Martin Zimmermann
Aalborg University

Synthesis of Infinite-state Systems

UniVr/UniUd Summer School on Formal Methods for Cyber-Physical Systems, Udine, August 29, 2023

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. **I pick a number $m > 0$.**
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. **You pick a number $a \geq m$.**
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. **I pick a number $d \in \{2, 3, 5, 7\}$.**
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. **You win if $a \bmod d = 0$, otherwise I win.**

Let's Play a Game

1. **I pick a number $m > 0$.**
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. **You pick a number $a \geq m$.**
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. **I pick a number $d \in \{2, 3, 5, 7\}$.**
4. You win if $a \bmod d = 0$, otherwise I win.

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. **You win if $a \bmod d = 0$, otherwise I win.**

Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

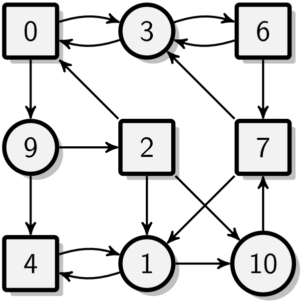
Let's Play a Game

1. I pick a number $m > 0$.
2. You pick a number $a \geq m$.
3. I pick a number $d \in \{2, 3, 5, 7\}$.
4. You win if $a \bmod d = 0$, otherwise I win.

A **winning strategy** for you:

Given my choice $m > 0$, pick $a = 210m$. This is winning, as we have $210m \bmod d = 0$ for every $d \in \{2, 3, 5, 7\}$.

Reminder: Parity Games



Player 0:

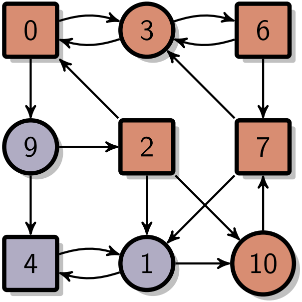
- Protagonist,
- round vertices,
- wins if maximal color seen infinitely often is even (has parity 0).

Player 1:

- Antagonist,
- square vertices,
- wins if maximal color seen infinitely often is odd (has parity 1).

A parity game (where we identify vertex names and colors)

Reminder: Parity Games



Player 0:

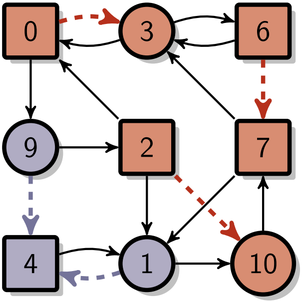
- Protagonist,
- round vertices,
- wins if maximal color seen infinitely often is even (has parity 0).

Player 1:

- Antagonist,
- square vertices,
- wins if maximal color seen infinitely often is odd (has parity 1).

A parity game (where we identify vertex names and colors), its winning regions (blue for Player 0, red for Player 1)

Reminder: Parity Games



Player 0:

- Protagonist,
- round vertices,
- wins if maximal color seen infinitely often is even (has parity 0).

Player 1:

- Antagonist,
- square vertices,
- wins if maximal color seen infinitely often is odd (has parity 1).

A parity game (where we identify vertex names and colors), its winning regions (blue for Player 0, red for Player 1), and (positional) winning strategies for both players (on their winning regions).

Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega \colon V \to \mathbb{N}$ of the vertices.

Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega \colon V \to \mathbb{N}$ of the vertices.

- A **play**: an infinite sequence $v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_n, v_{n+1}) \in E$ for all n.
- $v_0 v_1 v_2 \cdots$ is **winning** for Player 0 (Player 1): parity of the maximal color seen infinitely often is even (odd).

Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega \colon V \to \mathbb{N}$ of the vertices.

- A **play**: an infinite sequence $v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_n, v_{n+1}) \in E$ for all n.
- $v_0 v_1 v_2 \cdots$ is **winning** for Player 0 (Player 1): parity of the maximal color seen infinitely often is even (odd).
- **Strategy** for Player i: $\sigma \colon V^* V_i \to V$ such that $(v, \sigma(wv)) \in E$ for all $w \in V^*$ and $v \in V_i$.
- σ is **positional**: $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$.

Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega\colon V \to \mathbb{N}$ of the vertices.

- A **play**: an infinite sequence $v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_n, v_{n+1}) \in E$ for all n.
- $v_0 v_1 v_2 \cdots$ is **winning** for Player 0 (Player 1): parity of the maximal color seen infinitely often is even (odd).
- **Strategy** for Player i: $\sigma\colon V^* V_i \to V$ such that $(v, \sigma(wv)) \in E$ for all $w \in V^*$ and $v \in V_i$.
- σ is **positional**: $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$.
- $v_0 v_1 v_2 \cdots$ **consistent** with σ: $v_{n+1} = \sigma(v_0 \cdots v_n)$ for all n with $v_n \in V_i$.

Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega \colon V \to \mathbb{N}$ of the vertices.

- A **play**: an infinite sequence $v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_n, v_{n+1}) \in E$ for all n.
- $v_0 v_1 v_2 \cdots$ is **winning** for Player 0 (Player 1): parity of the maximal color seen infinitely often is even (odd).
- **Strategy** for Player i: $\sigma \colon V^* V_i \to V$ such that $(v, \sigma(wv)) \in E$ for all $w \in V^*$ and $v \in V_i$.
- σ is **positional**: $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$.
- $v_0 v_1 v_2 \cdots$ **consistent** with σ: $v_{n+1} = \sigma(v_0 \cdots v_n)$ for all n with $v_n \in V_i$.
- σ is **winning from $v \in V$**: all plays starting in v and consistent with σ are winning for Player i.

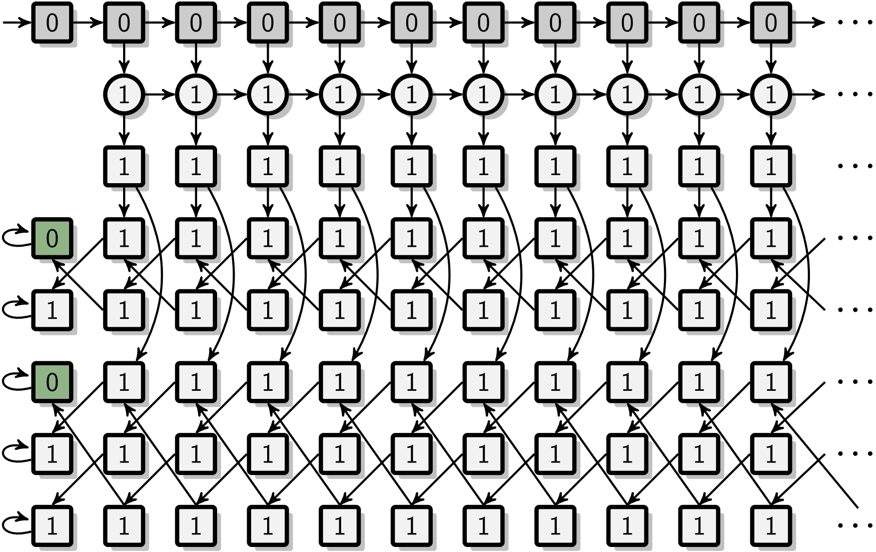
Parity Games on One Slide

A **parity game** (V, V_0, V_1, E, Ω) consists of

- a set V of vertices partitioned into
- the sets V_0 and V_1 of vertices of Player 0 and Player 1,
- a set $E \subseteq V \times V$ of directed edges (we assume that every vertex has at least one outgoing edge), and
- a coloring $\Omega\colon V \to \mathbb{N}$ of the vertices.

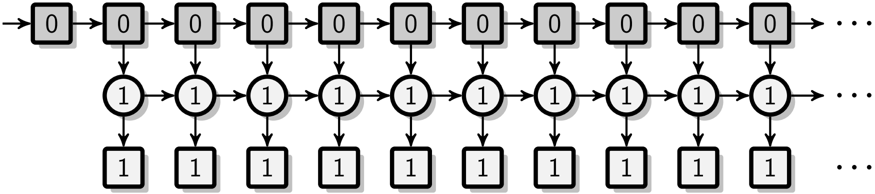
- A **play**: an infinite sequence $v_0 v_1 v_2 \cdots \in V^\omega$ such that $(v_n, v_{n+1}) \in E$ for all n.
- $v_0 v_1 v_2 \cdots$ is **winning** for Player 0 (Player 1): parity of the maximal color seen infinitely often is even (odd).
- **Strategy** for Player i: $\sigma\colon V^* V_i \to V$ such that $(v, \sigma(wv)) \in E$ for all $w \in V^*$ and $v \in V_i$.
- σ is **positional**: $\sigma(wv) = \sigma(v)$ for all $w \in V^*$ and $v \in V_i$.
- $v_0 v_1 v_2 \cdots$ **consistent** with σ: $v_{n+1} = \sigma(v_0 \cdots v_n)$ for all n with $v_n \in V_i$.
- σ is **winning from $v \in V$**: all plays starting in v and consistent with σ are winning for Player i.

