Strategy Complexity: How Much Does It Take to Win?

Pierre Vandenhove

UMONS – Université de Mons, Belgium

August 26, 2025 – Brussels Summer School of Mathematics

Overview

- Main topic: **Game theory**.
- **Game theory** is the study of mathematical models representing the **interaction** between multiple agents (called **players**), each pursuing an **objective**.
- Applications: economics, biology, social sciences, politics, **computer science**. . .
- Here, we will focus on what **strategies** look like; **how** to win? **What do the strategies look like?**
- **Disclaimer**: game theory is a vast field, with a plethora of models. I focus here on a particular model (*two-player turn-based games on graphs*) which is **well-studied** and still an **active research area**.

Table of contents

Table of contents

Nim game

We start with a simple special case of what we then consider: the **Nim game**.

- There are $n \geq 1$ **matchsticks**.
- Two players take turns removing **1, 2, or 3** matchsticks.
- The player who takes the **last** matchstick **loses**.

How to represent this game as a graph and **solve** it (i.e., find which player can enforce a win)?

$\rightsquigarrow$ **Blackboard.**

Why is the Nim game a simple game?

Two useful properties of the Nim game.

1. "**Finite horizon**": the interaction necessarily has a **bounded length**:
   - guaranteed to end in a "terminal state" within a bounded number of moves;
   - we can represent all possible plays as a finite tree;
   - lends itself well to a *backward induction*.
2. The objective is very **simple**: to reach a certain state.

We will discuss why it is useful to **relax these two properties** for expressiveness.

Other properties that we will **not** relax in this talk

- There are **two players**.
- The games are **zero-sum**: when one player wins, the other loses.
- The games are **turn-based**: only one player plays at a time.
- The games are **perfect-information**: players always know exactly what moves are played.
- The games are **deterministic**: no random transitions.

How to describe the strategies for the Nim game?

- Let V be the set of all possible **game states**.
  - Here, each state is described as the number of matchsticks remaining and the current player.
- Let $V_1 \subseteq V$ be the set of all **states where Player 1 is to move**.
- Let $V_2 \subseteq V$ be the set of all **states where Player 2 is to move**.
- Let $E \subseteq V \times V$ be the set of all possible **moves**.

What mathematical object is a **strategy** here?

First definition of a strategy

A **strategy** for Player ℓ ($\ell \in \{1, 2\}$) is a function that **observes the current state** of Player ℓ and decides what edge of the graph to follow; formally, it is a function

$$\sigma_\ell \colon V_\ell \to V$$

such that for all $v \in V_\ell$, $(v, \sigma_\ell(v)) \in E$.

Winning strategies

A **play** is a path $\rho = v_0 \to v_1 \to \cdots \to v_k$ in the game graph such that v_k is a terminal state (i.e., a state with no outgoing edges).

Play induced by a **pair** of strategies

Given an initial state v_0, σ_1 **is a strategy of Player 1**, and σ_2 **is a strategy of Player 2**, we can define a **unique play** $\rho_{v_0}^{\sigma_1,\sigma_2}$ as follows:

- The play starts at v_0.
- If $v_i \in V_1$, the next state v_{i+1} is $\sigma_1(v_i)$; if $v_i \in V_2$, the next state v_{i+1} is $\sigma_2(v_i)$.

A strategy σ_1 of Player 1 is **winning** from a state v_0 if, when **sticking to this strategy**, **no matter what Player 2 plays**, Player 1 wins.

Formally: if for all strategies σ_2 of Player 2, the play $\rho_{v_0}^{\sigma_1,\sigma_2}$ is winning for Player 1.

Which player has a winning strategy?

Let us rephrase **the existence of a winning strategy** from a state v_0:

- For Player 1: $\exists \sigma_1 \in \Sigma_1, \forall \sigma_2 \in \Sigma_2, \rho_{v_0}^{\sigma_1,\sigma_2}$ is winning for Player 1.
- For Player 2: $\exists \sigma_2 \in \Sigma_2, \forall \sigma_1 \in \Sigma_1, \rho_{v_0}^{\sigma_1,\sigma_2}$ is winning for Player 2.

These two statements are

- **mutually exclusive**: if a player has a winning strategy, the other player cannot have one...
- but **not negations of each other**: if a player does not have a winning strategy, it is not obvious that the other player has one!

For instance, the negation of the first statement is

$$\forall \sigma_1 \in \Sigma_1, \exists \sigma_2 \in \Sigma_2, \rho_{v_0}^{\sigma_1,\sigma_2} \text{ is winning for Player 2,}$$

which is weaker than stating that Player 2 has a winning strategy.

