

# Modern Algorithmic Game Theory

Martin Schmid

Department of Applied Mathematics  
Charles University in Prague

November 26, 2025



# Sequence Form Representation

# Sequence Form

---

- The conversion to normal-form games allows us to solve extensive-form games using the techniques we already know, e.g. linear programming in two-player zero-sum games
- Unfortunately, as we have seen, the constructed normal-form game can be exponentially large, making it unusable beyond smaller games
- One downside of pure strategies is that they require specifying an action for **every** information state of a player, even the ones that become unreachable given the player's previous moves
- Instead of representing all possible plans of actions, we can represent all paths in the tree, called **sequences**

# Sequences & Perfect Recall

---

- A sequence of moves of player  $i$  is the sequence of its actions (ignoring actions of other players) on the unique path from the root to the history  $h$ , denoted by  $\sigma_i(h)$
- Consider the history  $\{K, J, \text{check}, \text{bet}, \text{call}\}$  in Kuhn Poker:
  - $\sigma_1(K, J, \text{check}, \text{bet}, \text{call}) = (\text{check}, \text{call})$
  - $\sigma_2(K, J, \text{check}, \text{bet}, \text{call}) = (\text{bet})$
- Using the notion of sequences, we can now state a formal definition of **perfect recall**

## Definition: Perfect Recall

A game satisfies **perfect recall** if and only if, for all players  $i \in \mathcal{N}$  and for all pairs of histories  $h_1, h_2 \in \mathcal{S}_i$ , it holds that  $\sigma_i(h_1) = \sigma_i(h_2)$

# Sequences

---

- As we only consider games with perfect recall, we can denote the unique sequence of moves leading to an information state  $s \in \mathcal{S}_i$  as  $\sigma_s$
- Formally, we can define the set  $\Sigma_i$  of all sequences of player  $i$  as  
$$\Sigma_i = \{\emptyset\} \cup \{\sigma_s a \mid s \in \mathcal{S}_i, a \in \mathcal{A}_i(s)\}$$
- We can see that the size of  $\Sigma_i$  is the total number of unique moves player  $i$  can perform (plus 1 for the empty sequence), i.e.  $1 + \sum_{s \in \mathcal{S}_i} |\mathcal{A}_i(s)|$ , which is **linear in the size of the game tree!**

# Normal-Form vs. Sequence-Form Representation

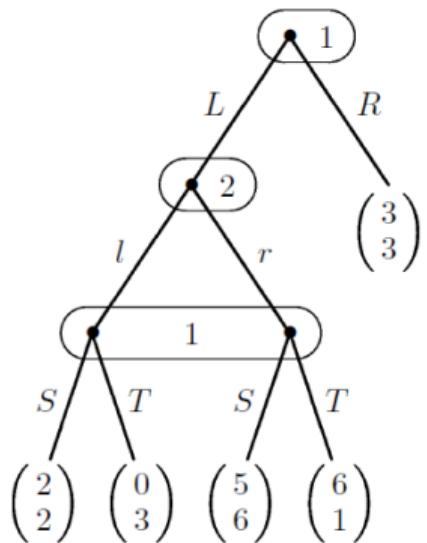

$$A = \begin{array}{cc|c} & & l \quad r \\ & & \hline 2 & 5 & \langle L, S \rangle \\ 0 & 6 & \langle L, T \rangle \\ 3 & 3 & \langle R, S \rangle \\ 3 & 3 & \langle R, T \rangle \end{array}$$
$$B = \begin{array}{cc|c} & & l \quad r \\ & & \hline 2 & 6 & \langle L, S \rangle \\ 3 & 1 & \langle L, T \rangle \\ 3 & 3 & \langle R, S \rangle \\ 3 & 3 & \langle R, T \rangle \end{array}$$
$$A = \begin{array}{ccc|c} & & l \quad r \\ & \emptyset & \hline 3 & & \emptyset \\ & & L \\ & 2 & R \\ & 5 & LS \\ 0 & 6 & LT \end{array}$$
$$B = \begin{array}{ccc|c} & & l \quad r \\ & \emptyset & \hline \emptyset & & \emptyset \\ & & L \\ 3 & & R \\ & 2 & LS \\ & 6 & LT \\ 3 & 1 & LT \end{array}$$