Modeling the Game

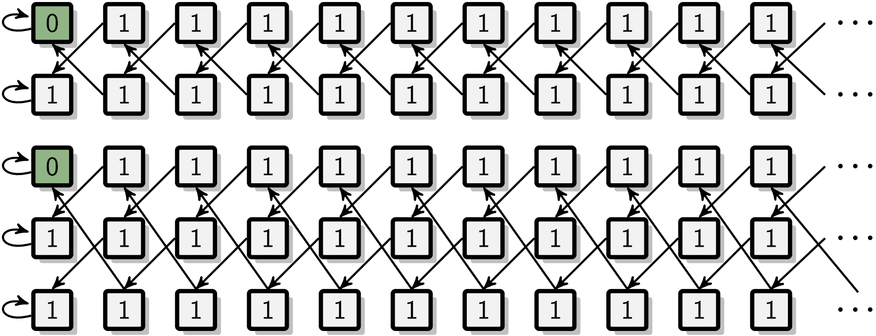


Parity condition: The circle player (i.e., you) wins if either only gray vertices are visited or a green vertex is visited.

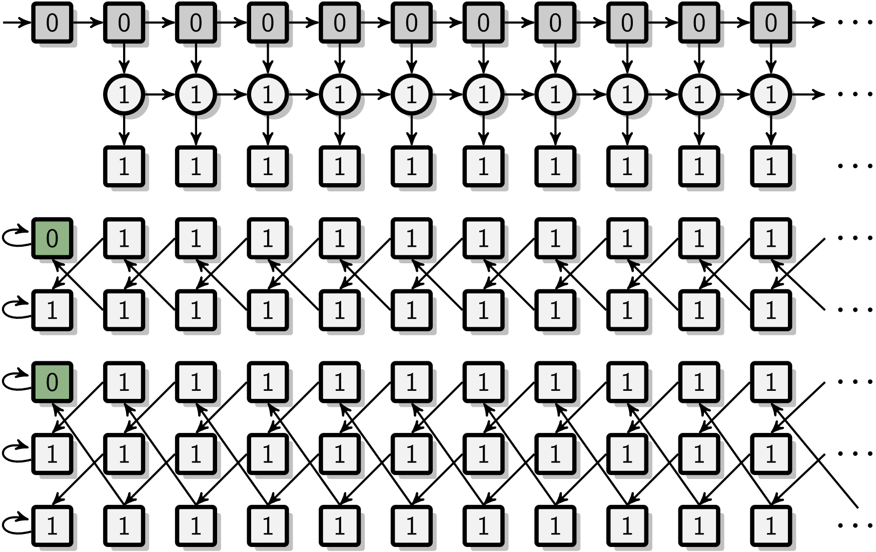
Modeling the Game



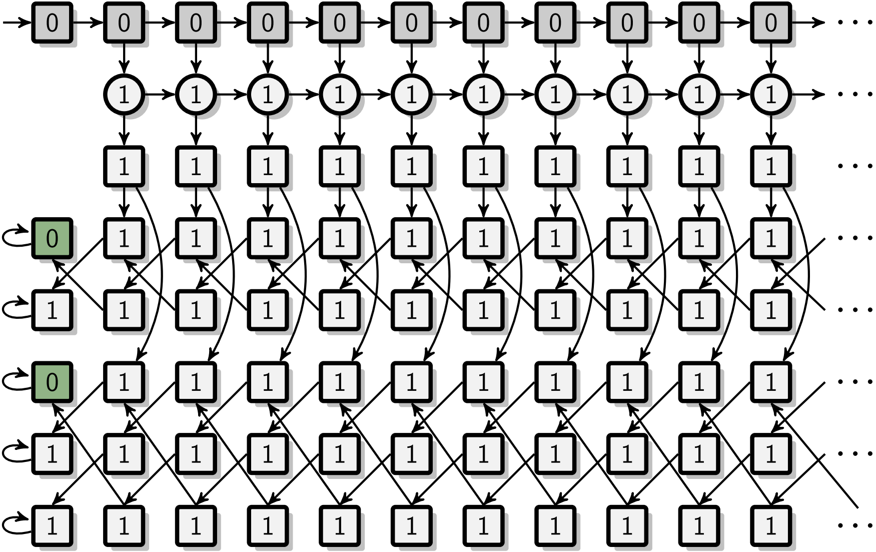
Modeling the Game



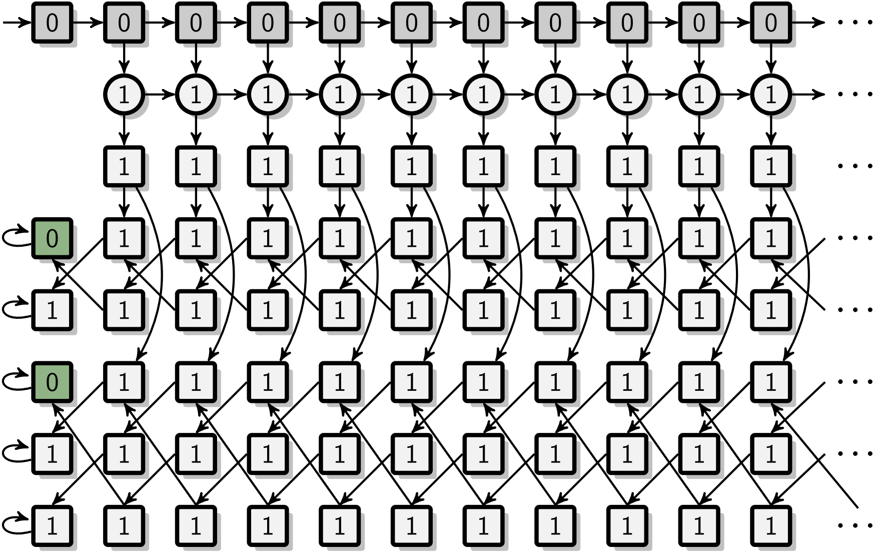
Modeling the Game



Modeling the Game



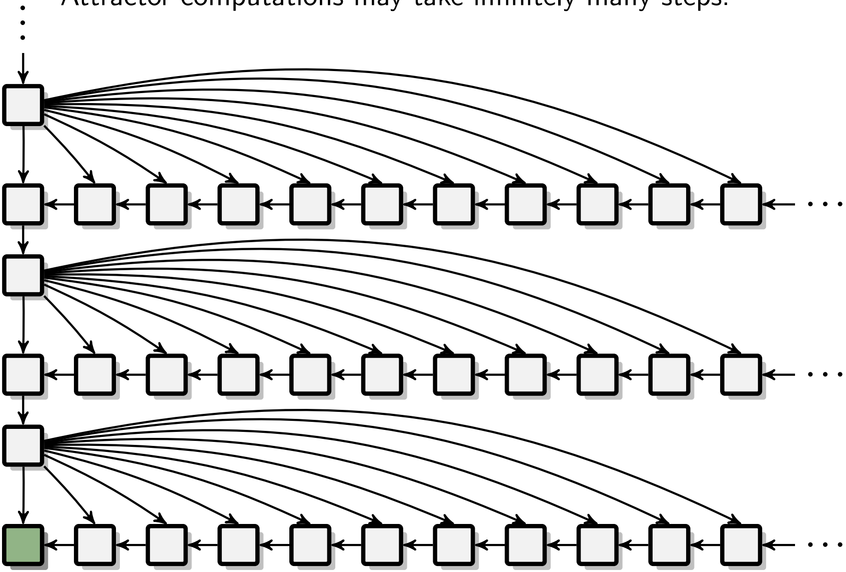
Modeling the Game



Parity condition: The circle player (i.e., you) wins if either only gray vertices are visited or a green vertex is visited.

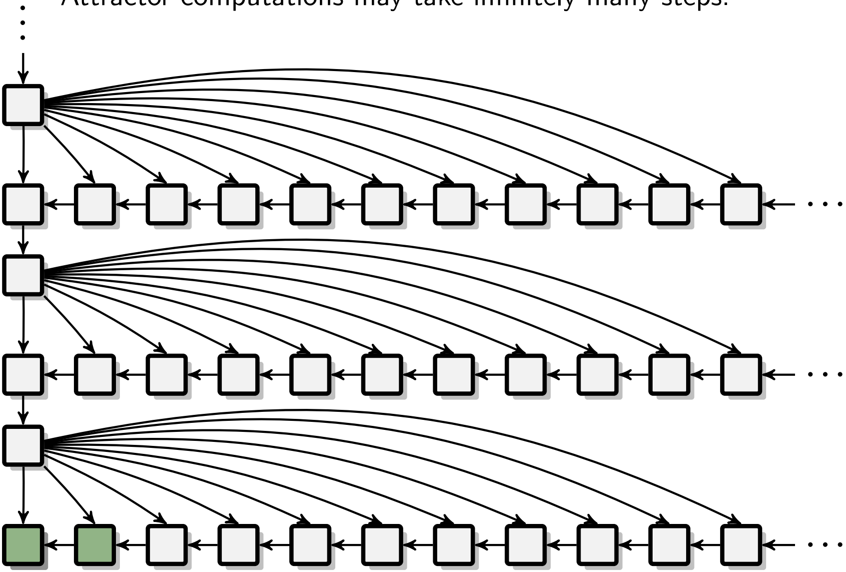
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