It could be that for all strategies of Player 1, Player 2 has a **counter strategy**,
yet Player 2 has no “uniformly” winning strategy.

Determinacy

- A game in which one of the players has a winning strategy is said to be **determined**.
- The above discussion suggests that **not all games may be determined**. Yet...

Zermelo's theorem for win/lose games (1913)

All **finite-horizon**, **two-player**, **zero-sum** games of **perfect information** are **determined**; one of the players has a winning strategy.

Proof: essentially a simple **backward induction** like we did for the Nim game!

Sometimes regarded as the **first result in game theory**.

Example: Chess

- Chess is a two-player, zero-sum, turn-based game of perfect information.
- The game tree is *huge*, but **finite** (thanks to the *threefold-repetition draw* rule).

Zermelo's theorem for win/lose/**draw** games

Using Zermelo's theorem (for win/lose/**draw** games), we can conclude that one of the following three statements is true for chess:

1. Player 1 has a winning strategy.
2. Player 2 has a winning strategy.
3. Both players can enforce a draw.

$\rightsquigarrow$ Yet, we don't know *which* of these statements is true!
There are an estimated 10^{120} possible chess games.

Applies to **other board games**: Tic-tac-toe, Connect 4, checkers, Go...
but **not** poker or Stratego (incomplete information).

What does a non-determined game looks like?

- We will see that exhibiting a **non-determined** *turn-based* game is challenging.
- However, if we relax the turn-based assumption and allow for **concurrent** moves, we can exhibit a non-determined game more easily.
- **Example**: using our current definitions of strategy and winning strategy, *rock-paper-scissors* is a non-determined game: for every strategy of Player 1, there exists a counter-strategy for Player 2.
- For concurrent games, the model of strategies we consider is too weak: some **randomness** may be useful to take the opponent by surprise... $\rightsquigarrow$ *Not for this talk* ☺

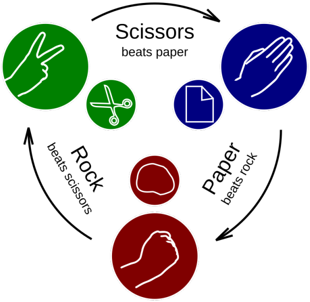


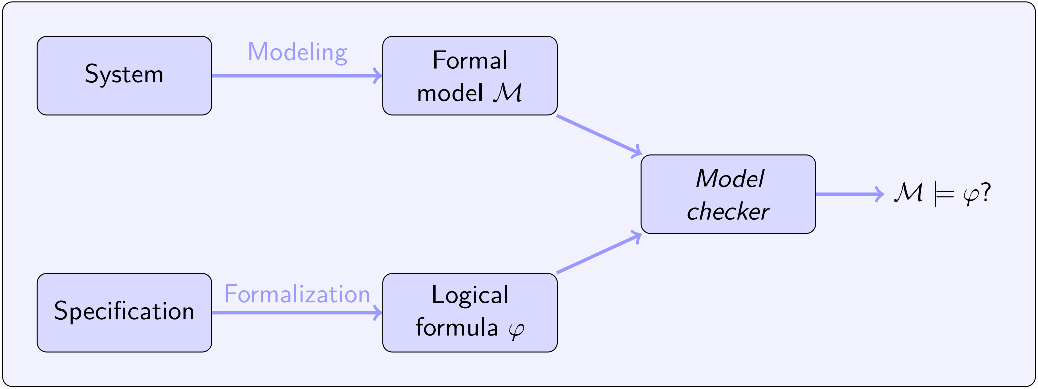
Table of contents

Motivation: reactive systems

- **Reactive systems** = systems that interact continuously with their environment.
  Examples: web server, robot vacuum cleaner “Roomba®”, elevator. . .
- **React** to uncontrollable events from their environment while achieving an **objective**.
- Subject to **errors**, sometimes severe (financial losses, deaths).
- Solution 1: **tests**? Not exhaustive.
- Solution 2: **formal verification** and **synthesis**.

