Infinite Games

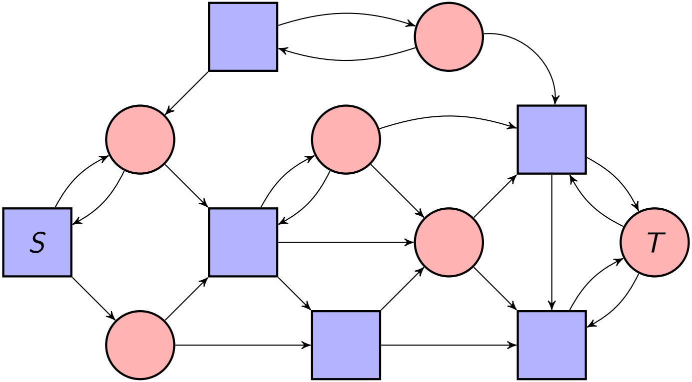
Martin Zimmermann

Saarland University

October 21st, 2014

Research Training Group SCARE, Oldenburg, Germany

Let's Play



You move at circles and want to reach T from S.

Motivation

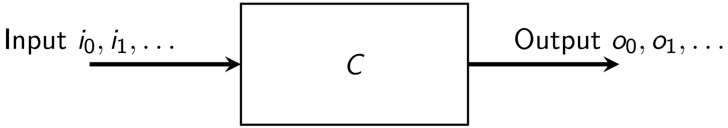
- Model-checking and satisfiability for fixed-point logics, e.g., the modal μ-calculus, CTL, CTL*.
- Automata emptiness often expressible in terms of games.
- Semantics of alternating automata in terms of games.
- Synthesis of correct-by-construction controllers for reactive systems (non-terminating, interacting with antagonistic environment).

Motivation

- Model-checking and satisfiability for fixed-point logics, e.g., the modal μ-calculus, CTL, CTL*.
- Automata emptiness often expressible in terms of games.
- Semantics of alternating automata in terms of games.
- Synthesis of correct-by-construction controllers for reactive systems (non-terminating, interacting with antagonistic environment).

Earliest appearance: **Church's problem (1957)**

Given requirement φ on input-output behavior of boolean circuits, compute a circuit C that satisfies φ (or prove that none exists).



Motivation

- Model-checking and satisfiability for fixed-point logics, e.g., the modal μ-calculus, CTL, CTL*.
- Automata emptiness often expressible in terms of games.
- Semantics of alternating automata in terms of games.
- Synthesis of correct-by-construction controllers for reactive systems (non-terminating, interacting with antagonistic environment).

Earliest appearance: **Church's problem (1957)**

Given requirement φ on input-output behavior of boolean circuits, compute a circuit C that satisfies φ (or prove that none exists).

Game theoretic formulation:

- Player 0 generates infinite stream of input bits.
- Player 1 has to answer each input bit by output bit.
- Player 1 wins, if combination of streams satisfies φ.

Church's Problem: Example

φ is conjunction of following properties:

1. Whenever the input bit is 1, then the output bit is 1, too.
2. If there are infinitely many 0's in the input stream, then there are infinitely many 0's in the output stream.
3. At least one out of every three consecutive output bits is a 1.

Church's Problem: Example

φ is conjunction of following properties:

1. Whenever the input bit is 1, then the output bit is 1, too.
2. If there are infinitely many 0's in the input stream, then there are infinitely many 0's in the output stream.
3. At least one out of every three consecutive output bits is a 1.

Winning strategy for the output player:

- Answer every 1 by a 1.
- Answer every 0 by a 0,

Church's Problem: Example

φ is conjunction of following properties:

1. Whenever the input bit is 1, then the output bit is 1, too.
2. If there are infinitely many 0's in the input stream, then there are infinitely many 0's in the output stream.
3. At least one out of every three consecutive output bits is a 1.

Winning strategy for the output player:

- Answer every 1 by a 1.
- Answer every 0 by a 0, unless it would be the third 0 in a row. Then, answer by a 1.

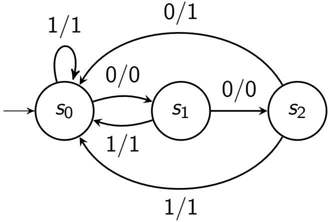
Church's Problem: Example

φ is conjunction of following properties:

1. Whenever the input bit is 1, then the output bit is 1, too.
2. If there are infinitely many 0's in the input stream, then there are infinitely many 0's in the output stream.
3. At least one out of every three consecutive output bits is a 1.

Winning strategy for the output player:

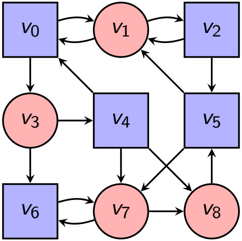
- Answer every 1 by a 1.
- Answer every 0 by a 0, unless it would be the third 0 in a row. Then, answer by a 1.



Outline

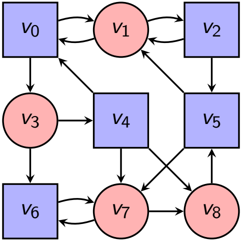
Arenas and Games

- An *arena* $\mathcal{A} = (V, V_0, V_1, E)$ consists of
  - a finite set V of vertices,
  - a set $V_0 \subseteq V$ of vertices owned by Player 0 (circles),
  - the set $V_1 = V \setminus V_0$ of vertices owned by Player 1 (squares),
  - a directed edge-relation $E \subseteq V \times V$.



Arenas and Games

- An *arena* $\mathcal{A} = (V, V_0, V_1, E)$ consists of
  - a finite set V of vertices,
  - a set $V_0 \subseteq V$ of vertices owned by Player 0 (circles),
  - the set $V_1 = V \setminus V_0$ of vertices owned by Player 1 (squares),
  - a directed edge-relation $E \subseteq V \times V$.



- A play is an infinite path through $\mathcal{A}$.

Strategies

- A strategy for Player i in $\mathcal{A}$ is a mapping $\sigma \colon V^* V_i \to V$ satisfying $(v_n, \sigma(v_0 \cdots v_n)) \in E$ (only legal moves).

Strategies

- A strategy for Player i in $\mathcal{A}$ is a mapping $\sigma \colon V^* V_i \to V$ satisfying $(v_n, \sigma(v_0 \cdots v_n)) \in E$ (only legal moves).
- A play $v_0 v_1 v_2 \cdots$ is consistent with σ, if $v_{n+1} = \sigma(v_0 \cdots v_n)$ for every n with $v_n \in V_i$.