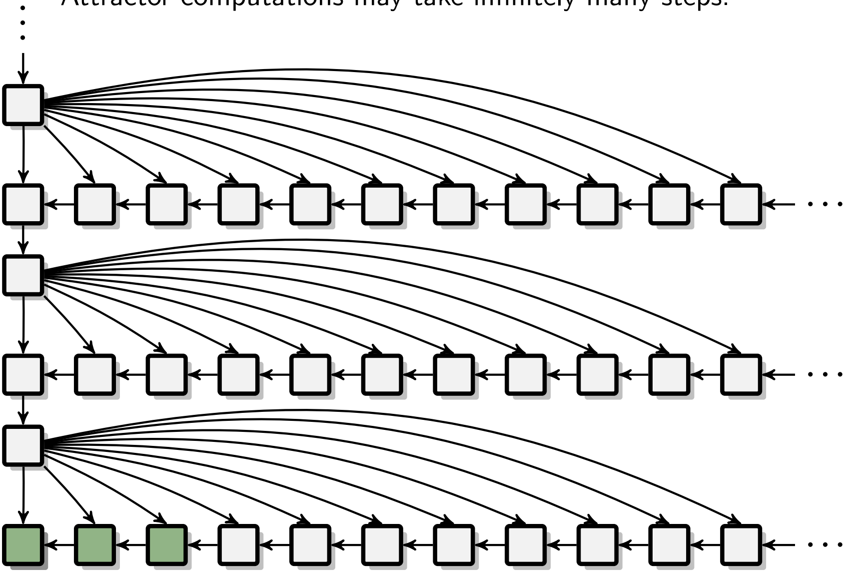
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



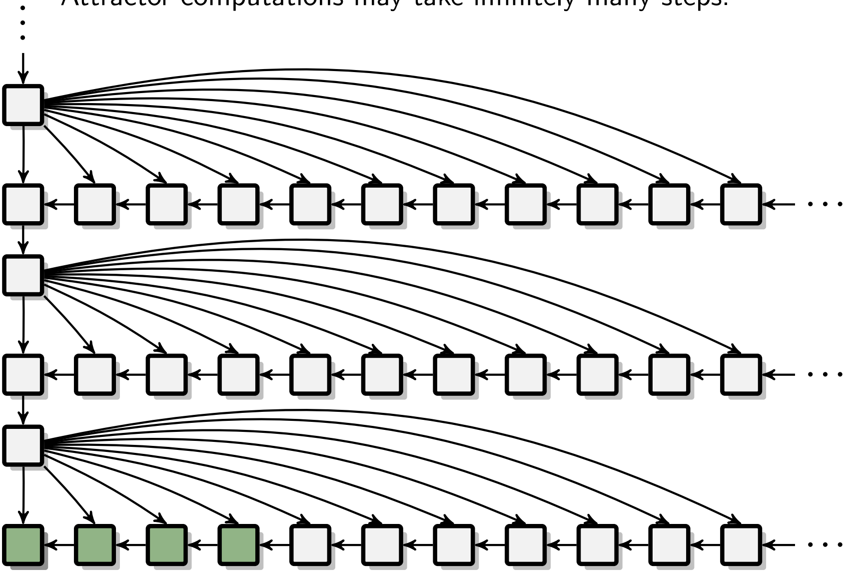
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



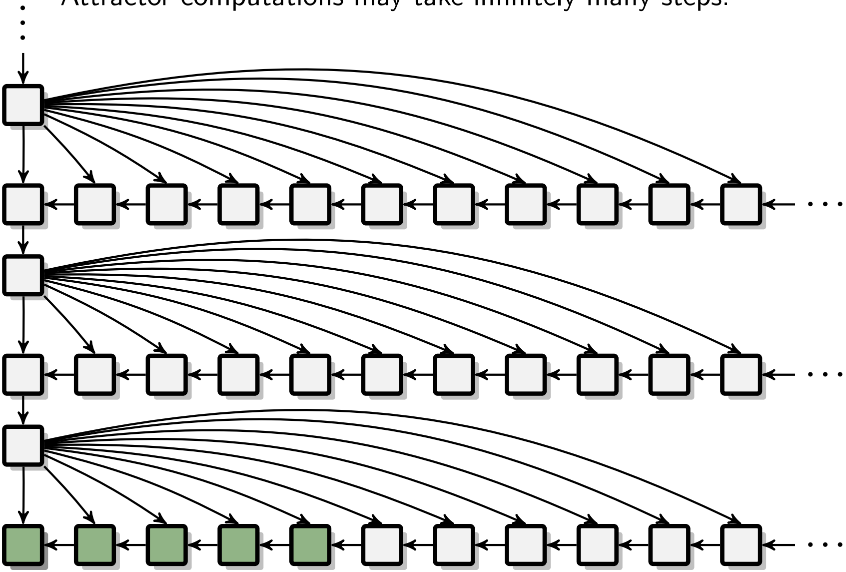
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



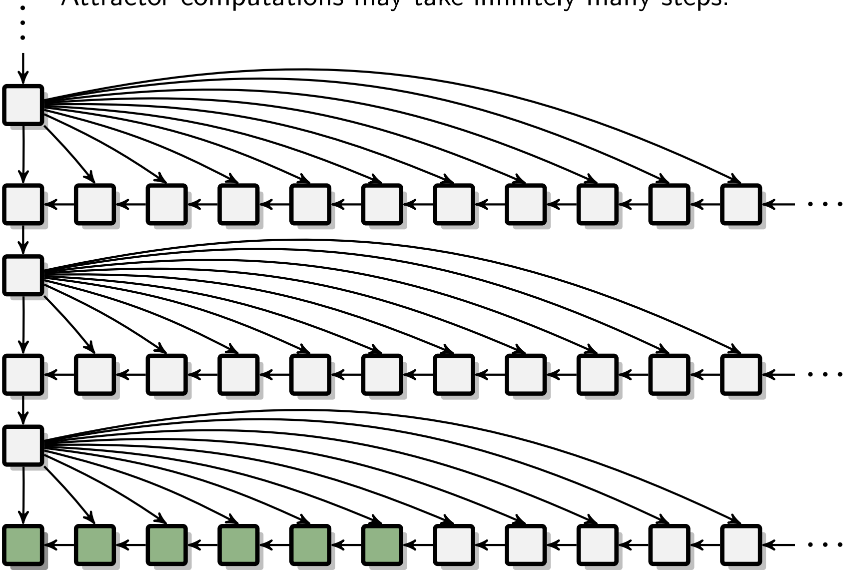
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



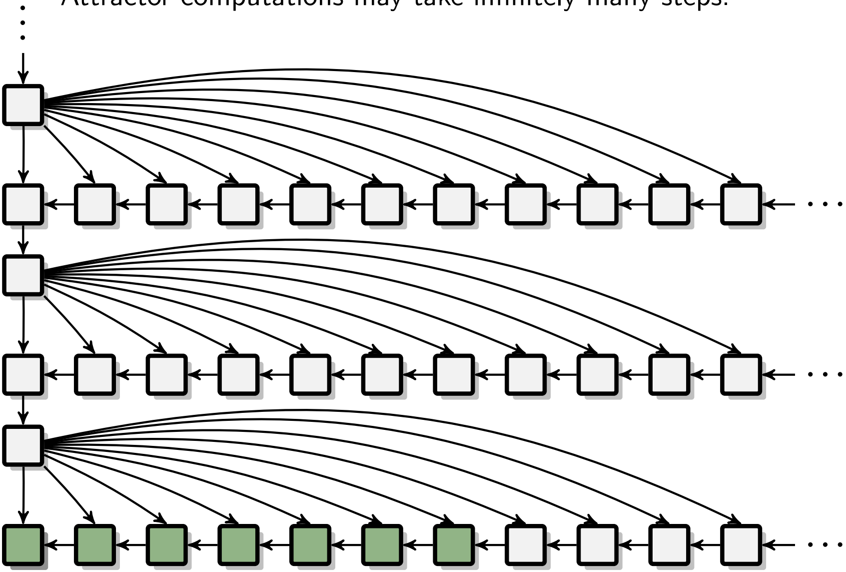
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



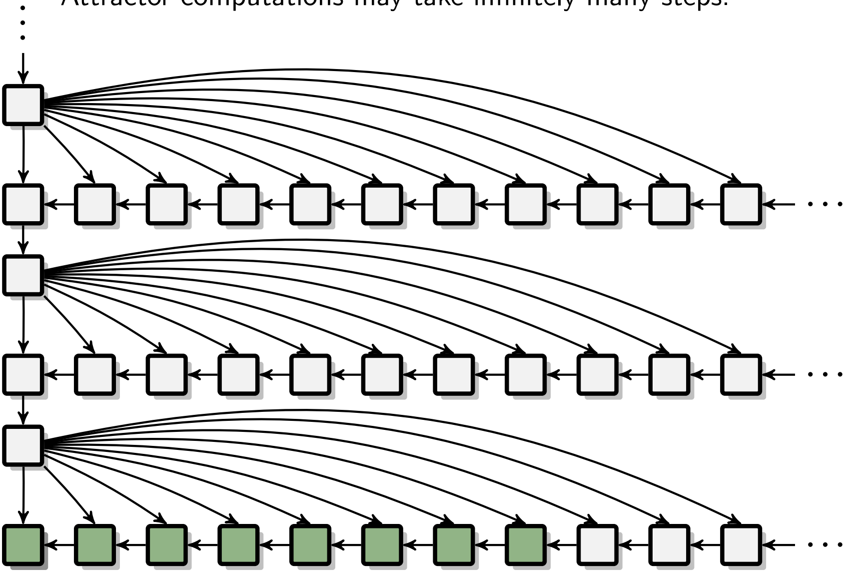
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