Figure: Up: Normal-form representation; Down: Sequence-form representation

# Realization Probabilities & Expected Utilities

---

- Given a behavioral strategy  $\pi_i$ , we define the **realization probability** of a sequence  $\sigma_s$  as  $\pi_i[\sigma_s] = \prod_{a \in \sigma_s} \pi_i(a)$ , where  $\pi_i(a)$  is the behavioral strategy at a corresponding information state along the sequence
- Given behavioral strategies for all players (including the chance player), we can express the expected utility of player  $i$  in terms of realization probabilities as

$$u_i(\pi) = \sum_{h \in \mathcal{Z}} u_i(h) \pi_c[\sigma_c(h)] \pi_1[\sigma_1(h)] \pi_2[\sigma_2(h)]$$

- However, such an expression is non-linear in individual behavioral probabilities  $\pi_i(a)$  which is unsuitable for a linear program
- Instead, we will consider the realization probabilities as functions of sequences  $\sigma_s$  directly

# Realization Probabilities & Plans

---

- Let us denote the realization probability of a sequence  $\sigma_s$  as  $x(\sigma_s)$
- Then, a vector  $x \in \mathbb{R}^{|\Sigma_i|}$  consisting of realization probabilities of all sequences  $\sigma_s \in \Sigma_i$  is called a **realization plan**
- To ensure that each  $x(\sigma_s)$  behaves as the product of behavioral strategies corresponding to  $\sigma_s$ , we will require each realization plan to satisfy the following linear constraints

$$x(\emptyset) = 1$$
$$\sum_{a \in \mathcal{A}_i(s)} x(\sigma_s a) = x(\sigma_s) \quad \forall s \in \mathcal{S}_i$$

- Since we will solve the LP using realization plans, we need to ensure that realization plans and behavioral strategies are equivalent representations

# From Realization Plans to Behavioral Strategies

---

- A realization plan  $x(\sigma_s)$  uniquely defines the behavioral strategy  $\pi_i$  of player  $i$
- For each information set  $s \in \mathcal{S}_i$  and action  $a \in \mathcal{A}_i(s)$ , we define

$$\pi_i(s, a) = \frac{x(\sigma_s a)}{x(\sigma_s)}$$

- This ratio is valid because realization plans satisfy the consistency condition

$$x(\sigma_s) = \sum_{a \in \mathcal{A}_i(s)} x(\sigma_s a)$$

- If  $x(\sigma_s) = 0$ , i.e. the information set is unreachable, the behavioral strategy at  $s$  can be chosen arbitrarily because it does not influence any expected utilities

# From Behavioral Strategies to Realization Plans

---

- Any behavioral strategy  $\pi_i$  induces a realization plan  $x(\sigma_s)$
- For a sequence  $\sigma_s = (a_1, a_2, \dots, a_k)$  of player  $i$ 's actions, taken at information sets  $s_1, s_2, \dots, s_k$ , the realization probability is

$$x(\sigma_s) = \prod_{j=1}^k \pi_i(s_j, a_j)$$

- This construction ensures that realization probabilities behave exactly as the product of the behavioral choices along the path
- Thus, behavioral strategies and realization plans are equivalent, but realization plans are linear and therefore suitable for LP formulations

# Sequence-Form Payoff Matrix & Expected Utilities

---

- For an extensive-form game with perfect recall and a fixed strategy  $\pi_c$  of the chance player, we define the entries of the **sequence-form payoff matrix** corresponding to a pair of sequences  $(\sigma, \tau)$  as

$$A_{\sigma, \tau} = \sum_{h \in \mathcal{Z}: \sigma_1(h) = \sigma, \sigma_2(h) = \tau} u_i(h) \pi_c[\sigma_c(h)]$$