Strategies

- A strategy for Player i in $\mathcal{A}$ is a mapping $\sigma\colon V^*V_i \to V$ satisfying $(v_n, \sigma(v_0 \cdots v_n)) \in E$ (only legal moves).
- A play $v_0v_1v_2\cdots$ is consistent with σ, if $v_{n+1} = \sigma(v_0 \cdots v_n)$ for every n with $v_n \in V_i$.
- Note: if we fix an initial vertex and strategies σ and τ for Player 0 and Player 1, then there is a unique play that starts in v and is consistent with σ and τ.

Strategies

- A strategy for Player i in $\mathcal{A}$ is a mapping $\sigma \colon V^* V_i \to V$ satisfying $(v_n, \sigma(v_0 \cdots v_n)) \in E$ (only legal moves).
- A play $v_0 v_1 v_2 \cdots$ is consistent with σ, if $v_{n+1} = \sigma(v_0 \cdots v_n)$ for every n with $v_n \in V_i$.
- Note: if we fix an initial vertex and strategies σ and τ for Player 0 and Player 1, then there is a unique play that starts in v and is consistent with σ and τ.

Special types of strategies:

- Positional strategies: $\sigma(v_0 \cdots v_n) = \sigma(v_n)$ for all $v_0 \cdots v_n$: move only depends on position the token is at at the moment.

Strategies

- A strategy for Player i in $\mathcal{A}$ is a mapping $\sigma \colon V^* V_i \to V$ satisfying $(v_n, \sigma(v_0 \cdots v_n)) \in E$ (only legal moves).
- A play $v_0 v_1 v_2 \cdots$ is consistent with σ, if $v_{n+1} = \sigma(v_0 \cdots v_n)$ for every n with $v_n \in V_i$.
- Note: if we fix an initial vertex and strategies σ and τ for Player 0 and Player 1, then there is a unique play that starts in v and is consistent with σ and τ.

Special types of strategies:

- Positional strategies: $\sigma(v_0 \cdots v_n) = \sigma(v_n)$ for all $v_0 \cdots v_n$: move only depends on position the token is at at the moment.
- Finite-state strategies: implemented by DFA with output reading play prefix $v_0 \cdots v_n$ and outputting $\sigma(v_0 \cdots v_n)$.

Winning

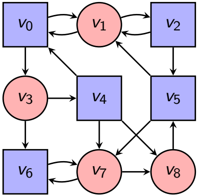
- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.

Winning

- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.
- Strategy σ for Player i is winning strategy from v, if every play that starts in v and is consistent with σ is winning for him.

Winning

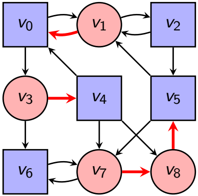
- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.
- Strategy σ for Player i is winning strategy from v, if every play that starts in v and is consistent with σ is winning for him.



$$\mathrm{Win} = \{ \rho \in V^\omega \mid \rho \text{ does not visit all vertices} \}$$

Winning

- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.
- Strategy σ for Player i is winning strategy from v, if every play that starts in v and is consistent with σ is winning for him.

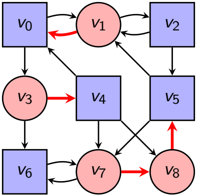


$$\mathrm{Win} = \{\, \rho \in V^\omega \mid \rho \text{ does not visit all vertices} \,\}$$

Player 0 wins from every vertex

Winning

- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.
- Strategy σ for Player i is winning strategy from v, if every play that starts in v and is consistent with σ is winning for him.



$$\mathrm{Win} = \{\, \rho \in V^\omega \mid \rho \text{ does not visit all vertices} \,\}$$

Player 0 wins from every vertex with positional strategies.

Winning

- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ consists of an arena $\mathcal{A}$ and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0.
- Set of winning plays for Player 1: $V^\omega \setminus \mathrm{Win}$.
- Strategy σ for Player i is winning strategy from v, if every play that starts in v and is consistent with σ is winning for him.

- Winning region $W_i(\mathcal{G})$: set of vertices from which Player i has a winning strategy.
- Always: $W_0(\mathcal{G}) \cap W_1(\mathcal{G}) = \emptyset$.
- $\mathcal{G}$ determined, if $W_0(\mathcal{G}) \cup W_1(\mathcal{G}) = V$.
- Solving a game: determine the winning regions and winning strategies.

Three Types of Winning Conditions

Win is an (possibly) infinite set of infinite words, and therefore unsuitable as input to an algorithm $\Rightarrow$ need finite representation.

Three Types of Winning Conditions

Win is an (possibly) infinite set of infinite words, and therefore unsuitable as input to an algorithm $\Rightarrow$ need finite representation.

- Reachability games: for $R \subseteq V$ define

$$\textsc{reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$

Three Types of Winning Conditions

Win is an (possibly) infinite set of infinite words, and therefore unsuitable as input to an algorithm $\Rightarrow$ need finite representation.

- Reachability games: for $R \subseteq V$ define

$$\textsc{reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$

- Parity games: for $\Omega\colon V \to \mathbb{N}$ define

$$\textsc{parity}(\Omega) = \{\, \rho \in V^\omega \mid \text{minimal priority seen infinitely often during } \rho \text{ is even} \,\}$$

Three Types of Winning Conditions

Win is an (possibly) infinite set of infinite words, and therefore unsuitable as input to an algorithm $\Rightarrow$ need finite representation.

- Reachability games: for $R \subseteq V$ define

$$\textsc{Reach}(R) = \{ \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \}$$

- Parity games: for $\Omega \colon V \to \mathbb{N}$ define

$$\textsc{Parity}(\Omega) = \{ \rho \in V^\omega \mid \text{minimal priority seen infinitely often during } \rho \text{ is even} \}$$

- Muller games: for $\mathcal{F} \subseteq 2^V$ define

$$\textsc{Muller}(\mathcal{F}) = \{ \rho \in V^\omega \mid \text{set of vertices seen infinitely often during } \rho \text{ is in } \mathcal{F} \}$$

Three Types of Winning Conditions

Win is an (possibly) infinite set of infinite words, and therefore unsuitable as input to an algorithm $\Rightarrow$ need finite representation.

