Dresden University of Technology

Computational Logic ∴ Group

Hannes Strass

Faculty of Computer Science, Institute of Artificial Intelligence, Computational Logic Group

Sequential Games with Perfect Information

Lecture 4, 4th May 2026 // Algorithmic Game Theory, SS 2026

Previously . . .

- Approximate Nash Equilibria allow instability up to a certain threshold ε.
- Computing an ε-**Nash** equilibrium of a normal-form game is **PPAD-complete**.
- A **correlated equilibrium** can be seen as providing players with private signals they can use to best-respond to each other's strategies.
- Computing a **correlated** equilibrium of a normal-form game is **in FP**.

Penalties

Two football players face off at a (simplified) single penalty kick. The kicker can kick left or right; the goal keeper can jump left or right. The kicker scores a goal iff they choose a different side than the keeper.

(`Kicker`, `Keeper`)	`JumpL`	`JumpR`
`KickL`	(–1, 1)	(1, –1)
`KickR`	(1, –1)	(–1, 1)

Overview

Motivation

Sequential Games with Perfect Information
- Game Trees
- Example: Media Streaming
- Backward Induction

Combinatorial Games

Motivation

Motivation

Strategic Games in Normal Form:

- Only one decision is made by the players (choice of strategy)
- Strategies are seen as black boxes, no fine-grained analysis possible
- Simultaneous decisions: players do not know others' contribution to the final outcome

But many "real" games (Chess, Checkers, Go) are not like that. Instead:

- Several decisions are being made throughout the game
- Strategies can be seen as advice on which move to pick, for every possible situation
- Players move sequentially (typically taking turns), know each other's previous contributions to the final outcome

In both models, there is complete information about the game:
All players know possible strategies as well as utilities of all players.

Sequential Games with Perfect Information

Sequential Games with Perfect Information

Definition

A **sequential game with perfect information** consists of the following:

1. A set $P = \{1, \ldots, n\}$ of players.
2. An n-tuple $\mathbf{M} = (M_1, \ldots, M_n)$ of sets M_i of **moves** for each player i.
3. A set H of **histories**, sequences $[m_1, \ldots, m_k]$ of moves $m_j \in M_1 \cup \ldots \cup M_n$.
4. A subset $Z \subseteq H$ of **terminal** histories.
5. A **player function** $p\colon H \setminus Z \to P$ (indicating whose turn it is).
6. An n-tuple $\mathbf{u} = (u_1, \ldots, u_n)$ of utility functions $u_i\colon Z \to \mathbb{R}$.

Starting with the **empty history** $[]$, in each history $h = [m_1, \ldots, m_k] \in H \setminus Z$, player $i = p(h)$ chooses a move $m \in M_i$, leading to $[h; m] := [m_1, \ldots, m_k, m]$.

- Perfect information: All players know all previous moves of all players.
- Move m is **possible** in history $h \in H$ iff $[h; m] \in H$.

Background: Directed Trees

Definition

1. A **directed graph** is a pair $G = (V, E)$ with V a set of nodes and $E \subseteq V \times V$ a set of edges (ordered pairs of nodes).
2. For $u, v \in V$, a **path** from u to v is a sequence $\langle v_0, v_1, \ldots, v_m \rangle$ of nodes that are all distinct and where $u = v_0$, $v_m = v$, and $(v_{i-1}, v_i) \in E$ for $1 \leq i \leq m$.
3. A (directed) **tree** is a directed graph $G = (V, E)$ with one distinguished node $r \in V$, the **root**, such that for every node $v \in V$ there is a unique path from r to v.
4. For an edge $(u, v) \in E$ we say that u is the **parent** of v, resp. that v is the **child** of u. A node $v \in V$ without children is a **leaf**.

Observation

Every node in a tree is either the root or it has a unique parent.