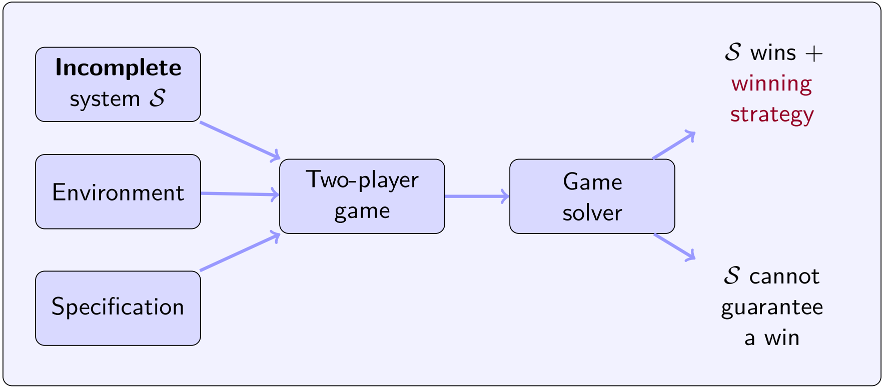
Formal verification

- We want a **formal proof** of the correct behavior of a system.
- We work with **models**/abstractions of systems.
- **Specification**: description of the acceptable behaviors of the system.



Synthesizing a controller

- More ambitious: automatically generate a **controller** that guarantees the specification.
- **Incomplete** definition of the system.
- Environment seen as an **antagonistic** player.



Modeling through **game theory**.

Game-theoretic metaphor for synthesis

- Two-player **graph game** capturing the **states of the system**.
- Certain vertices ◯ controlled by **the system** (**Player 1**), others □ by **the environment** (**Player 2**).
- For generality, we assume an interaction of **infinite duration** between the two players. Useful to model, e.g., a web server that handles requests indefinitely, or a Roomba that must vacuum for eternity.
- We define an **objective** such that Player 1 wins iff the system achieves **its specification**.

Table of contents

Reachability games

- We now **relax the finite-horizon hypothesis**: game graphs can have cycles.
- A **game graph** is a tuple $\mathcal{G} = (V, V_1, V_2, E)$ with $V = V_1 \uplus V_2$ and $E \subseteq V \times V$.
- We assume there is no terminating state: for convenience, all states have an outgoing edge.
- A **play** is now an **infinite path** $v_0 \to v_1 \to \cdots$ in the game graph.
- What is the players' objectives? We assume there are **colors** from a set C labeling the states through a function $\mathrm{col}\colon V \to C$.

Reachability objective

A **reachability objective** can be defined with $C = \{\top, \bot\}$:

- the objective of Player 1 is to reach a state labeled with $\top$;
- still **zero-sum**, so the objective of Player 2 is to prevent this from happening (*forever*).

Example (blackboard).

How to solve reachability games? (1/2)

We want an **algorithm** that decides whether Player 1 has a winning strategy from a state v_0.

Algorithm for reachability games

We compute iteratively **all the states from which Player 1 wins**:

- We start with
$$T_0 = \{v \in V \mid \mathrm{col}(v) = \top\}.$$
If we start in such a state, Player 1 wins in 0 move!
- Then, we iteratively expand this set:
$$T_{i+1} = T_i \cup \{v \in V_1 \mid \exists u \in T_i, (v, u) \in E\} \cup \{v \in V_2 \mid \forall u, (v, u) \in E \Rightarrow u \in T_i\}.$$
- The sequence $(T_i)_{i \geq 0}$ is non-decreasing: at some point, we reach a **fixed point** $T_k = T_{k+1}$.

How to solve reachability games? (2/2)

Reminder:

$$T_0 = \{v \in V \mid \mathrm{col}(v) = \top\},$$
$$T_{i+1} = T_i \cup \{v \in V_1 \mid \exists u \in T_i, (v, u) \in E\} \cup \{v \in V_2 \mid \forall u, (v, u) \in E \Rightarrow u \in T_i\}.$$

Theorem

For k such that $T_k = T_{k+1}$, Player 1 has a winning strategy from all states in T_k.
Player 2 has a winning strategy from all states in $V \setminus T_k$.

Blackboard proof.

The set T_k is an **attractor**: the states from which Player 1 can **attract** Player 2 to a $\top$-state.

Corollary: complexity of solving reachability games

Computing the winning regions is doable in linear time: $\mathcal{O}(|V| + |E|)$.

Strategies for reachability games

As a by-product, we obtain the **determinacy of reachability games**.

Determinacy of reachability games

In a **reachability** game, from all states, **either Player 1 or Player 2 has a winning strategy**.

But the proof also shows **what winning strategies look like**: for Player ℓ ($\ell \in \{1, 2\}$), they are functions

$$\sigma_\ell \colon V_\ell \to V.$$

Such a strategy is called **memoryless**: it only observes the current state, not the past interaction. Never useful to try another move if revisiting the same state.

Memoryless determinacy of reachability games

In a reachability game, from all states, either Player 1 or Player 2 has a **memoryless** winning strategy.