- Reachability games: for $R \subseteq V$ define

$$\textsc{reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$

- Parity games: for $\Omega \colon V \to \mathbb{N}$ define

$$\textsc{parity}(\Omega) = \{\, \rho \in V^\omega \mid \text{minimal priority seen infinitely often during } \rho \text{ is even} \,\}$$

- Muller games: for $\mathcal{F} \subseteq 2^V$ define

$$\textsc{muller}(\mathcal{F}) = \{\, \rho \in V^\omega \mid \text{set of vertices seen infinitely often during } \rho \text{ is in } \mathcal{F} \,\}$$

There are many other winning conditions.

What Are We Interested in?

Given a type of winning condition (e.g., reachability, parity, Muller),..

- .. are games with this condition always determined?
- .. what kind of strategy do the players need (e.g., positional, finite-state)?
- .. if finite-state strategies are necessary, how large do they have to be?
- How hard is it to solve the game?

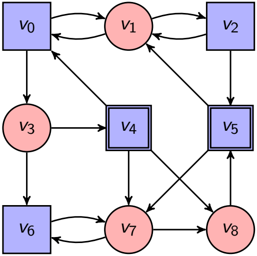
Outline

1. Definitions
2. **Reachability Games**
3. Parity Games
4. Muller Games
5. Outlook

Reachability Games

Reachability games: for $R \subseteq V$ define

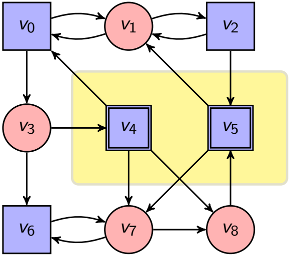
$$\textsc{Reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$



Reachability Games

Reachability games: for $R \subseteq V$ define

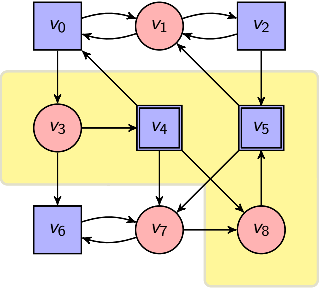
$$\textsc{reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$



Reachability Games

Reachability games: for $R \subseteq V$ define

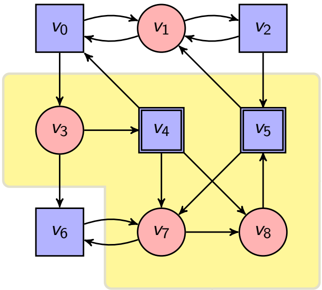
$$\textsc{Reach}(R) = \{\, \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \,\}$$



Reachability Games

Reachability games: for $R \subseteq V$ define

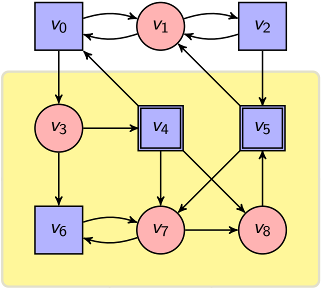
$$\textsc{reach}(R) = \{ \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \}$$



Reachability Games

Reachability games: for $R \subseteq V$ define

$$\textsc{Reach}(R) = \{ \rho \in V^\omega \mid \rho \text{ visits } R \text{ at least once} \}$$



Attractor Construction

$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Attractor Construction

$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n\in\mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Attractor Construction

$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.

Attractor Construction

$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n\in\mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.

Attractor Construction

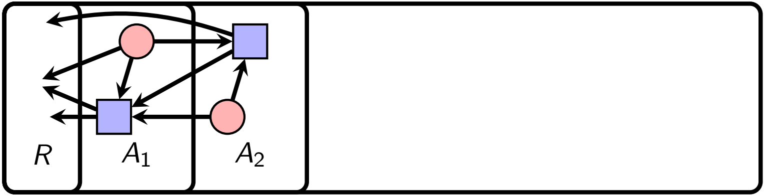
$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.



Attractor Construction

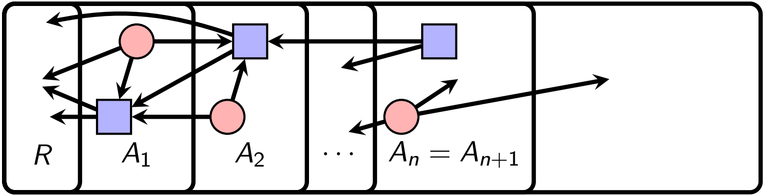
$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.



Attractor Construction

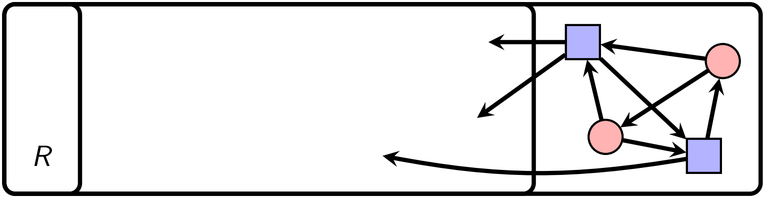
$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.



Attractor Construction

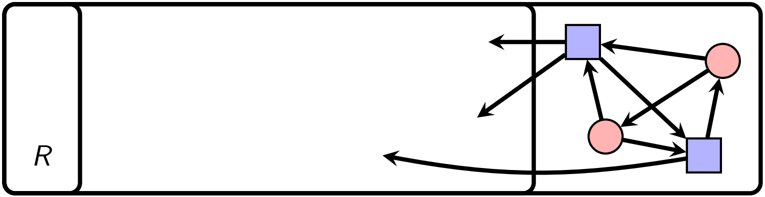
$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n\in\mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v,v') \in E \text{ s.t. } v' \in A_j\}$$
$$\cup \{v \in V_{1-i} \mid \forall (v,v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.



□

Attractor Construction

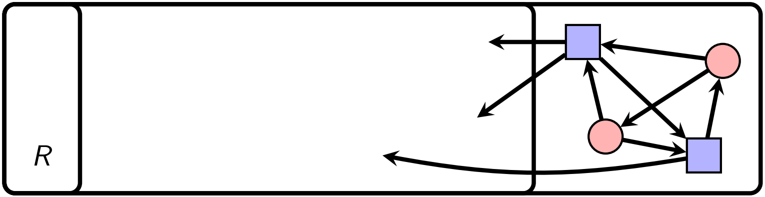
$\mathrm{Attr}_i^{\mathcal{A}}(R) = \bigcup_{n \in \mathbb{N}} A_n$ where $A_0 = R$ and

$$A_{j+1} = A_j \cup \{v \in V_i \mid \exists (v, v') \in E \text{ s.t. } v' \in A_j\} \cup \{v \in V_{1-i} \mid \forall (v, v') \in E \text{ we have } v' \in A_j\}$$

Theorem

Reachability games are determined with positional strategies.