Representation via Game Trees

Definition

A sequential game with perfect information is represented as a **game tree**:

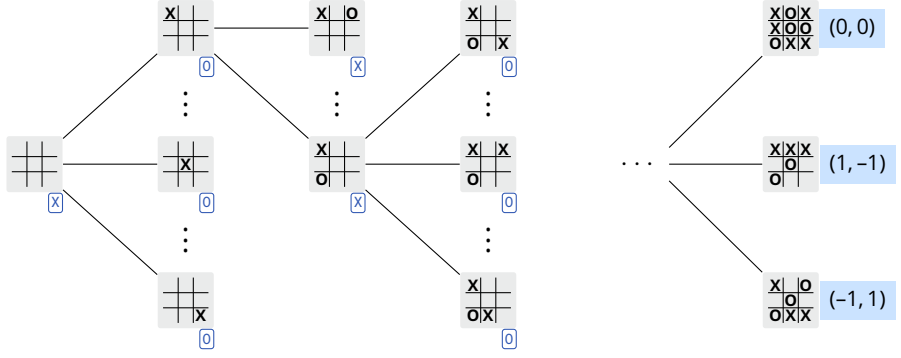
1. Each node of the game tree corresponds to a history $h \in H$, where
   - the root corresponds to the empty history $[]$, and
   - leaves (and only leaves) correspond to terminal histories $z \in Z$.
2. Each non-leaf node (**decision node**) h is labelled by $p(h)$.
3. The children of a node h are given by the possible moves of $p(h)$.
4. Each leaf node z is labelled by its utility value $\mathbf{u}(z) = (u_1(z), \ldots, u_n(z))$.

- Possible (i.e. legal) moves are encoded implicitly in the tree.
- Other forms of modelling legality are possible (e.g. another function).
- Representation of (sequential) games by trees is sometimes referred to as **extensive form** representation (as opposed to the normal form).

Example: Tic-Tac-Toe

Tic-Tac-Toe

Two players, X and O, take turns marking initially empty fields of a 3×3-table with their symbols X and O, respectively. Whoever manages to mark a full row, column, or diagonal with their symbol wins. X moves first.



Example: Media Streaming (Story)

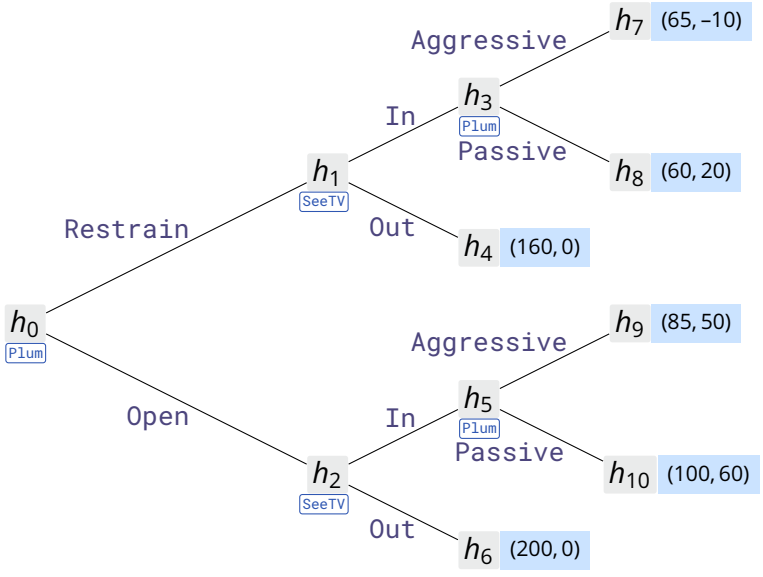
Media Streaming

(Gillman & Housman, 2019)

Tech company `Plum` plans the marketing campaign for their new device. `Aggressive` marketing is more expensive (and successful) than `Passive` marketing. Competitor `SeeTV` is working on a clone which will need some more time to market. `SeeTV` must decide whether to continue this work to be `In` on the market (or discontinue and be `Out`). `SeeTV` can clone the device better and more cheaply if they recruit some of `Plum`'s engineers to work for `SeeTV`. In anticipation, `Plum` can put `Restrain`ing clauses into their engineers' contracts (barring them from working for competitors if they leave). However, such clauses require `Plum` to pay higher salaries than for `Open` contracts.

The market is expected to support sales sufficient to generate 200m in net revenue. Restraining clauses cost additional 40m in salaries; aggressive marketing costs additional 25m and yields a 65% market share with restraining clauses and a 55% share without; passive marketing yields a 50% share. It costs `SeeTV` 40m to develop the clone with `Plum` engineers, and 80m without.

Example: Media Streaming (Game Tree)



Strategies in Game Trees

A strategy for a player in a game tree must assign to each decision node of that player exactly one (legal/possible) move.