Curiosity: infinite game graphs, ordinals

- Our algorithm terminates for **finite** game graphs.
- It may not terminate for **infinite** game graphs.
  - **Blackboard example.**
- However, it would still work if we could apply it **transfinitely** many times!
- For instance, apply the operator infinitely many times... *and then apply it just one more time*.
- This can be used to show that even reachability games on **infinite game graphs** are **memoryless-determined**.
- **Exercise**: Find a reachability game that requires ω^2 (or ω^ω) iterations to solve.

Table of contents

Summary up to now

- We have studied **reachability games**, which generalize finite-horizon games.
- They are **determined**, and even **memoryless-determined**.
- We will now consider **more complex objectives**.
- First question: what do we mean by *objective* in general?

Game objectives

- When both players stick to a strategy, they generate a **play**, which induces an element of C^ω.
  - $C^\omega = \{c_0 c_1 \ldots \mid \forall i \geq 0, c_i \in C\}$ is the set of infinite sequences of colors.
- To specify an objective, it suffices to specify all sequences that Player 1 is happy to obtain.

Definition of objective

An **objective** for Player 1 is a set $\mathcal{O} \subseteq C^\omega$ of infinite sequences of colors.

As games are **zero-sum**, the objective of Player 2 is $C^\omega \setminus \mathcal{O}$.
In this framework, the reachability objective is

$$\text{Reach}(\top) = \{c_0 c_1 c_2 \ldots \in C^\omega \mid \exists i \geq 0, c_i = \top\}.$$

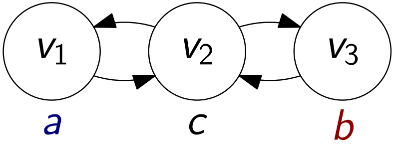
Its complement is the *safety objective*

$$\text{Safe}(\top) = \{c_0 c_1 c_2 \ldots \in C^\omega \mid \forall i \geq 0, c_i \neq \top\}.$$

Memoryless strategies do not always suffice

- $C = \{a, b, c\}$.
- Objective: see infinitely often a **and** infinitely often b:

$$\mathcal{O} = \{c_0 c_1 \ldots \in C^\omega \mid \exists^\infty i \geq 0, c_i = a \land \exists^\infty i \geq 0, c_i = b\}.$$



- In this game, Player 1 wins by playing $acbcacbc\ldots$ but **not in a memoryless way**!
- We need to define a more general kind of strategy...

More general definition of **strategy**

A **history** is a finite path $v_0 \to v_1 \to \cdots \to v_n \in V^*$ of the game graph.

For $\ell \in \{1, 2\}$, we denote by $\mathsf{Hists}_\ell(\mathcal{G})$ the histories $v_0 v_1 \dots v_n$ such that $v_n \in V_\ell$.

General definition of a strategy

A **strategy** of $\mathcal{P}_\ell$ is a function $\sigma\colon \mathsf{Hists}_\ell(\mathcal{G}) \to V$ such that if $\sigma(v_0 v_1 \dots v_i) = v_{i+1}$, then (v_i, v_{i+1}) is an edge of $\mathcal{G}$.

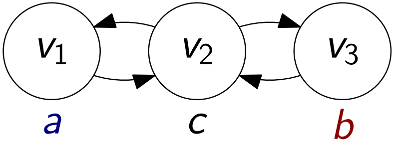
Less convenient for **implementation** purposes:

- there are infinitely many strategies, so you cannot try them all;
- $\mathsf{Hists}_\ell(\mathcal{G})$ is infinite, so representing the strategy in your computer may be challenging.

Back to the previous example

- Memoryless strategies do not suffice for the previous example.
- $C = \{a, b, c\}$:

$$\mathcal{O} = \{c_1 c_2 \ldots \in C^\omega \mid \exists^\infty i \geq 1, c_i = a \land \exists^\infty i \geq 1, c_i = b\}.$$



- But we would still like something implementable!
- Compromise: use **finite memory**. Here, it suffices to remember if we just saw a or b!

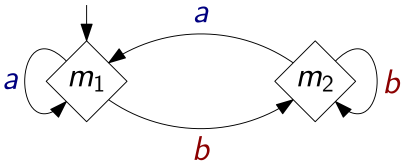
Finite-memory strategy

- We **condense** information from histories $\text{Hists}_\ell(\mathcal{G})$ into a **finite object**
  $\rightsquigarrow$ loss of information, but hopefully sufficient to make decisions!
- A common computational model derives from **automata**.

Definition

Memory structure $(M, m_{\text{init}}, \alpha_{\text{upd}})$:
finite set of states M, initial state $m_{\text{init}} \in M$, update function $\alpha_{\text{upd}} \colon M \times C \to M$.