- This matrix is **sparse**; it contains zeros whenever a pair of sequences  $(\sigma, \tau)$  does not lead to a terminal history
- To compute the expected payoff given the realization plans for both players, we just need to go through all the terminal nodes and weight them accordingly

$$\sum_{h \in \mathcal{Z}} u_i(h) \pi_c[\sigma_c(h)] x_1(\sigma_1(h)) x_2(\sigma_2(h))$$

- Or equivalently, we can express it in the familiar form  $x^\top A y$

# Realization Plan Constraints in Matrix Form

---

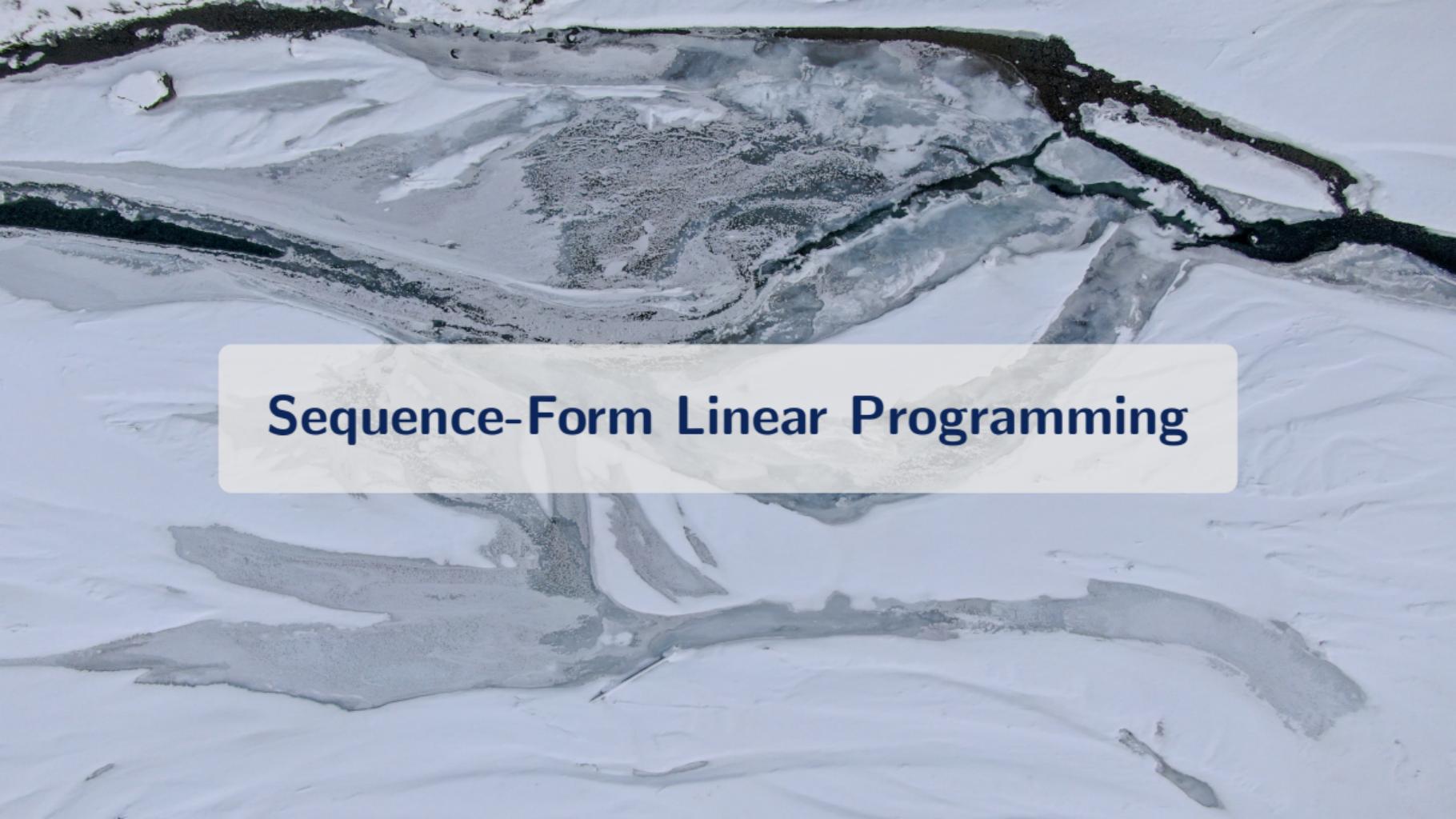
- Consider the realization plan constraints again