Proof.



□

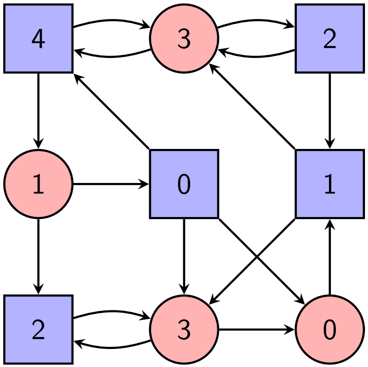
Remark: Attractors can be computed in linear time in $|E|$.

Outline

Parity Games

Parity games: for $\Omega \colon V \to \mathbb{N}$ define

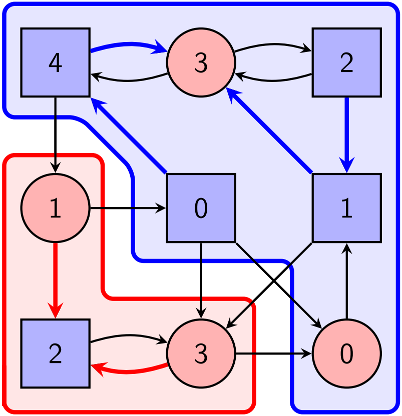
$$\textsc{parity}(\Omega) = \{\, \rho \in V^\omega \mid \text{minimal priority seen infinitely often during } \rho \text{ is even} \,\}$$



Parity Games

Parity games: for $\Omega\colon V \to \mathbb{N}$ define

$\textsc{parity}(\Omega) = \{ \rho \in V^\omega \mid$ minimal priority seen infinitely often during ρ is even $\}$



Parity Games

Applications:

- Normal form for ω-regular languages: deterministic parity automata.
- Model-checking game of the modal μ-calculus.
- Emptiness of parity tree automata equivalent to parity games.
- Semantics of alternating automata on infinite objects.

Parity Games

Applications:

- Normal form for ω-regular languages: deterministic parity automata.
- Model-checking game of the modal μ-calculus.
- Emptiness of parity tree automata equivalent to parity games.
- Semantics of alternating automata on infinite objects.

Theorem

Parity games are determined with positional strategies.

Parity Games

Applications:

- Normal form for ω-regular languages: deterministic parity automata.
- Model-checking game of the modal μ-calculus.
- Emptiness of parity tree automata equivalent to parity games.
- Semantics of alternating automata on infinite objects.

Theorem

Parity games are determined with positional strategies.

Proof Sketch:

By induction over the number n of vertices.

Parity Games

Applications:

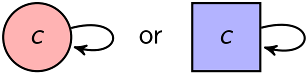
- Normal form for ω-regular languages: deterministic parity automata.
- Model-checking game of the modal μ-calculus.
- Emptiness of parity tree automata equivalent to parity games.
- Semantics of alternating automata on infinite objects.

Theorem

Parity games are determined with positional strategies.

Proof Sketch:

By induction over the number n of vertices. $n = 1$ trivial:



Parity Games

Applications:

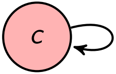
- Normal form for ω-regular languages: deterministic parity automata.
- Model-checking game of the modal μ-calculus.
- Emptiness of parity tree automata equivalent to parity games.
- Semantics of alternating automata on infinite objects.

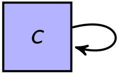
Theorem

Parity games are determined with positional strategies.

Proof Sketch:

By induction over the number n of vertices. $n = 1$ trivial:

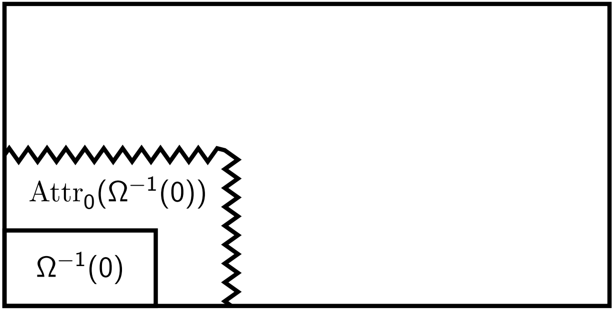


or


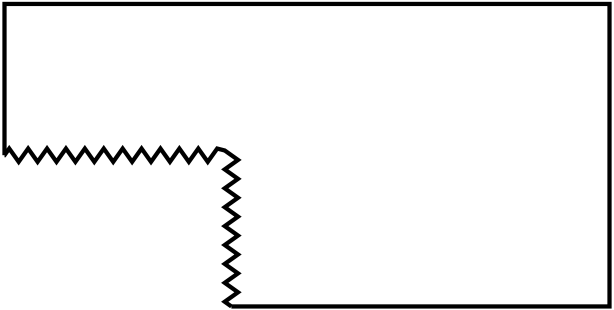
Player i wins iff $\mathrm{Par}(c) = i$

Proof Sketch

Now $n > 1$ and $\min \Omega(V) = 0$.

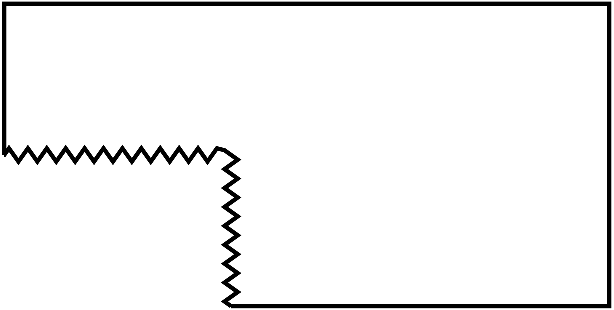


Proof Sketch



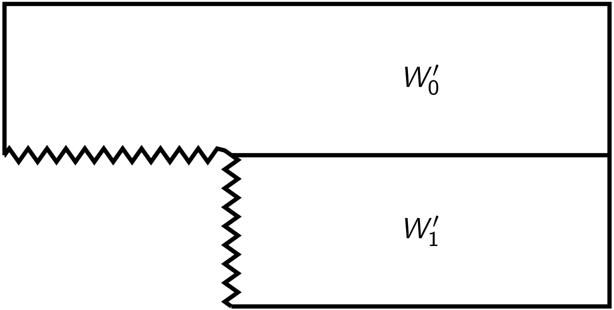
Proof Sketch

Induction hypothesis applicable..