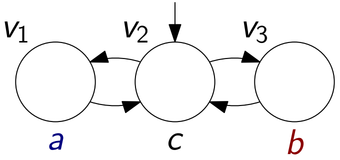
- Example to remember if a or b was seen last:

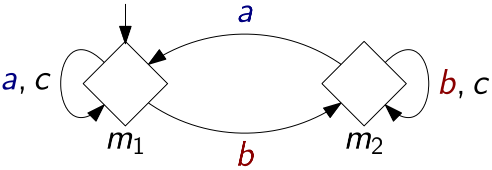


- To play, we rely on the current state of $\mathcal{G}$ **and** on the current state of the memory (here, m_1 or m_2).

Illustration

We define a winning strategy $\sigma\colon V_i \times M \to V$.

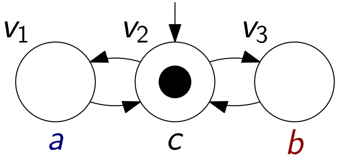


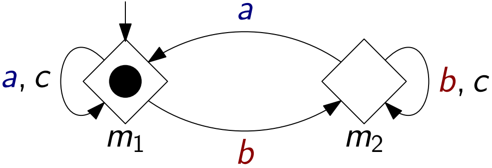


- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Illustration

We define a winning strategy $\sigma\colon V_i \times M \to V$.

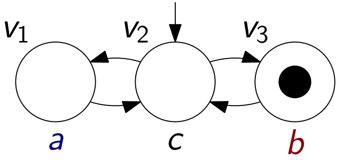




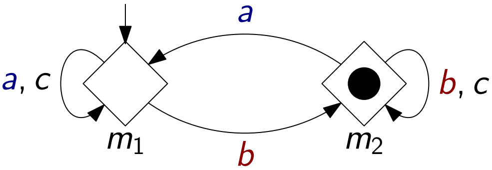
- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Illustration

We define a winning strategy $\sigma \colon V_i \times M \to V$.



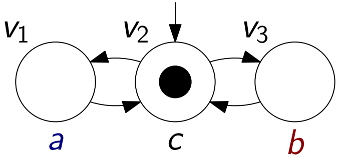
$$\sigma(v_2, m_1) = v_3$$



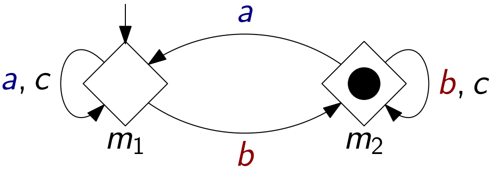
- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Illustration

We define a winning strategy $\sigma\colon V_i \times M \to V$.



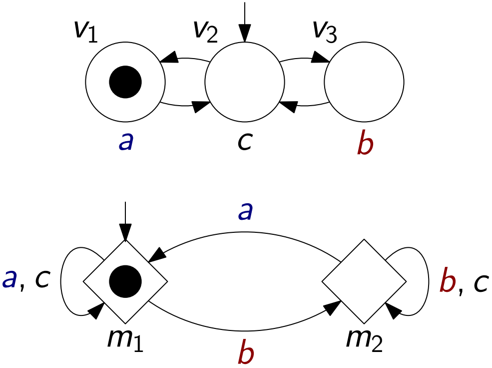
$$\sigma(v_2, m_1) = v_3$$
$$\sigma(v_3, \_) = v_2$$



- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Illustration

We define a winning strategy $\sigma\colon V_i \times M \to V$.

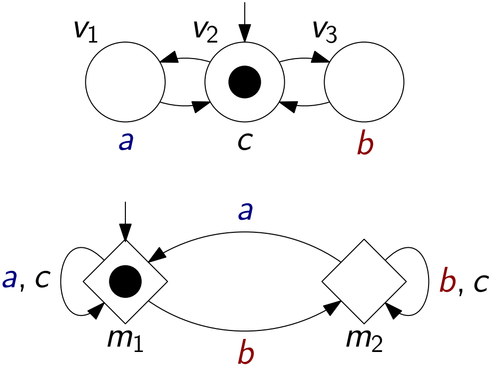


$$\sigma(v_2, m_1) = v_3$$
$$\sigma(v_3, \_) = v_2$$
$$\sigma(v_2, m_2) = v_1$$

- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Illustration

We define a winning strategy $\sigma\colon V_i \times M \to V$.



$$\sigma(v_2, m_1) = v_3$$
$$\sigma(v_3, \_) = v_2$$
$$\sigma(v_2, m_2) = v_1$$
$$\sigma(v_1, \_) = v_2$$

- This information from this memory structure is **sufficient to win in this graph**.
- Actually, this is more general: in **any game graph**, if winning is possible, then this structure is sufficient! $\rightsquigarrow$ We will discuss why.
- We say that objective $\mathcal{O}$ is **finite-memory determined**.