Example

- In media streaming, player `Plum` has the decision nodes $h_0 = []$, $h_3 = [\texttt{Restrain}, \texttt{In}]$, and $h_5 = [\texttt{Open}, \texttt{In}]$ (with two moves each).
- Player `SeeTV` has the decision nodes $h_1 = [\texttt{Restrain}]$ and $h_2 = [\texttt{Open}]$ (with two possible moves each).
- Thus player `Plum` has eight different strategies, among them $\{h_0 \mapsto \texttt{Open}, h_3 \mapsto \texttt{Passive}, h_5 \mapsto \texttt{Aggressive}\}$ (“use open contracts; do passive [aggressive] marketing for restrained [open] contracts”).
- Player `SeeTV` has four different strategies, among them $\{h_1 \mapsto \texttt{In}, h_2 \mapsto \texttt{In}\}$ written as (`In`, `In`) (“enter the market in any case”) and (`Out`, `In`) (“enter the market if (and only if) `Plum` uses open contracts”).

From Game Trees to Strategic Games

Definition

Let G be a finite sequential game with perfect information and P its players.

The **associated normal-form game** $G' = (P, \mathbf{S}, \mathbf{u}')$ is as follows:

- Denote by $p^{-1}(i) := \{h \in H \setminus Z \mid p(h) = i\}$ the set of histories where it is i's turn.
- $\mathbf{S} = (S_1, \ldots, S_n)$ with each S_i the set of all functions

$$s_i \colon p^{-1}(i) \to M_i \quad \text{with} \quad [h; s_i(h)] \in H \text{ for all } h \in p^{-1}(i)$$

- $\mathbf{u}' = (u'_1, \ldots, u'_n)$ with each u'_i mapping the profile $\mathbf{s} = (s_1, \ldots, s_n)$ to $u_i(\langle\!\langle \mathbf{s}, [] \rangle\!\rangle)$, where the function $\langle\!\langle \mathbf{s}, \cdot \rangle\!\rangle \colon H \to Z$ is defined by structural induction via

$$\langle\!\langle \mathbf{s}, h \rangle\!\rangle := \begin{cases} h & \text{if } h \in Z, \\ \langle\!\langle \mathbf{s}, [h; s_i(h)] \rangle\!\rangle \text{ for } i = p(h) & \text{otherwise.} \end{cases}$$

Strategies s_i can be reduced by removing all move assignments to histories the strategy cannot reach.

Solving Media Streaming (1)

Using the possible strategies of `Plum` and `SeeTV`, we can formulate media streaming as strategic game in normal form:

(`Plum`, `SeeTV`)	(`In`, `In`)	(`In`, `Out`)	(`Out`, `In`)	(`Out`, `Out`)
(`Restrain`, `Aggressive`, `Aggressive`)	(65, –10)	(65, –10)	(160, 0)	(160, 0)
(`Restrain`, `Aggressive`, `Passive`)	(65, –10)	(65, –10)	(160, 0)	(160, 0)
(`Restrain`, `Passive`, `Aggressive`)	(60, 20)	(60, 20)	(160, 0)	(160, 0)
(`Restrain`, `Passive`, `Passive`)	(60, 20)	(60, 20)	(160, 0)	(160, 0)
(`Open`, `Aggressive`, `Aggressive`)	(85, 50)	(200, 0)	(85, 50)	(200, 0)
(`Open`, `Aggressive`, `Passive`)	(100, 60)	(200, 0)	(100, 60)	(200, 0)
(`Open`, `Passive`, `Aggressive`)	(85, 50)	(200, 0)	(85, 50)	(200, 0)
(`Open`, `Passive`, `Passive`)	(100, 60)	(200, 0)	(100, 60)	(200, 0)

Solving Media Streaming (2)