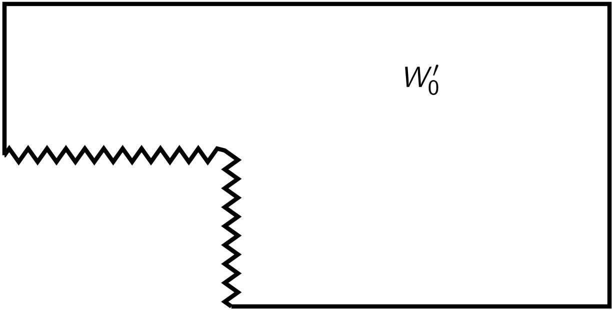
Proof Sketch

.. yields winning regions W_i' and positional strategies σ', τ'.



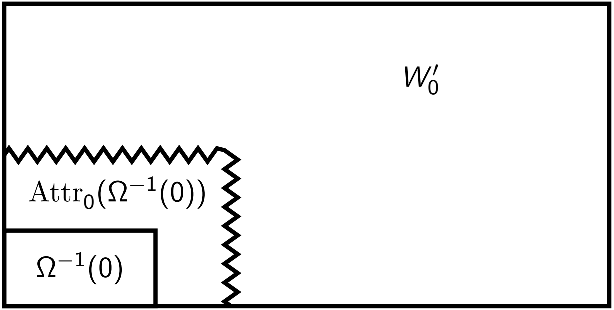
Proof Sketch

W_1' empty:



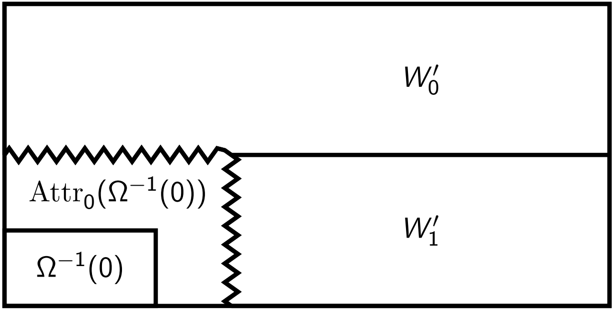
Proof Sketch

W_1' empty: Player 0 wins from everywhere.
Winning strategy: combine σ' and attractor strategy, play arbitrarily at $\Omega^{-1}(0)$.



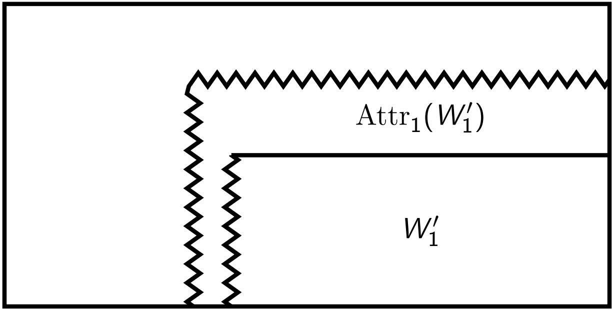
Proof Sketch

W_1' non-empty:



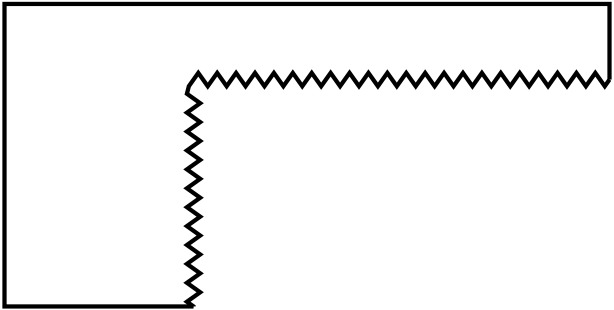
Proof Sketch

W_1' non-empty:



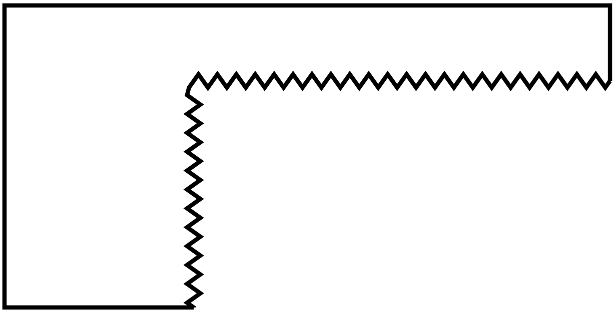
Proof Sketch

W_1' non-empty:



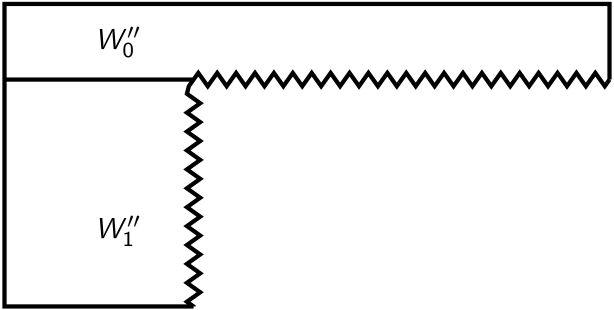
Proof Sketch

W_1' non-empty: Induction hypothesis applicable..



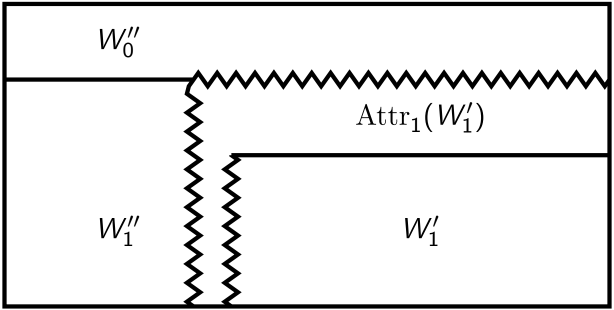
Proof Sketch

W_1' non-empty:.. yields winning regions W_i'' and positional strategies σ'', τ''.