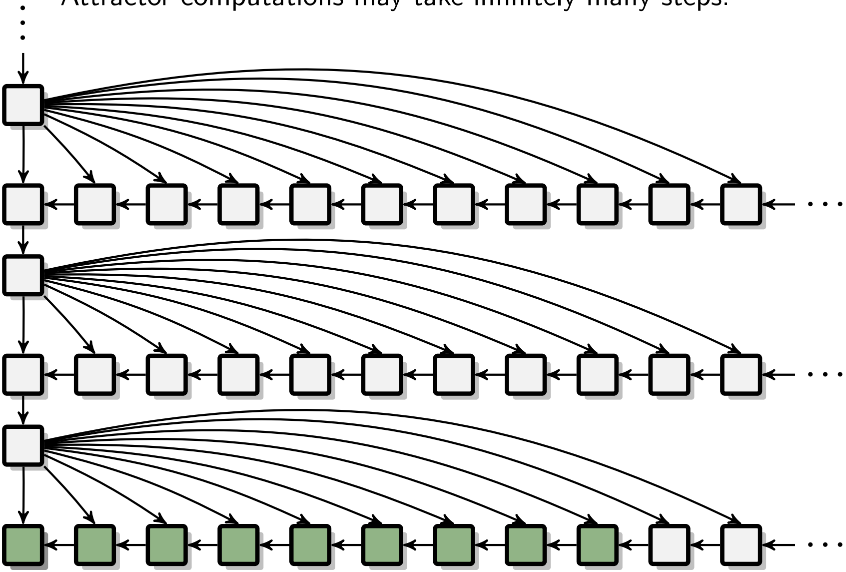
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



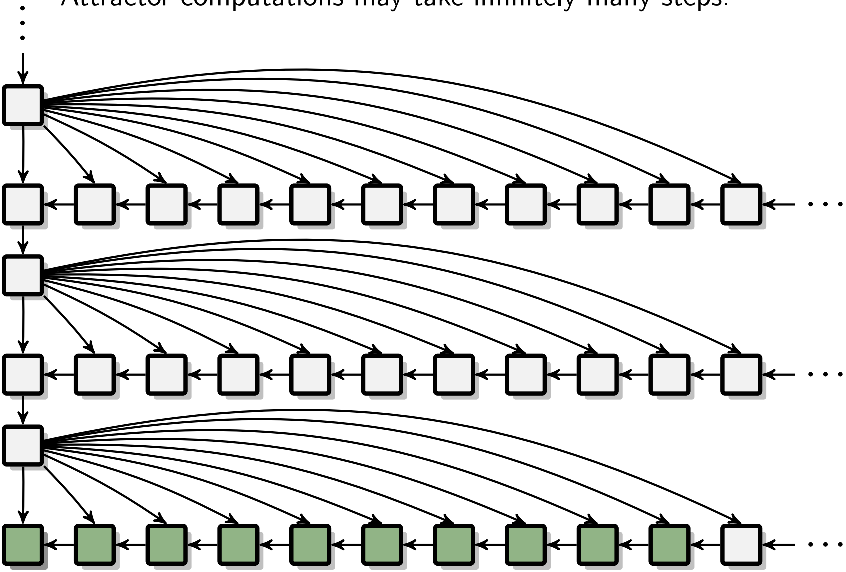
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



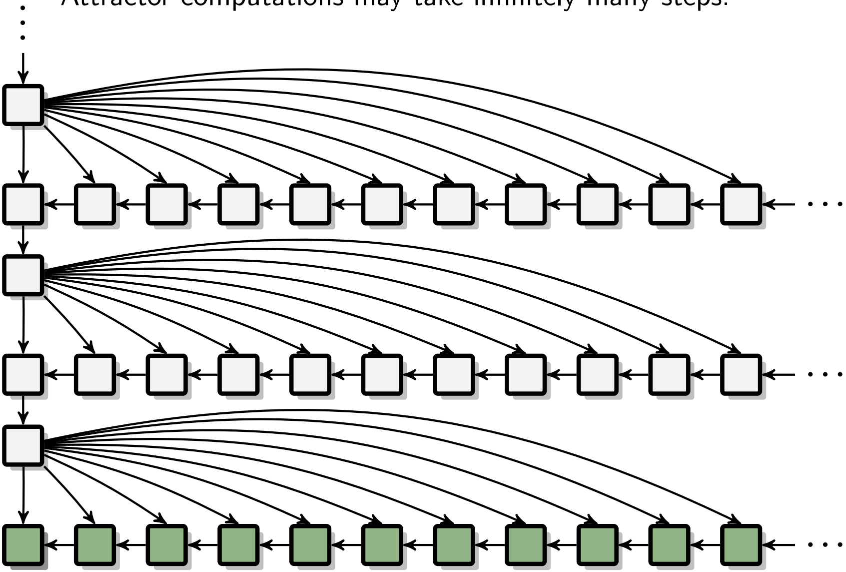
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



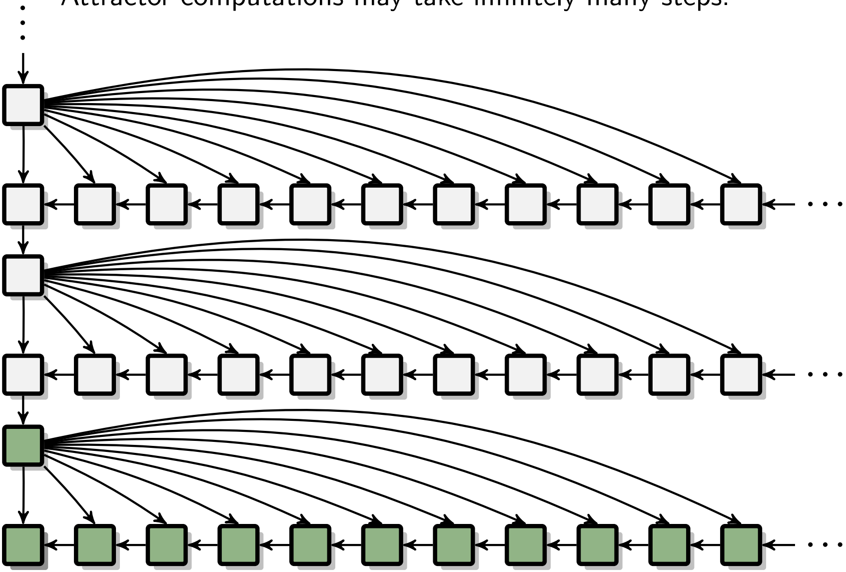
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



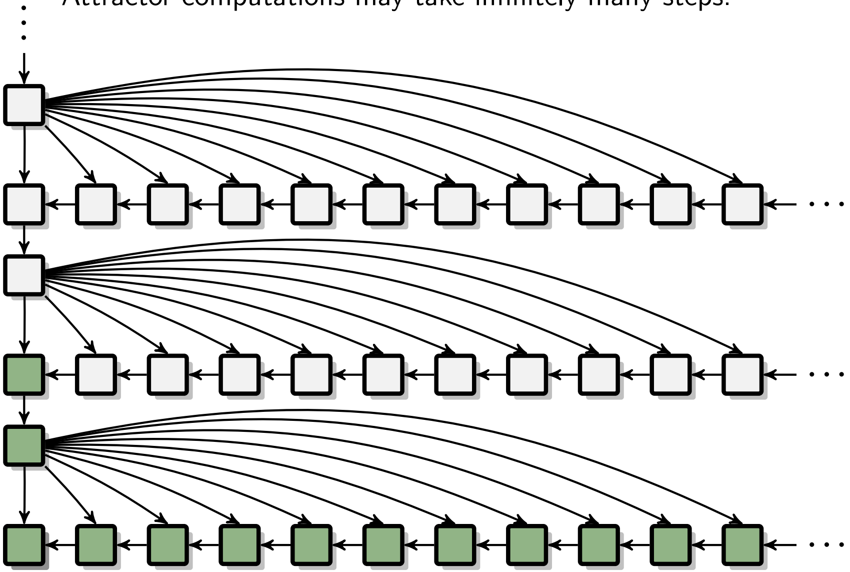
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