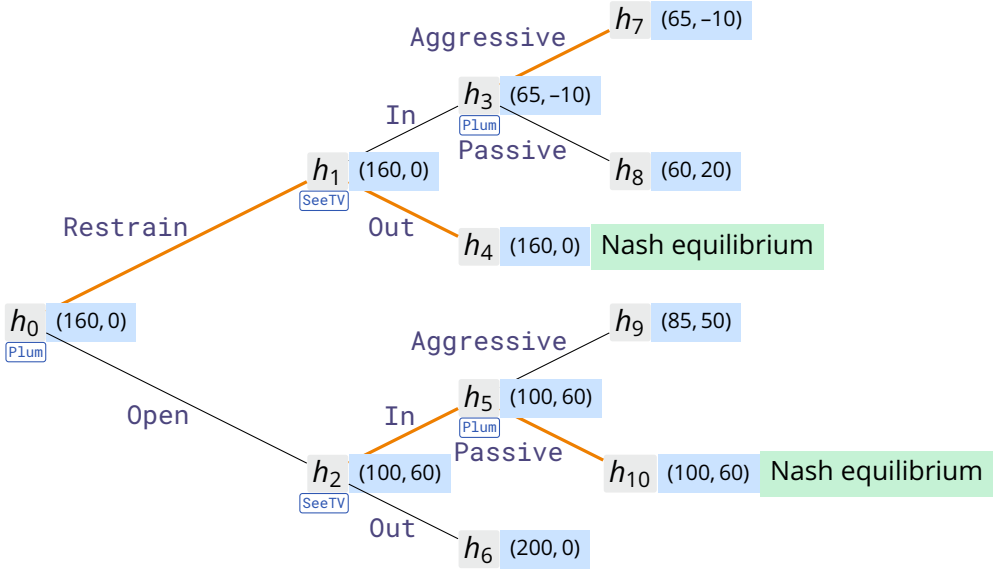
Using the possible strategies of `Plum` and `SeeTV`, we can formulate media streaming as reduced strategic game in normal form:

(`Plum`, `SeeTV`)	(`In`, `In`)	(`In`, `Out`)	(`Out`, `In`)	(`Out`, `Out`)
(`Restrain`, `Aggressive`, ∗)	(65, –10)	(65, –10)	(160, 0)	(160, 0)
(`Restrain`, `Passive`, ∗)	(60, 20)	(60, 20)	(160, 0)	(160, 0)
(`Open`, ∗, `Aggressive`)	(85, 50)	(200, 0)	(85, 50)	(200, 0)
(`Open`, ∗, `Passive`)	(100, 60)	(200, 0)	(100, 60)	(200, 0)

There are two Nash equilibria in pure strategies (and inessential further equilibria in mixed strategies).

Is this the “right” way to model and solve this game?

Solving Media Streaming (3)



Backward Induction

The approach used for solving the example can be generalised:

```
function backward-induction() { label(u, {} , []) }          // start recursion at root

function label(u, s, h) {
    if h ∈ Z then return                                     // terminal nodes are labelled by u
    i := p(h)                                                // obtain player to move
    v* = (v*_1, ..., v*_n) := (−∞, ..., −∞)                  // initialise current best (for i) payoff
    for each m ∈ M_i with [h; m] ∈ H {                        // for each possible move
        label(u, s, [h; m])                                  // recursively label child
        if u_i([h; m]) > v*_i then {                         // better move found
            m* := m                                          // update current best move
            v* := u([h; m]) }                                // update current best value
    }
    u(h) := v*                                               // set utility value of this history
    s_i(h) := m* }                                           // set best move for this history
```

Zermelo's Theorem

Theorem (Zermelo, 1913)

Every sequential game tree with a finite number of nodes has a backward induction solution, that is, a strategy profile and (utility) outcome.

Proof.

By induction on the length n of the game's longest terminal history:

- $n = 1$: Pick the action with the highest payoff.
- $n \rightsquigarrow n + 1$:
  - Consider a game G with longest terminal history of length $n + 1$.
  - Label all parents h of terminal nodes with the payoff vector of the terminal node of their best move m_h^*.
  - Define game G' by taking these h to be terminal (removing their children).
  - Game G' has a longest terminal history of length n.
  - Thus by induction hypothesis, G' can be solved by backward induction.
  - To obtain the strategy profile for G, append the moves m_h^* to that of G'. □

Subgames

Definition

Let G be a sequential perfect-information game and h be a history therein.
The **subgame** $G(h)$ of G beginning at h is as follows:

1. Players P and moves $\mathbf{M}$ of $G(h)$ are as in G.
2. Histories of $G(h)$ are $H_h := \{h' \mid [h; h'] \in H\}$
3. Terminal histories of $G(h)$ are $Z_h := \{h' \mid [h; h'] \in Z\}$
4. Player function and utilities of $G(h)$ are those of G.