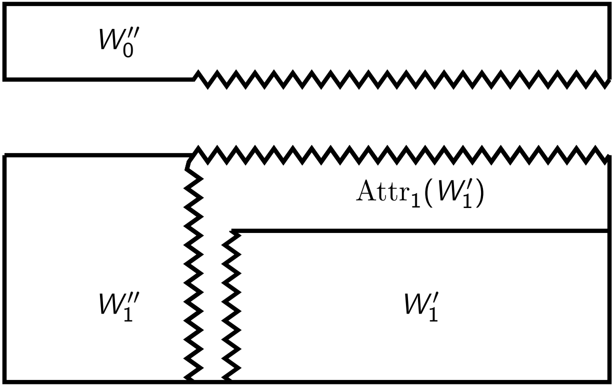
Proof Sketch

W_1' non-empty:.. yields winning regions W_i'' and positional strategies σ'', τ''.



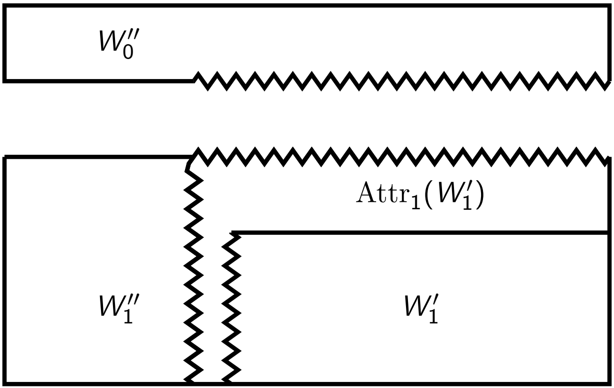
Proof Sketch

W_1' non-empty: Player 0 wins from W_0'' with σ''.



Proof Sketch

W_1' non-empty: Player 1 wins from $W_1'' \cup \mathrm{Attr}_1(W_1')$.
Winning strategy: combine τ', τ'', and attractor strategy.



Algorithms for Parity Games

- Determinacy proof yields recursive algorithm with exponential running time.
- Best deterministic algorithms: $\mathcal{O}(m \cdot n^{\frac{c}{3}})$.

Algorithms for Parity Games

- Determinacy proof yields recursive algorithm with exponential running time.
- Best deterministic algorithms: $\mathcal{O}(m \cdot n^{\frac{c}{3}})$.
- Intriguing complexity-theoretic status: in $\mathrm{NP} \cap \mathrm{Co\text{-}NP}$ (even in $\mathrm{UP} \cap \mathrm{Co\text{-}UP}$ and thus unlikely to be complete for NP or $\mathrm{Co\text{-}NP}$).

Algorithms for Parity Games

- Determinacy proof yields recursive algorithm with exponential running time.
- Best deterministic algorithms: $\mathcal{O}(m \cdot n^{\frac{c}{3}})$.
- Intriguing complexity-theoretic status: in $\mathrm{NP} \cap \mathrm{Co\text{-}NP}$ (even in $\mathrm{UP} \cap \mathrm{Co\text{-}UP}$ and thus unlikely to be complete for NP or $\mathrm{Co\text{-}NP}$).
- **Open problem:** is solving parity games in polynomial time?

Outline

Muller Games

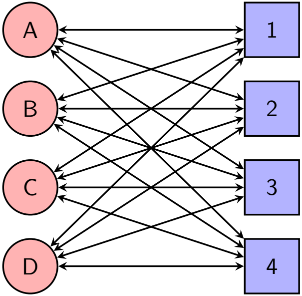
Muller games: for $\mathcal{F} \subseteq 2^V$ define

$$\textsc{Muller}(\mathcal{F}) = \{\, \rho \in V^\omega \mid \text{set of vertices seen infinitely often during } \rho \text{ is in } \mathcal{F} \,\}$$

Muller Games

Muller games: for $\mathcal{F} \subseteq 2^V$ define

$$\textsc{Muller}(\mathcal{F}) = \{\, \rho \in V^\omega \mid \text{set of vertices seen infinitely often during } \rho \text{ is in } \mathcal{F} \,\}$$



$F \in \mathcal{F}$

iff

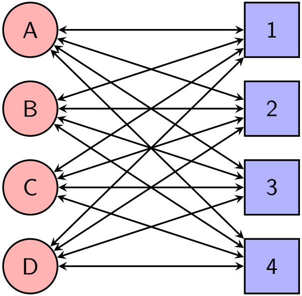
$|F \cap \{A, B, C, D\}| = \max(F \cap \{1, 2, 3, 4\})$

Muller Games

Muller games: for $\mathcal{F} \subseteq 2^V$ define

$$\textsc{muller}(\mathcal{F}) = \{\, \rho \in V^\omega \mid \text{set of vertices seen infinitely often during } \rho \text{ is in } \mathcal{F} \,\}$$

in general: DJW_n
here: DJW_4



$$F \in \mathcal{F}$$

iff

$$|F \cap \{A, B, C, D\}| = \max(F \cap \{1, 2, 3, 4\})$$

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C   A B C D #

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C (A B C D #)

4

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C   A B C D #

4

B   B A # C D

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C A B C D #

4

B B A # C D

2

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #
4	
B	B A # C D
2	
D	D B A C #

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C A B C D #

4

B B A # C D

2

D D B A C #

4

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4			
B	B A # C D		
2			
D	D B A C #		
4			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4		3	
B	B A # C D		
2			
D	D B A C #		
4			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4		3	
B	B A # C D	C	C A D B #
2			
D	D B A C #		
4			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4		3	
B	B A # C D	C	C A D B #
2		4	
D	D B A C #		
4			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4		3	
B	B A # C D	C	C A D B #
2		4	
D	D B A C #	C	C # A D B
4			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C
4		3	
B	B A # C D	C	C A D B #
2		4	
D	D B A C #	C	C # A D B
4		1	

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3			
B	B A # C D	C	C A D B #		
2		4			
D	D B A C #	C	C # A D B		
4		1			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #		
2		4			
D	D B A C #	C	C # A D B		
4		1			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #	C	C A # D B
2		4			
D	D B A C #	C	C # A D B		
4		1			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #	C	C A # D B
2		4		2	
D	D B A C #	C	C # A D B		
4		1			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #	C	C A # D B
2		4		2	
D	D B A C #	C	C # A D B	A	A C # D B
4		1			

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #	C	C A # D B
2		4		2	
D	D B A C #	C	C # A D B	A	A C # D B
4		1		2	

Latest Appearance Records

Need to *estimate* set of vertices in $\{A, B, C, D\}$ visited infinitely often during the play:

- Track order of last appearance of vertices in $\{A, B, C, D\}$

C	A B C D #	A	A D B # C	A	A C # D B
4		3		2	
B	B A # C D	C	C A D B #	C	C A # D B
2		4		2	
D	D B A C #	C	C # A D B	A	A C # D B
4		1		2	

- From some point onwards only vertices that are visited infinitely often are in front of #, and
- infinitely often exactly the set of vertices that are visited infinitely often is in front of #.