Product game

Another way to look at memory is through the product game.

$$\text{Playing with memory } \mathcal{M} \text{ in game graph } \mathcal{G}$$
$$\approx$$
$$\text{Playing memoryless in the game graph } \mathcal{G} \ltimes \mathcal{M}$$

Blackboard illustration.

- In the first case, the state space is V and the strategy looks at M as well.
- In the second case, the state space is $V \times M$ and the strategy is memoryless.

Memory corresponds to **additional information** to **“inject” in the game graph** to make **memoryless strategies sufficient**.

Finite memory is not always sufficient

- Unfortunately, sometimes, even **finite memory** is **insufficient**.
- Let $C = \{-1, 0, +1\}$.
- Objective: either there are only $+1$, or the sum of colors eventually stabilizes to 0:

$$\mathcal{O} = \{(+1)^\omega\} \cup \{c_0 c_1 \ldots \in C^\omega \mid \lim_{n\to\infty} \sum_{i=0}^{n} c_i \text{ exists and is } 0\}.$$

Blackboard game graph.

- This objective requires **infinite memory** in some game graphs! There is a winning strategy, but no (finite) memory structure suffices, as counting "to infinity" must be possible.

Strategy complexity

- Hierarchy of strategies:

$$\text{Memoryless } (V_\ell \to V) \subsetneq \text{Finite memory } (V_\ell \times M \to V) \subsetneq \text{General } (\mathsf{Hists}_\ell(\mathcal{G}) \to V).$$

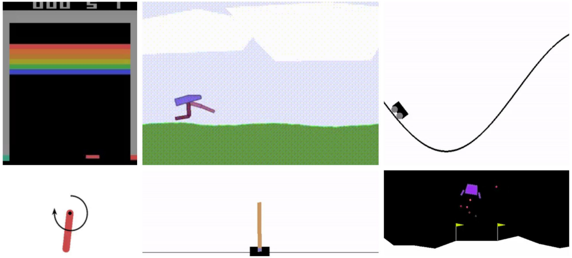
- Research agenda: understand in which contexts **simple** strategies suffice.
  - Classes of game graphs (finite, infinite, *stochastic*, etc.).
  - Classes of objectives ($\mathcal{O} \subseteq C^\omega$, maximizing a function $f\colon C^\omega \to \mathbb{R}$, maximizing the probability of an event, etc.).
- Algorithms, complexity of computing the amount of memory needed for a given objective.

Why study strategy complexity?

- Finite bounds on the size of strategies usually leads to the **decidability** of the synthesis problem.
    - Over finite game graphs, there are then **finitely many strategies** to consider.
- Trying them all works but is not efficient; strategy complexity gives bounds on the search space, **helping design more efficient algorithms**.
- For implementations, we like having **compact controllers**.

Aside: Reinforcement learning

- A related area is **reinforcement learning**, a subfield of machine learning concerned with how agents take actions in environments to achieve some objective.
- Most reinforcement learning algorithms (such as *Q-learning*) **assume memoryless strategies suffice**: they learn one action to play for each state.
- **Crucial to understand strategy complexity** to learn decisions for **complex objectives**!



Gymnasium environments

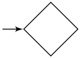
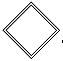
Table of contents

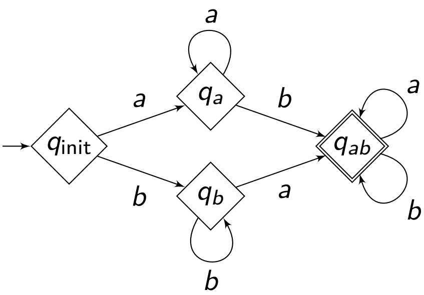
ω-regular objectives

- The **ω-regular objectives** are very common objectives.
- As we will see, they hold also a special place with respect to strategy complexity.
- Before defining them, we introduce **regular objectives**.

Table of contents

Regular objectives (1/2)

Finite automata are often used to define sets of **finite** words. They accept the finite words that can be read from the **initial state**  to the **final state** .



This automaton

- accepts *aab* ✔
- rejects *aa* ✗
- accepts *baab* ✔
- ...

This automaton accepts exactly finite words that see both *a* and *b*.

Exercise

Let $C = \{a, b\}$.

Build a finite automaton that accepts all finite words containing **two** a**'s in a row**.

Regular objectives (2/2)

Sets of words that can be defined by such a finite automaton are called **regular**.