Example

- Tic-Tac-Toe has a subgame starting at  where X can force a win.
- In media streaming's subgame at h_2, SeeTV can guarantee payoff 60.
- Every sequential perfect-information game G is a subgame of itself.

Subgame Perfect Equilibria

Definition

Let G be a sequential perfect-information game with players $P = \{1, \dots, n\}$.
A strategy profile $\mathbf{s} = (s_1, \dots, s_n)$ is a **subgame perfect equilibrium** iff for every non-terminal history $h \in H$, for every player $i \in P$ and for every $s_i' \in S_i$:

$$u_i(\langle\langle \mathbf{s}, h \rangle\rangle) \geq u_i\big(\langle\langle (s_1, \dots, s_{i-1}, s_i', s_{i+1}, \dots, s_n), h \rangle\rangle\big)$$

where $\langle\langle \mathbf{t}, h \rangle\rangle$ again denotes the terminal history of playing according to $\mathbf{t}$ in $G(h)$.

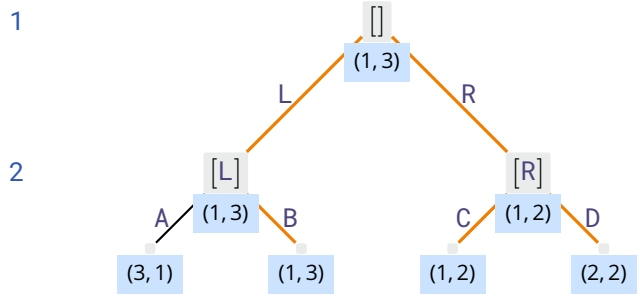
Theorem

In any sequential perfect-information game, subgame perfect equilibria coincide with the strategy profiles obtained by backward induction.

Proof Idea.

In each step of backward induction, we choose a best response. □

Subgame Perfect Equilibria: Example



- (L, BC), (R, BC), (R, BD) are all subgame-perfect equilibria of this game.
- (L, BD) is not subgame-perfect, as L is not a best response to D.

Combinatorial Games

Combinatorial Games

Definition

A **combinatorial game** is a sequential perfect-information game with two players $P = \{1, 2\}$ where in each terminal history $z \in Z$, either:

- player i gets $u_i(z) = 1$ and the other player $3 - i$ gets $u_{3-i}(z) = -1$; or
- both players get $u_1(z) = u_2(z) = 0$.

Since for all $z \in Z$ we have $u_1(z) + u_2(z) = 0$, those games are **zero-sum**.

Examples: Tic-Tac-Toe, Checkers, Chess, Go

Observation

For each combinatorial game, either:

- Exactly one player has a winning strategy, or
- both players can guarantee a draw.

Solving Games

Note

Backward induction can in principle be used to solve finite games.

In practice, the size of the game (number of positions/histories) is key:

- Tic-Tac-Toe is (trivially) strongly solved, i.e. the optimal moves for every history of the game are known.
- Connect-Four is strongly solved (the first player can force a win).
- Checkers has been weakly solved in 2007, i.e. it is known that from the initial position, perfect play by both sides leads to a draw ($\approx 5 \cdot 10^{20}$ pos.).
- Othello has recently been claimed to be solved (draw, $\approx 10^{28}$ positions).
- Chess remains unsolved (6-piece endgame is strongly solved).
- Go remains unsolved (it is weakly solved on a 7×7-board).

From the perspective of computational complexity, however, all of these games can be solved in constant time (due to their finite size).

$\rightsquigarrow$ Analyse arbitrary-size generalisations of games.

A Game on Maps

Geography

Two players take turns naming countries (from an English dictionary). The first country can start with an arbitrary letter, but each next country (where repetition is disallowed) must start with the last letter of the previous country.

Example

`Ann` and `Bob` play geography. `Ann` starts with `germany`, to which `Bob` responds with `yemen`. Then `Ann` says `norway` and the game is over with `Ann` winning.

Geography

For a fixed directed graph $G = (V, E)$ and vertex $s \in V$, two players, `exists` and `forall`, take turns in marking vertices in G. Player `exists` starts and has to mark a successor v_1 of s, and from then on each player has to mark a vertex v_{i+1} that is a successor of the node v_i marked by the opponent in the previous round. No vertex may be marked twice, and a player unable to move loses.