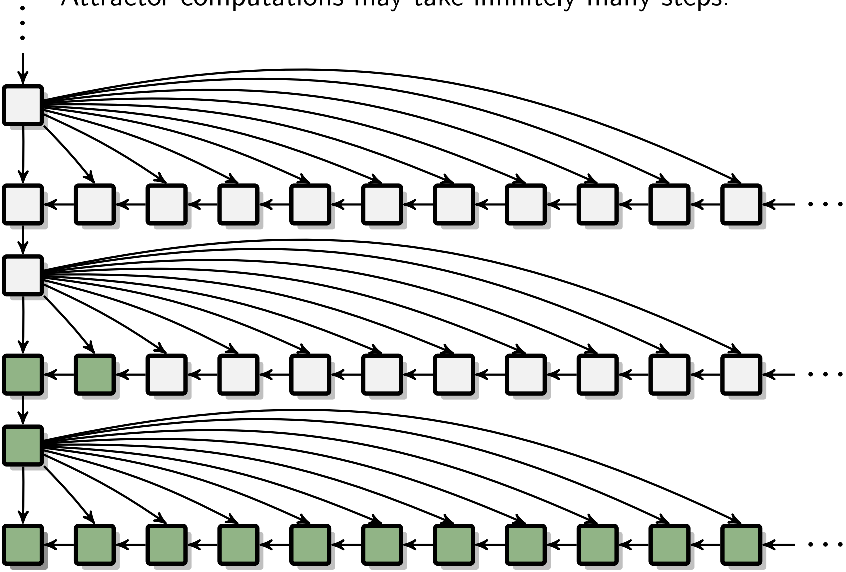
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



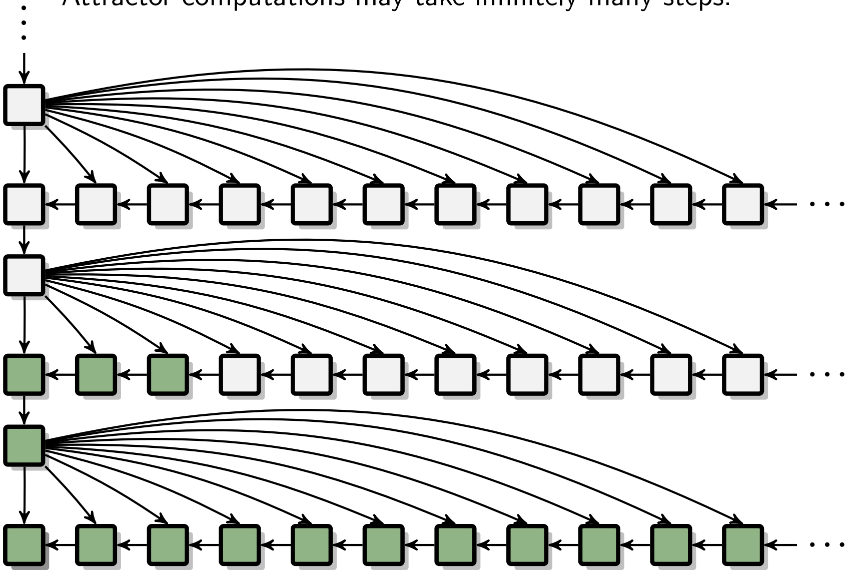
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



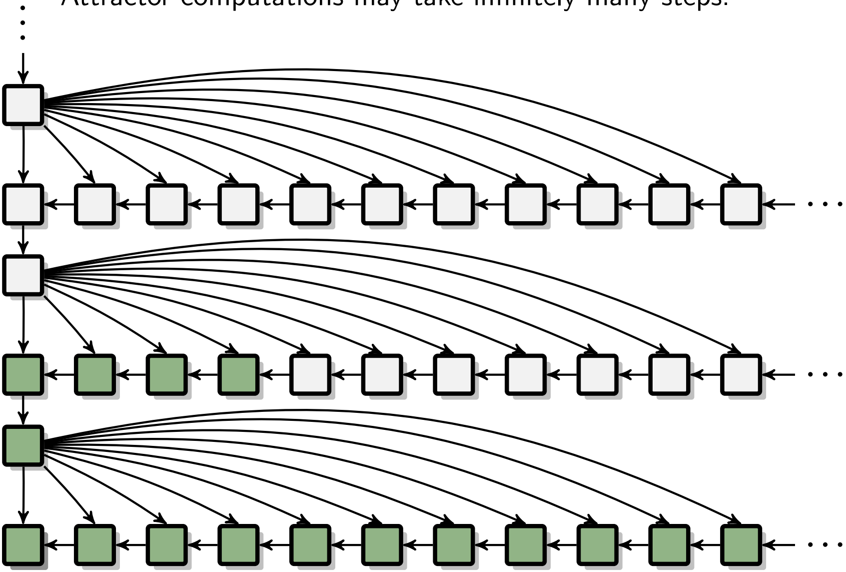
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



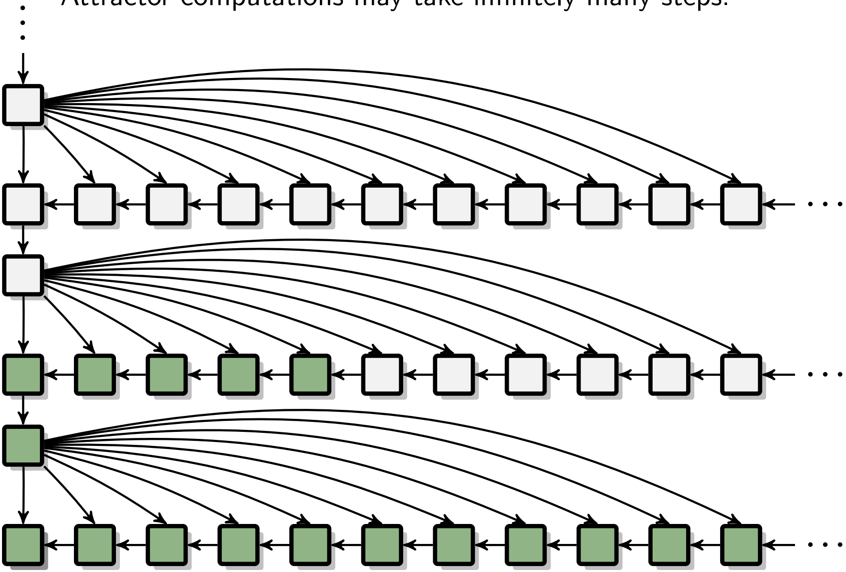
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



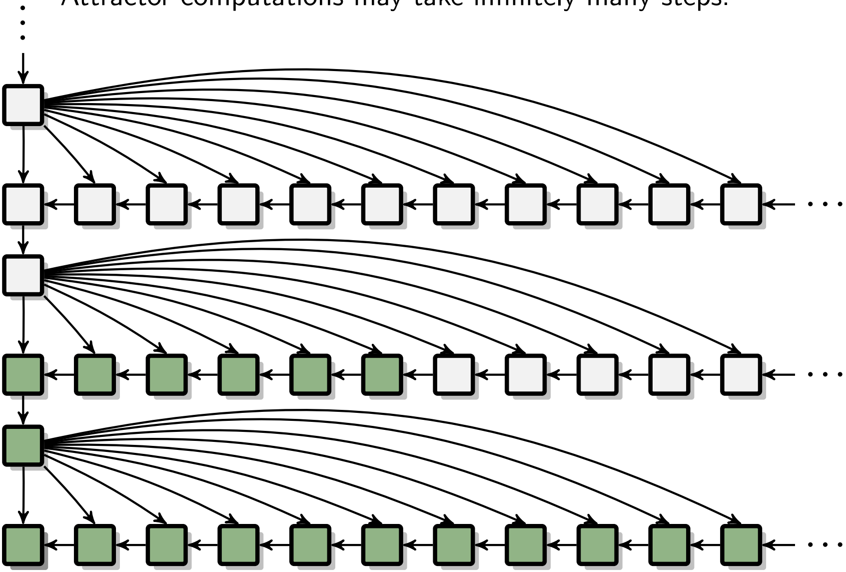
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



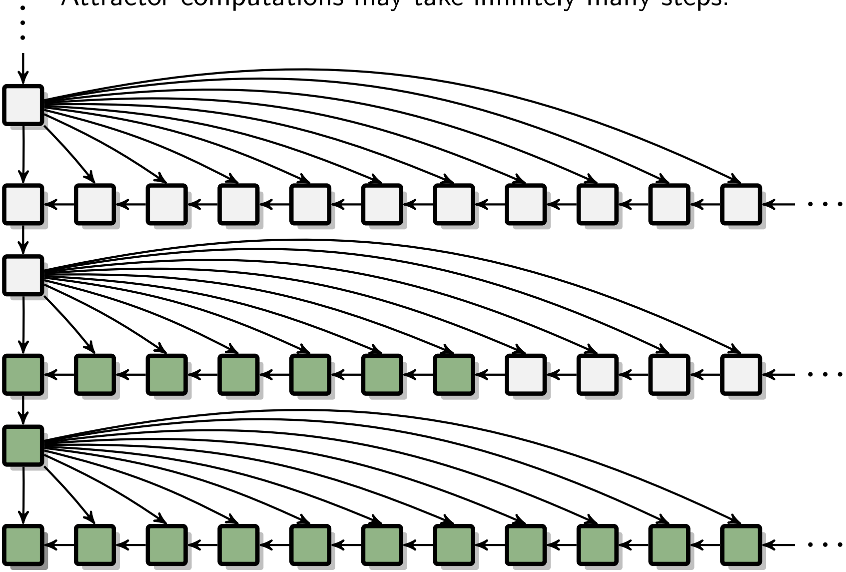
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