Strategy complexity of regular objectives

Assume the objective of Player 1 is to achieve a word from a regular set L (i.e., $\mathcal{O} = LC^\omega$). Then, a deterministic automaton recognizing L **always suffices as a memory structure to implement winning strategies**.

Proof: If we take the product of the game graph with the automaton, we reduce to a standard reachability objective on the product, which is memoryless-determined!

Blackboard example.

In particular, games with regular objectives are **finite-memory determined**!

From reachability to regular objectives

From

the **memoryless determinacy of reachability objectives**,

we have deduced easily

the **finite-memory determinacy of regular objectives**.

Are there other "canonical" objectives, such as reachability, that we could exploit?

Table of contents

More complex objectives

Remember the objective

$$\mathcal{O} = \{c_0 c_1 \ldots \in C^\omega \mid \exists^\infty i \geq 0, c_i = a \land \exists^\infty i \geq 0, c_i = b\}.$$

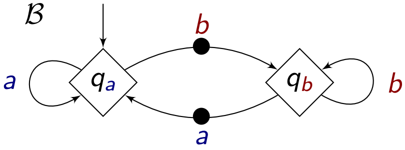
It is **not** a regular objective ☹.

Can we still capture it with a more general class of automata? YES!

Deterministic Büchi automata

A **deterministic Büchi automaton** $\mathcal{B}$ on C

- reads **infinite** words (in C^ω),
- accepts words that see infinitely many **Büchi transitions** $\bullet$.



This automaton

- accepts *abababab a*... ✔
- accepts *aabaab*... ✔
- rejects *bbbaaaaa*... ✗
- ...

What is the set of words accepted by this automaton?

$$\{w \in \{a, b\}^\omega \mid w \text{ sees } \infty\text{ly many } a \text{ and } \infty\text{ly many } b\}$$

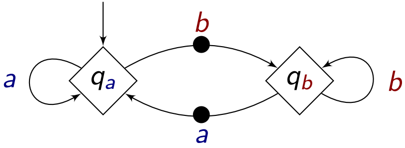
Exercise

Let $C = \{a, b\}$.

Build a deterministic Büchi automaton that accepts all infinite words containing **infinitely many** a**'s**, or **two** a**'s in a row** at some point.

Link with strategy complexity

Do you recognize the following automaton?



- It has the same structure as the **memory structure we used to win for this objective**!
- Instead of a **reachability acceptance condition**, we use a **Büchi acceptance condition**.
- A **Büchi objective** requires to see some color infinitely often:

$$\text{Büchi}(\top) = \{c_0 c_1 \ldots \in C^\omega \mid \exists^\infty i \geq 0, c_i = \top\}.$$

- It turns out Büchi objectives are **also** memoryless-determined!

Memoryless determinacy of Büchi objectives

In a game with a Büchi objective, from all states, either Player 1 or Player 2 has a **memoryless** winning strategy.

From Büchi objectives to objectives recognizable by a Büchi automaton

From

the **memoryless determinacy of Büchi objectives**,

we can deduce

the **finite-memory determinacy of objectives recognizable by a deterministic Büchi automaton**.

Proof: By taking the product of the game graph with a deterministic Büchi automaton recognizing the objective, we reduce to a standard Büchi objective on the product game, which is memoryless-determined!

The need for determinism

- Observe that our memory structures are **deterministic**: when reading a color from a given state, there is always exactly **one** possible transition.
- Some objectives are only recognizable by **non-deterministic** Büchi automata...
- This is a problem to use them as memory structures ☹

Example: the complement of a Büchi objective is a coBüchi objective:

$$\text{coBüchi}(\top) = \{c_0c_1 \ldots \in C^\omega \mid \text{there are at most finitely many } i\text{'s s.t. } c_i = \top\}.$$

Proposition

There is a **non-deterministic** Büchi automaton recognizing coBüchi($\top$),
but no **deterministic** Büchi automaton.

Blackboard proof.

Non-deterministic Büchi automata

The objectives recognized by non-deterministic Büchi automata are the

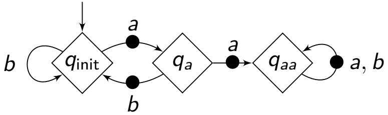
ω-regular objectives.

They are **canonical** in that they have multiple equivalent representations:

ω-regular expressions

$b^* a b^* a C^\omega$

ω-automata



Linear temporal logic (LTL)

$\mathbf{GF}a$

They are also closed under union, intersection, and complement.

We would like to understand their **determinacy**.

Determinacy of ω-regular objectives

As a first observation, we can use the following **big** theorem:

Theorem (Martin, 1975)

All games with **Borel objectives** are determined.