Muller Games

- Bookkeeping works in general (use permutations over V).
- Product of arena and LAR-structure can be turned into *equivalent* parity game from which finite-state strategies can be derived ("Muller games are reducible to parity games").

Muller Games

- Bookkeeping works in general (use permutations over V).
- Product of arena and LAR-structure can be turned into *equivalent* parity game from which finite-state strategies can be derived ("Muller games are reducible to parity games").

Theorem

Muller games are determined with finite-state strategies of size $n \cdot n!$.

Muller Games

- Bookkeeping works in general (use permutations over V).
- Product of arena and LAR-structure can be turned into *equivalent* parity game from which finite-state strategies can be derived ("Muller games are reducible to parity games").

Theorem

Muller games are determined with finite-state strategies of size $n \cdot n!$.

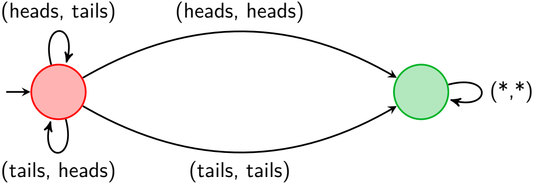
- Matching lower bounds via DJW_n games.
- Complexity depends on encoding of $\mathcal{F}$:
  - P, if $\mathcal{F}$ is given as list of sets.
  - $\mathrm{NP} \cap \mathrm{Co\text{-}NP}$, if $\mathcal{F}$ is encoded by a tree.
  - PSPACE-complete, if $\mathcal{F}$ is encoded by circuit or boolean formula (with variables V).

Outline

Concurrent Games

- Both players choose their moves simultaneously

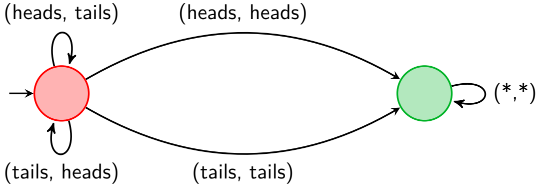
Matching pennies:



Concurrent Games

- Both players choose their moves simultaneously

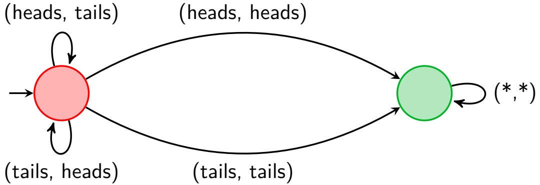
Matching pennies: randomized strategy winning with probability 1.



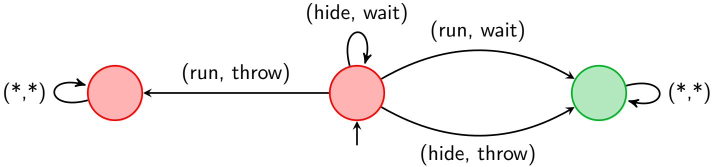
Concurrent Games

- Both players choose their moves simultaneously

Matching pennies: randomized strategy winning with probability 1.



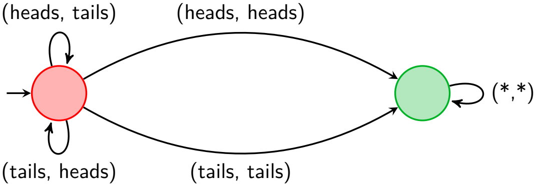
The "Snowball Game":



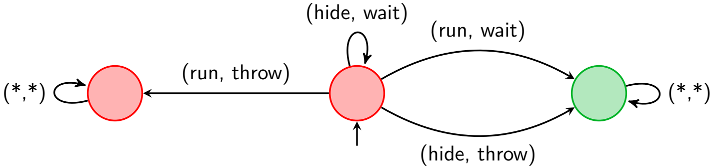
Concurrent Games

- Both players choose their moves simultaneously

Matching pennies: randomized strategy winning with probability 1.

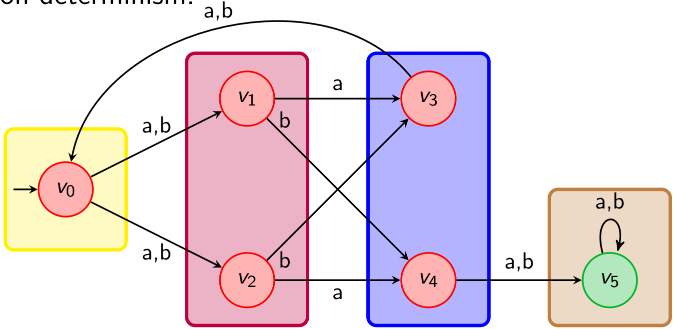


The "Snowball Game": for every ε, randomized strategy winning with probability $1 - \varepsilon$.



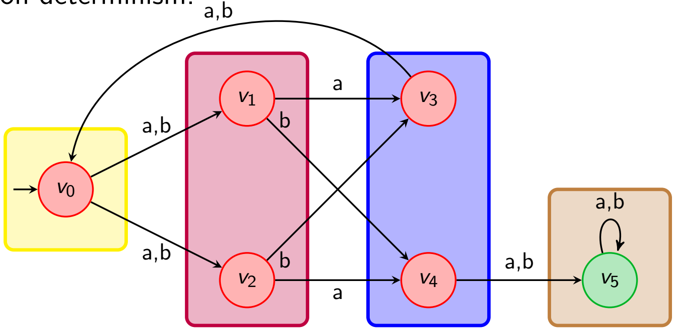
Games of Imperfect Information

- Players do not observe sequence of states, but sequence of non-unique observations (yellow, purple, blue, brown).
- Player 0 picks action (a or b), Player 1 resolves non-determinism.