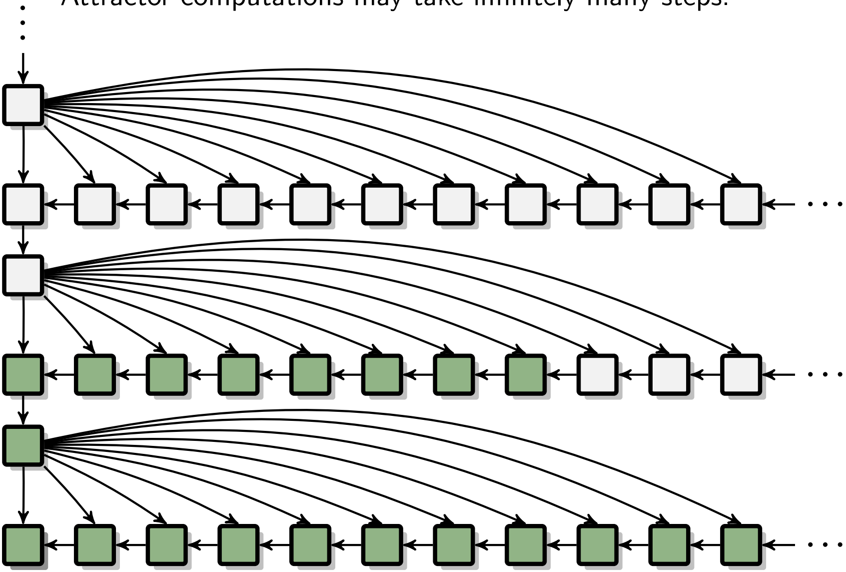
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



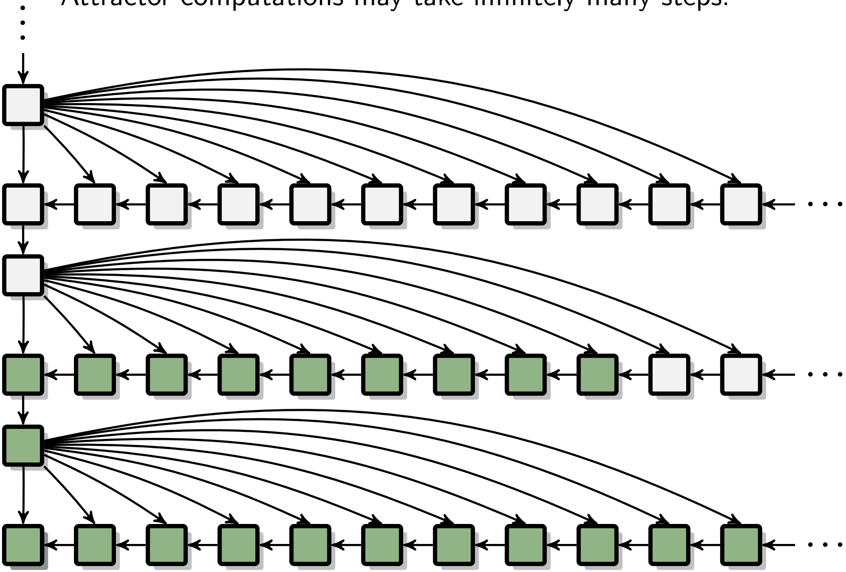
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



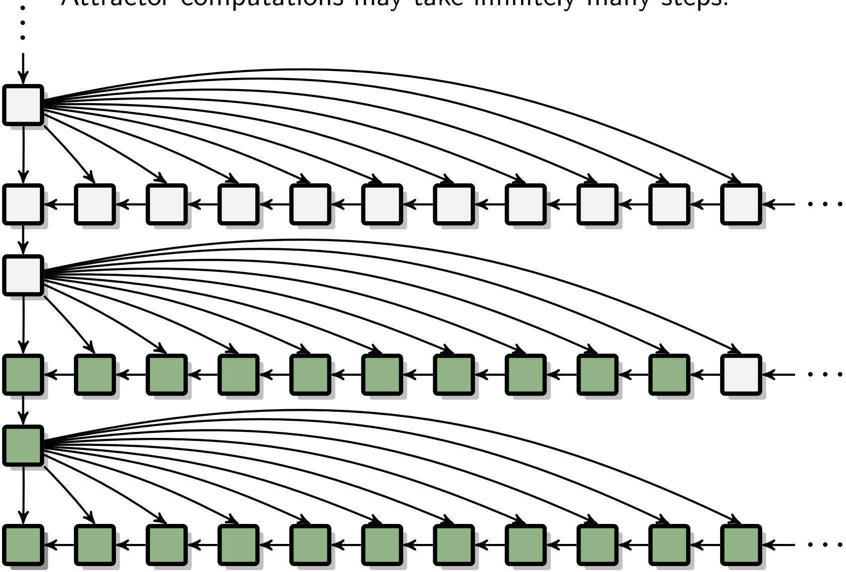
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



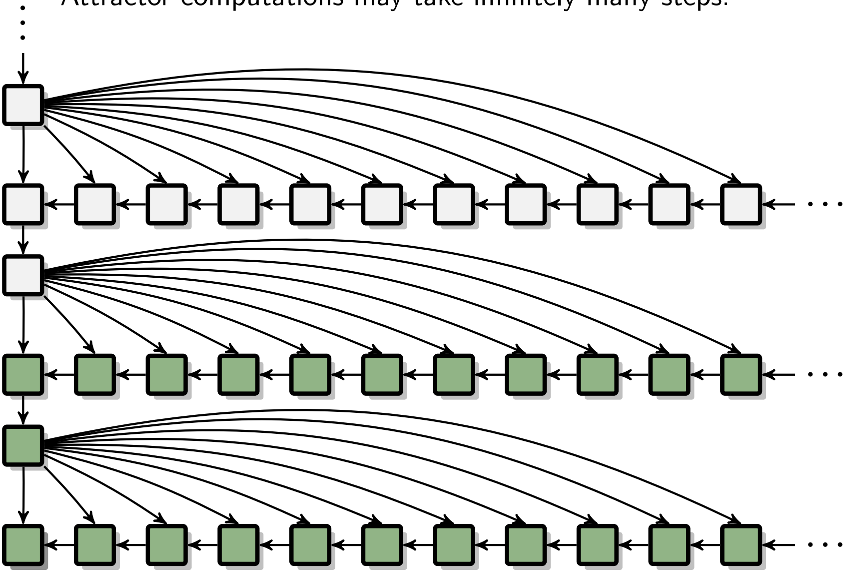
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



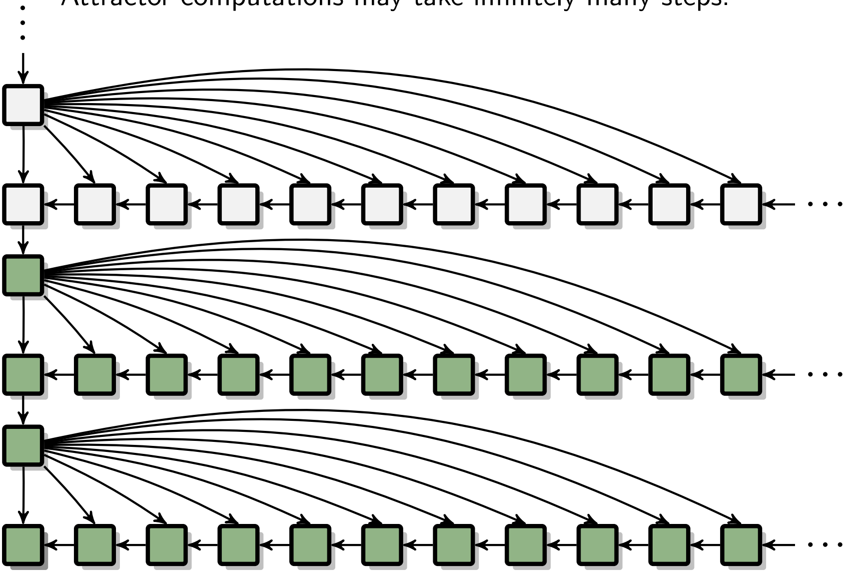
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



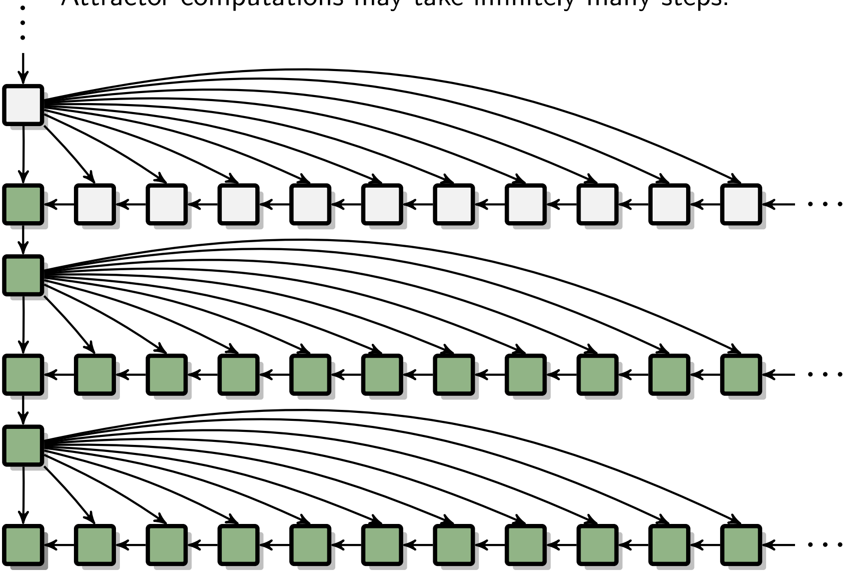
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