No definition of Borel objectives here; however...

- to define a non-Borel objective, you need the **axiom of choice**;
- this implies that non-determined games are necessarily at least *a bit strange*!
- Borel objectives are ***much*** more general than ω-regular objectives!

Corollary

All games with ω-regular objectives are determined.

Can we obtain a stronger kind of determinacy?

What we want

- Büchi automata were introduced by Büchi in the 1960s.[1]
- First kind of automata on **infinite** words.
- The issue here is that they need **non-determinism** to recognize all ω-regular objectives $\rightsquigarrow$ not good for memory structures.

We are looking for

- a class of **deterministic** automata that recognize all ω-regular objectives,
- while using a **memoryless-determined** acceptance condition?

There is exactly such a class!

[1] Büchi and Landweber, "Definability in the Monadic Second-Order Theory of Successor", 1969.

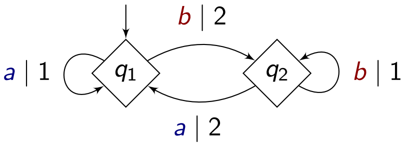
Table of contents

Parity automata

- We still consider deterministic automata reading infinite words, but we change the acceptance condition.
- We assume transitions are labeled by integers in a set $\{0, 1, \ldots, d\}$.

An **infinite** word is accepted if
the largest integer seen infinitely often is even.

Example, $C = \{a, b\}$:



- Word $aabaabaab\ldots = (aab)^\omega \rightsquigarrow 112212212\ldots = 112(212)^\omega$. ✔
- Word $abaaa\ldots = aba^\omega \rightsquigarrow 12211\ldots = 1221^\omega$. ✗
- ...

$$\mathcal{O} = \{w \in C^\omega \mid a \text{ is seen } \infty\text{ly often and } b \text{ is seen } \infty\text{ly often along } w\}$$

Exercise

Let $C = \{a, b\}$.

Build a parity automaton recognizing the set of words that **eventually end with** *abababab*...
(i.e., $C^*(ab)^\omega$)?

Parity games

- Let $C = \{0, 1, \ldots, d\}$ for some $d \in \mathbb{N}$.
- The **parity objective** is defined as follows: a play is winning for Player 1 if the highest color that appears infinitely often is even.
- Formally,

$$\mathsf{Parity}(C) = \{c_0 c_1 \ldots \in C^\omega \mid \limsup_{n \to \infty} c_n \text{ is even}\}.$$

Memoryless determinacy of parity games [Emerson, Jutla, 1991]

Games with a parity objective are **memoryless-determined**.

From parity objectives to ω-regular objectives

From

the **memoryless determinacy of parity objectives**,

and

the fact that **deterministic parity automata recognize all ω-regular objectives**,

we can deduce

the **finite-memory determinacy of ω-regular objectives**.

Proof: By taking the product of the game graph with a deterministic parity automaton recognizing an ω-regular objective, we reduce to a standard parity objective on the product game, which is memoryless-determined!

Conclusion

- The finite-memory determinacy of ω-regular objectives is arguably **the most important result in the theory of infinite games**.
- First shown by Rabin in 1969 for the **decidability of a logical theory** (*S2S*), in a much more complex form.[2]
- Subsequent articles greatly simplified the proof, **with a more direct use of games**.[3]
- Today, this result is still heavily used to **solve synthesis problems**.[4]
- All competitive synthesis algorithms reduce to a parity game, then solve the parity game.

[2] Rabin, "Decidability of Second-Order Theories and Automata on Infinite Trees", 1969.

[3] Gurevich and Harrington, "Trees, Automata, and Games", 1982; Emerson and Jutla, "Tree Automata, Mu-Calculus and Determinacy (Extended Abstract)", 1991.

[4] Jacobs et al., "The Reactive Synthesis Competition (SYNTCOMP): 2018-2021", 2024.

Two open problems for the future

Open problem #1

What is the complexity of *solving* parity games?

- They are in NP $\cap$ coNP.[5]
- Main breakthrough (2017):[6] they can be solved in **quasi-polynomial time**: $\sim n^{\log d}$.
- **Can they be solved in polynomial time?**

Open problem #2

How to find the smallest possible *memory structure* for a given ω-regular objective?

- The parity automaton suffices, but not always minimal!
- Recent breakthrough (2025): the related decision problem is in NP.[7] Not known to be in P.

Thanks!

[5] Follows from their memoryless determinacy: **exercise**!

[6] Calude et al., "Deciding parity games in quasipolynomial time", 2017.

[7] Casares and Ohlmann, "The Memory of ω-Regular and BC(Σ_0^2) Objectives", 2025.