Games of Imperfect Information

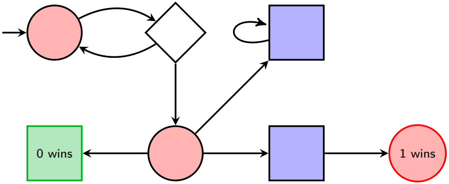
- Players do not observe sequence of states, but sequence of non-unique observations (yellow, purple, blue, brown).
- Player 0 picks action (a or b), Player 1 resolves non-determinism.



No winning strategy for Player 0: every fixed choice of actions to pick at ( )*( ) can be countered by going to v_1 or v_2.

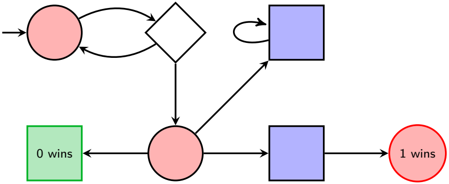
(Simple) Stochastic Games

- Enter a new player ($\Diamond$), it flips a coin to pick a successor.



(Simple) Stochastic Games

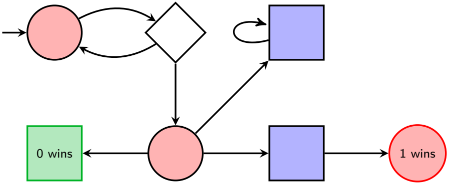
- Enter a new player ($\Diamond$), it flips a coin to pick a successor.



- No (sure) winning strategy...
- ...but one with probability 1.

(Simple) Stochastic Games

- Enter a new player ($\Diamond$), it flips a coin to pick a successor.



- No (sure) winning strategy...
- ...but one with probability 1.

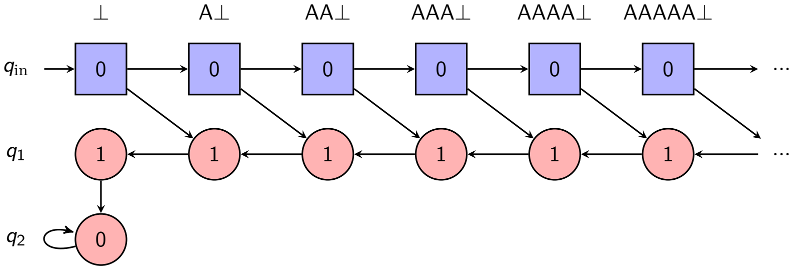
More formally: Value of the game

$$\max_{\sigma} \min_{\tau} p_{\sigma,\tau}$$

where $p_{\sigma,\tau}$ is the probability that Player 0 wins when using strategy σ and Player 1 uses strategy τ.

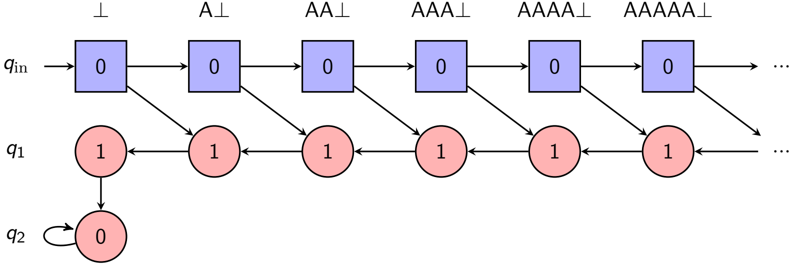
Pushdown Games

Use configuration graphs of pushdown machines as arena (in general infinite).



Pushdown Games

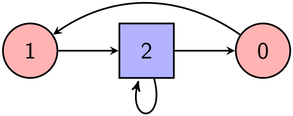
Use configuration graphs of pushdown machines as arena (in general infinite).



- Positional determinacy still holds, but positional strategies are infinite objects!
- Solution: winning strategies implemented by pushdown machines with output.

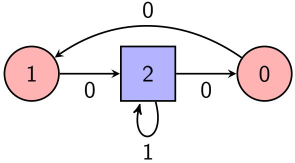
Quantitative Winning Conditions

- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long two “answer” 1 by 0.



Quantitative Winning Conditions

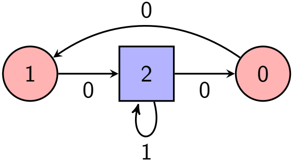
- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long two "answer" 1 by 0.



- Add edge-costs: Player 0 wins if there is a bound b and a position n such that every odd color after n is followed by a smaller even color with cost $\leq b$ in between

Quantitative Winning Conditions

- Parity game: Player 0 wins from everywhere, but it takes arbitrarily long two "answer" 1 by 0.



- Add edge-costs: Player 0 wins if there is a bound b and a position n such that every odd color after n is followed by a smaller even color with cost $\leq b$ in between
- Player 1 wins example from everywhere (stay at 2 longer and longer).

Many other variants

- Many more winning conditions.

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.
- Games on timed automata: uncountable arenas.

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.
- Games on timed automata: uncountable arenas.
- Play even longer: games of ordinal length.

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.
- Games on timed automata: uncountable arenas.
- Play even longer: games of ordinal length.
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.
- Games on timed automata: uncountable arenas.
- Play even longer: games of ordinal length.
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.
- More than two players: no longer zero-sum games. Requires whole new theory (equilibria).

Many other variants

- Many more winning conditions.
- Games on infinite arenas beyond pushdown graphs.
- Games on timed automata: uncountable arenas.
- Play even longer: games of ordinal length.
- Games with delay: Player 0 is allowed to skip some moves to obtain lookahead on Player 1's moves. Basic question: what kind of lookahead is necessary to win.
- More than two players: no longer zero-sum games. Requires whole new theory (equilibria).

And: any combination of extensions discussed above.

Literature

- Lecture notes "Infinite Games" (*hidden* in the Teaching section)

  `www.react.uni-saarland.de/teaching/infinite-games-13-14`
- Lectures in Game Theory for Computer Scientists. Krzysztof Apt and Erich Grädel (Eds.), Cambridge University Press, 2011.
- Automata, Logics, and Infinite Games. Erich Grädel, Wolfgang Thomas, and Thomas Wilke (Eds.), LNCS 2500, Springer-Verlag, 2002.