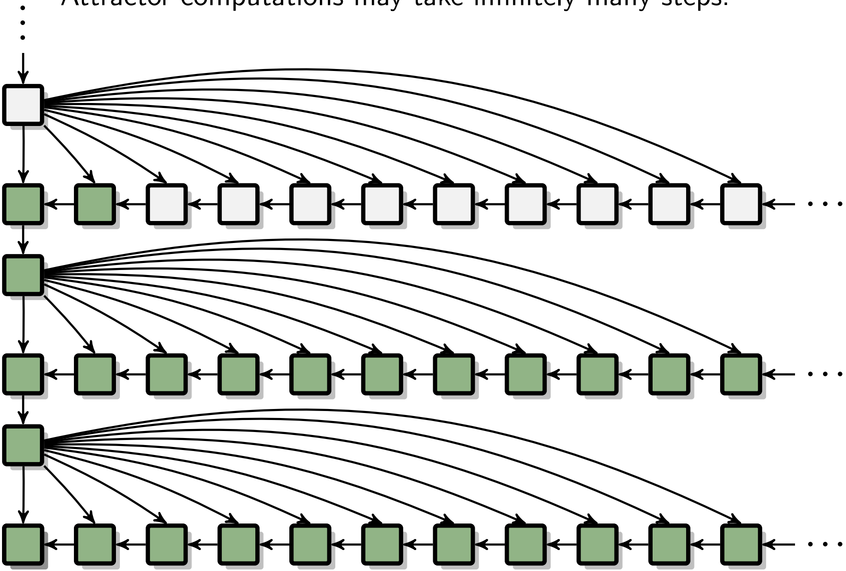
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



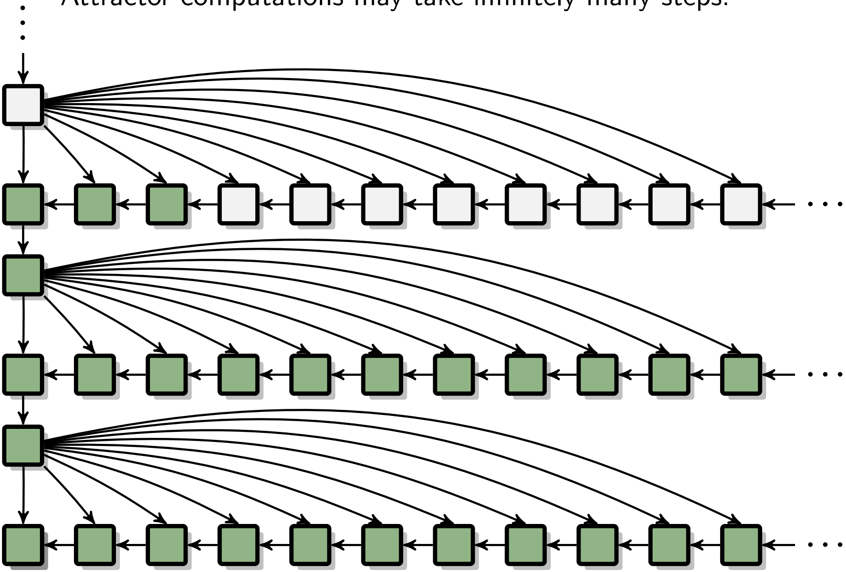
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



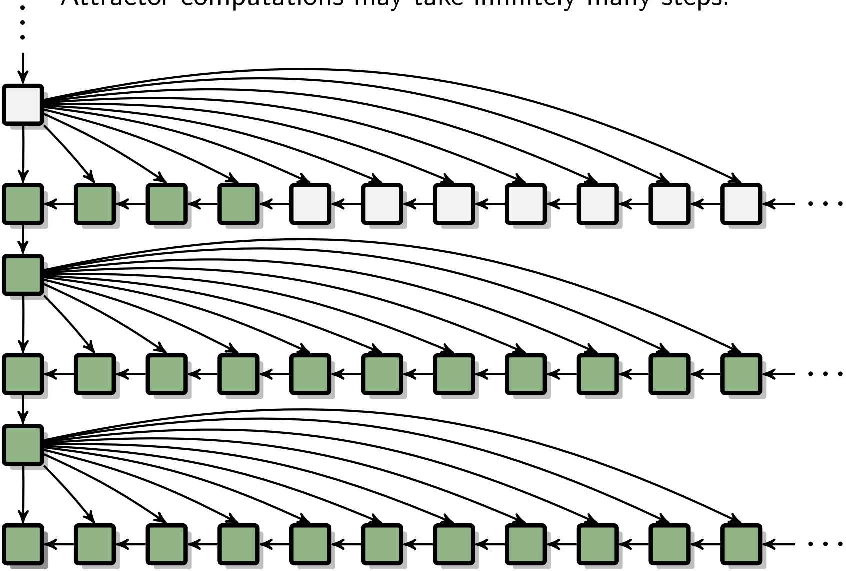
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



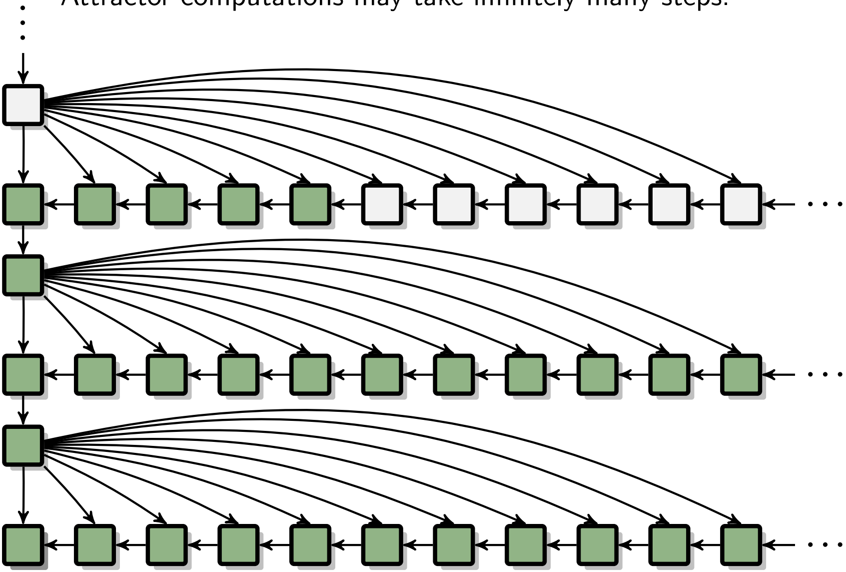
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



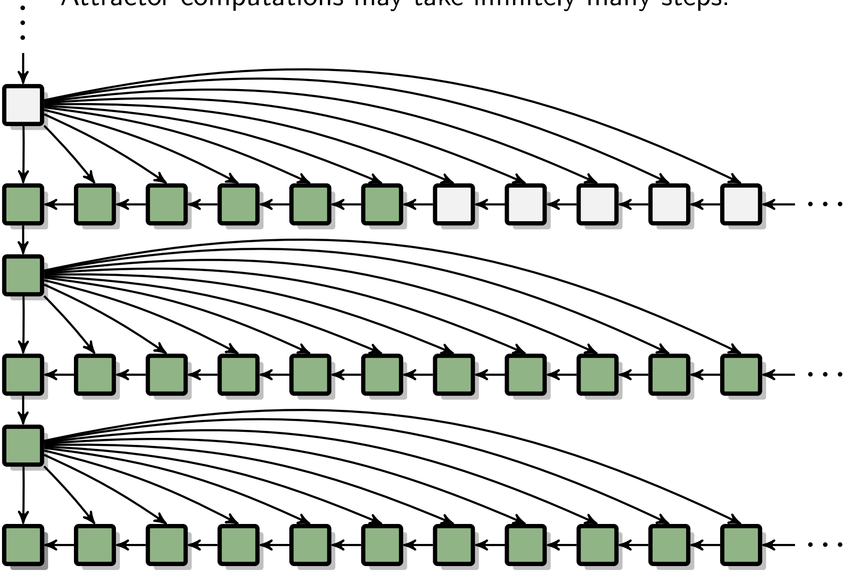
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



