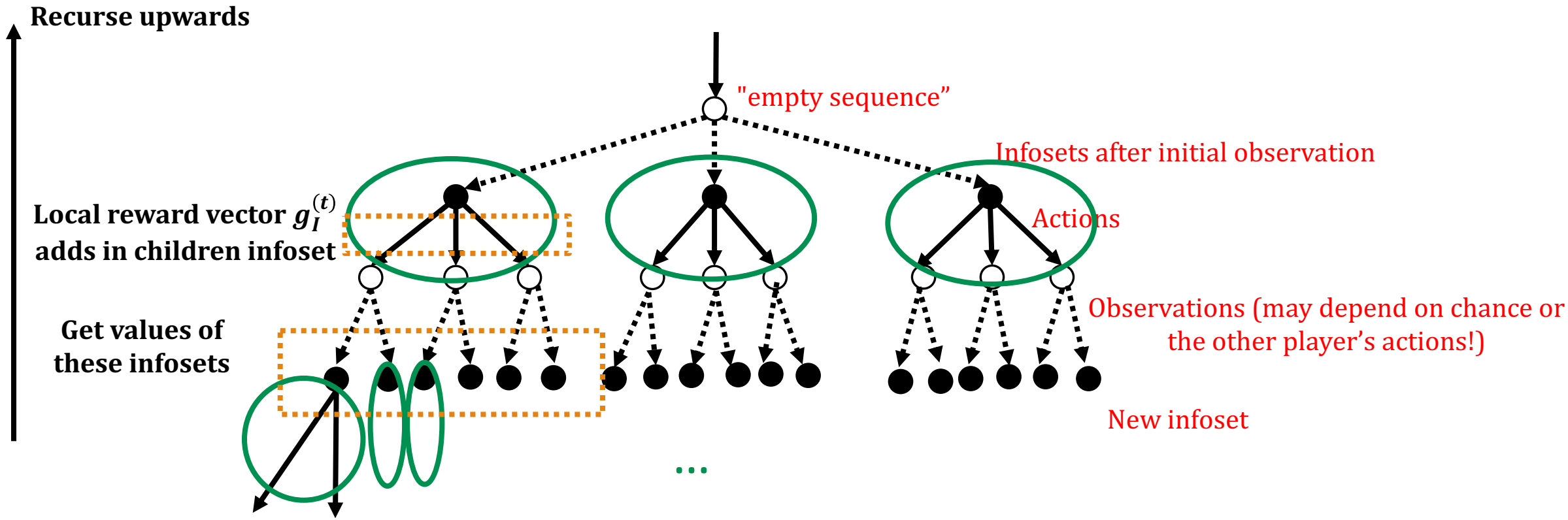
How do we do this efficiently?

Work in behavioral strategies and propagate loss (or reward) vector $g^{(t)}$ upwards the tree. Compute value of each infoset v_I



How do we do this efficiently?

Work in behavioral strategies and propagate loss (or reward) vector $g^{(t)}$ upwards the tree. Compute value of each infoset v_I



This gives us a vector $\hat{g}^{(t)}$ that considers the "weighted cumsum" of all a sequence's descendants

Summary of CFR

For each player

- For the regret minimizer in each infoset
  - Get next strategy, concat to form behavioral strategy
- Convert to sequence form (store for averaging later)
- Sent $A^T x^{(t)}, Ay^{(t)}$ to other player

Summary of CFR

For each player

- For the regret minimizer in each infoset
  - Get next strategy, concat to form behavioral strategy
- Convert to sequence form (store for averaging later)
- Sent $A^T x^{(t)}, A y^{(t)}$ to other player

For each player

- Get $g^{(t)}$, compute $\hat{g}^{(t)}$
- For each infoset, observe relevant cells of $\hat{g}^{(t)}$
  - Observe rewards/losses in the regret minimizer for that infest

Summary of CFR

For each player

- For the regret minimizer in each infoset
  - Get next strategy, concat to form behavioral strategy
- Convert to sequence form (store for averaging later)
- Sent $A^T x^{(t)}, Ay^{(t)}$ to other player

For each player

- Get $g^{(t)}$, compute $\hat{g}^{(t)}$
- For each infoset, observe relevant cells of $\hat{g}^{(t)}$
  - Observe rewards/losses in the regret minimizer for that infest

Average **sequence form** strategies converge to Nash

The end!

Reminders

- HW1
- Quiz 1
- Project topic proposal