$$x(\emptyset) = 1 \text{ and } \sum_{a \in \mathcal{A}_i(h)} x(\sigma_h a) = x(\sigma_h) \quad \forall h \in \mathcal{H}_i$$

- We can express these in a matrix form as  $Ex = e$ , where each row in  $E$  represents the left-hand side of the constraint for the corresponding sequence of Player 1
- Similarly, we can use  $Fy = f$  for Player 2
- Player 1's realization plan constraints from the example game are

$$\begin{bmatrix} 1 \\ -1 & 1 & 1 \\ & -1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

- The sequence-form payoff matrices, together with the realization plan constraints define the **sequence form** of an extensive-form game



# Sequence-Form Linear Programming

# Sequence-Form LP Derivation

---

- Now that we have defined the sequence-form of an extensive-form game and the realization plans, we are ready to derive the **Sequence-form LP**
- It is now easy to see that for a fixed realization plan  $y$  of our opponent, we can compute a best response using the following linear program

$$\max_x x^\top A y$$

$$Ex = e$$

$$x \geq 0$$

# Sequence-Form LP Derivation

---

- The dual LP to the LP from the previous slide looks as follows

$$\begin{aligned} \min_u & u^\top e \\ & E^\top u \geq Ay \end{aligned}$$

- Both LPs have feasible solutions and by the strong duality theorem, their optimal values coincide
- We can intuitively view the dual LP as a *certificate* of how large the best response value can be as it upper-bounds  $x^\top Ay$

# Sequence-Form LP Derivation

---

- We know that a Minimax strategy of Player 2 is the solution to the following expression

$$\min_y \max_x x^\top A y$$

- However, by the Strong Duality Theorem  $\max_x x^\top A y = \min_u u^\top e$  subject to  $E^\top u \geq A y$  for a fixed  $y$
- Substituting this to the above expression, we get

$$\begin{aligned} & \min_{y,u} u^\top e \\ & E^\top u \geq A y \end{aligned}$$

- Even after adding  $y$  as a variable, the above LP is still **linear** in both  $y$  and  $u$

# Sequence-Form LP

---

- Finally, we add the realization plan constraints for  $y$  and arrive at the LP on the left-hand side which finds a Minimax strategy for Player 2
- Dualizing this LP leads to the LP on the right-hand side for finding a Maximin strategy for Player 1 and looks as follows

$$\min_{y,u} u^\top e$$

$$Fy = f$$

$$E^\top u \geq Ay$$

$$y \geq 0$$

$$\max_{x,v} v^\top f$$

$$Ex = e$$

$$F^\top v \leq A^\top x$$

$$x \geq 0$$

- As we have previously proven, a pair of Maximin strategies corresponds to a Nash equilibrium in zero-sum games!

# Sequence-Form LP

---

- We have derived an LP formulation that finds a Nash equilibrium of a zero-sum extensive-form game by reformulating the game in terms of its sequence form
- This means that there is a polynomial time algorithm for finding Nash equilibria for two-player zero-sum extensive-form games
- This formulation leads to **exponentially smaller** representation compared to the induced normal-form representation
- It is possible to follow a very similar line of reasoning even for general-sum games
- However, the resulting problem is no longer a linear program but instead a **linear complementarity problem**
- The perhaps most well-known method for solving linear complementarity problems is **Lemke's algorithm**, which has exponential running time in the worst case

# Week 8 Homework

---

You can find more detailed descriptions of homework tasks in the GitHub repository.

1. Extensive-form to normal-form games conversion
2. Finding Nash equilibria in zero-sum games using sequence-form linear programming