Winning Strategies in Geography

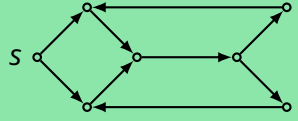
Geography

Given: A directed graph $G = (V, E)$ and a designated vertex $s \in V$.

Question: Is there a winning strategy for player `exists` in geography based on G with starting vertex s?

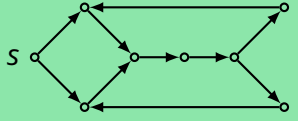
Examples

Consider $G_1 = (V_1, E_1)$:



`exists` has no winning strategy, thus $G_1 \notin$ **Geography**.

Consider $G_2 = (V_2, E_2)$:



`exists` has a winning strategy, thus $G_2 \in$ **Geography**.

Geography and Computational Complexity

Recall: Computational Complexity

- Complexity class PSpace comprises all decision problems (languages) that can be decided by polynomially space-bounded Turing machines.
- A decision problem is PSpace-hard iff every problem in PSpace can be reduced to it in polynomial time.
- A problem is PSpace-complete iff it is in PSpace and PSpace-hard.

Theorem

Geography is PSpace-complete.

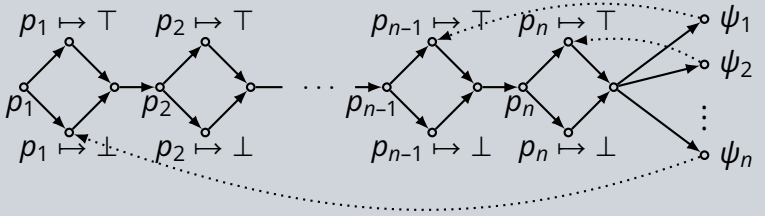
Proof (containment).

We use depth-first-search to traverse the game tree, where we only ever keep one path from the root to some decision node in memory, along with a list of children already considered for each node along that path. □

Geography is PSpace-complete

Proof (hardness).

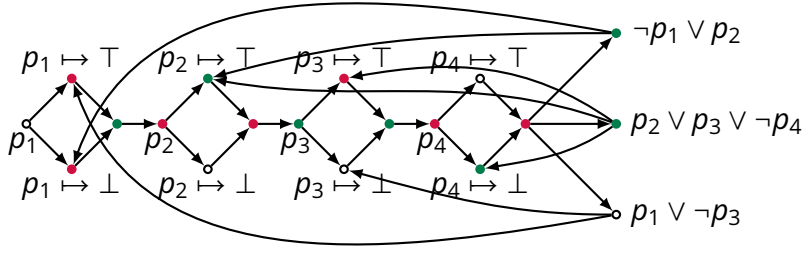
We reduce from the PSpace-hard problem **QBF-Truth**. Consider a quantified Boolean formula $\Phi = \exists p_1.\forall p_2.\cdots\exists p_{n-1}.\forall p_n.\Psi$ where $\Psi = \psi_1 \wedge \ldots \wedge \psi_m$ is a quantifier-free conjunction of clauses (disjunctions of literals). We create the following instance of **Geography**:



where for all $1 \leq j \leq m$ we add an edge from φ_j to $p_i = \top$ (resp. $p_i = \bot$) iff p_i occurs in ψ_j (resp. $\neg p_i$ occurs in ψ_j). Then `exists` has a winning strategy iff Φ is true: `exists` chooses $p_i \mapsto \top$ iff $p_i \mapsto \bot$ witnesses Φ being true. □

Hardness of Geography: Example

Consider QBF $\Psi = \exists p_1.\forall p_2.\exists p_3.\forall p_4.((\neg p_1 \vee p_2) \wedge (p_2 \vee p_3 \vee \neg p_4) \wedge (p_1 \vee \neg p_3))$. The construction yields the following **Geography** instance (start node p_1):



In this game of geography, `exists` has no winning strategy (`forall` has one), which is expected as Ψ is not true.

Conclusion

Summary

- **Game trees** are used to represent sequential (**extensive form**) games.
- Sequential games give rise to (different) strategic (normal form) games.
- In a game tree, a **strategy** assigns a move to each decision node.
- **Backward induction** can be used to solve sequential games.
- The **subgame perfect equilibria** of a sequential game coincide with its backward induction solution(s).
- Geography is a (variable-size) game on graphs for which deciding existence of winning strategies is PSpace-complete.

Action Points

- Implement backward induction and use it to solve Tic-Tac-Toe.