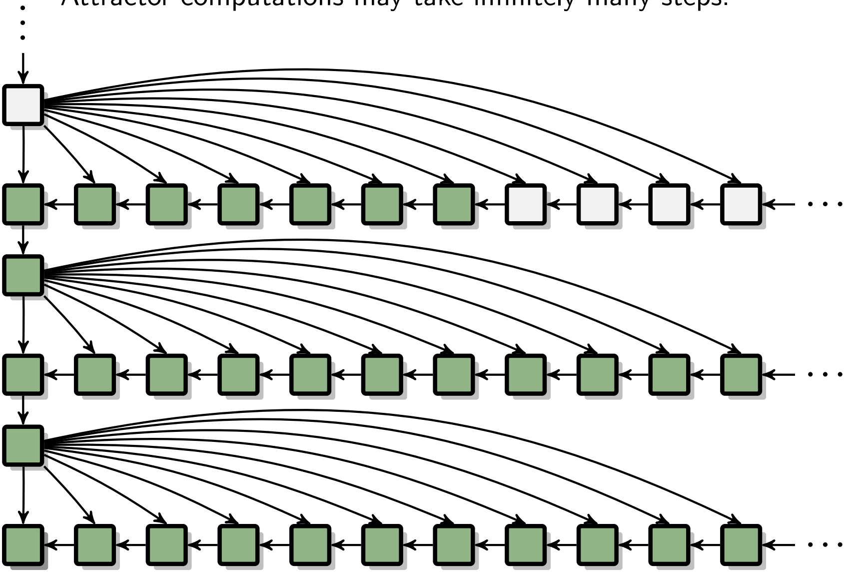
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



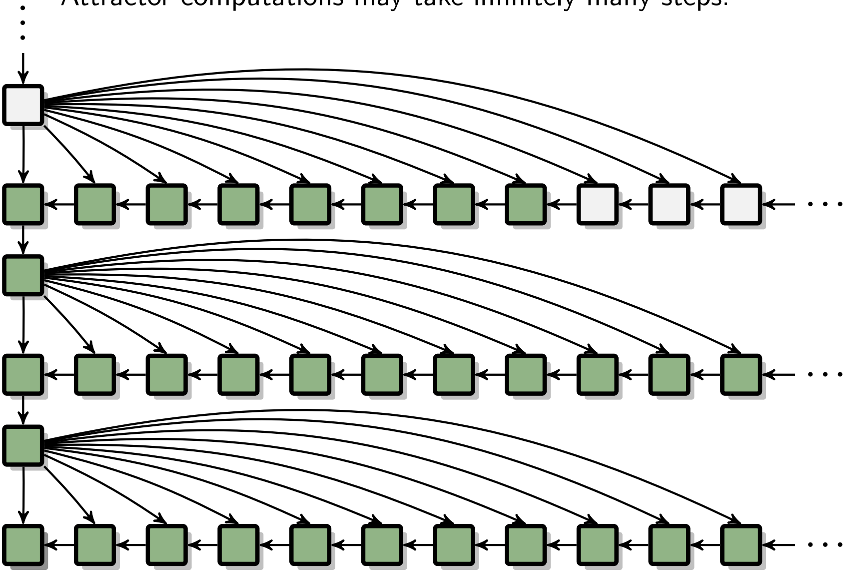
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



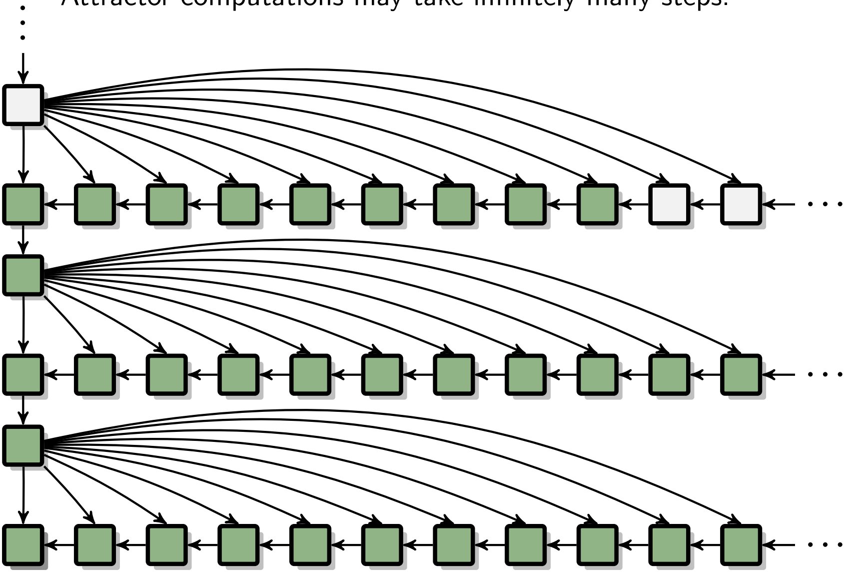
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



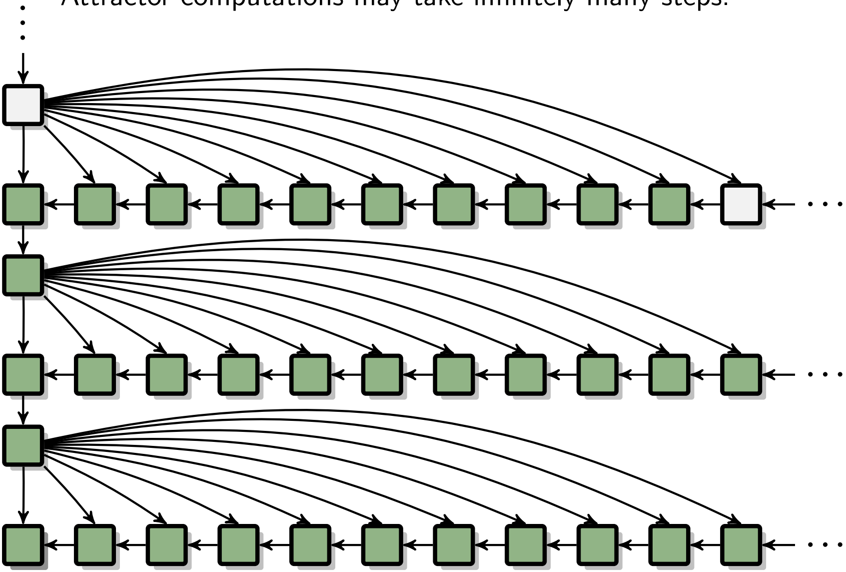
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



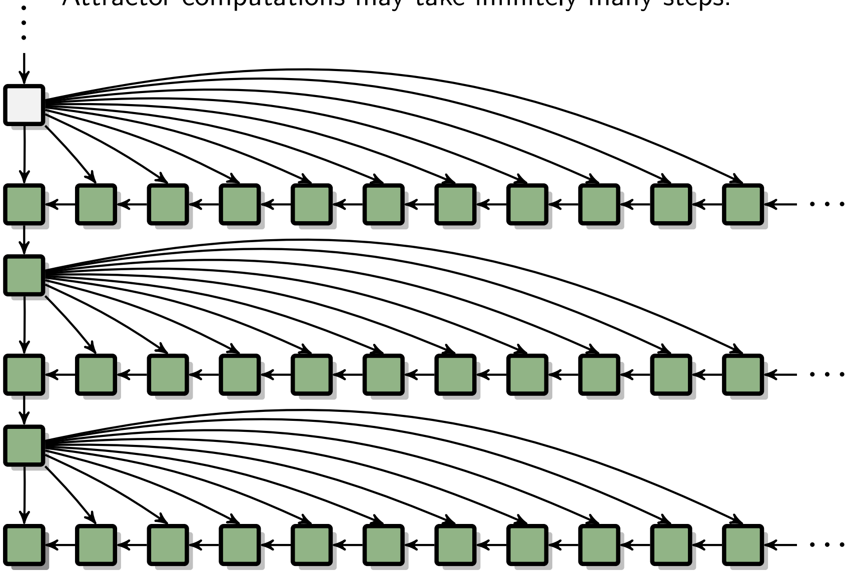
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.



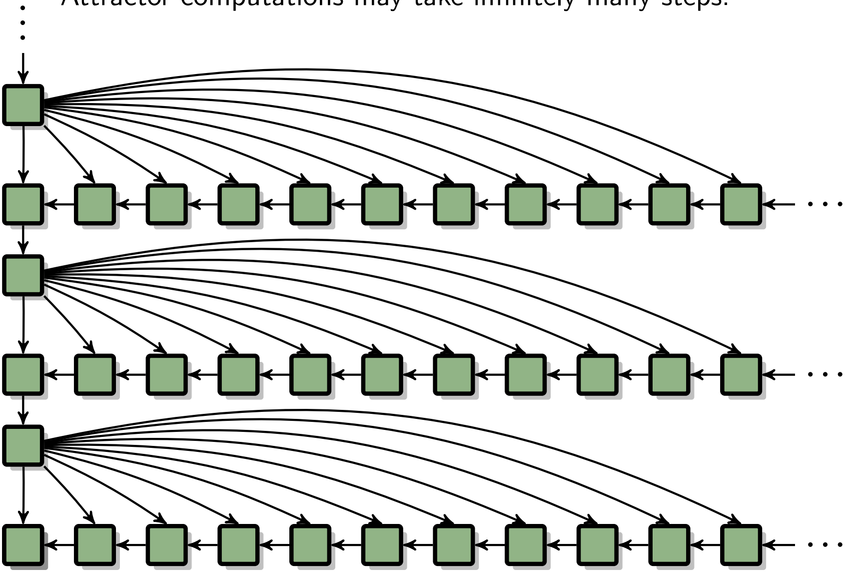
Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.

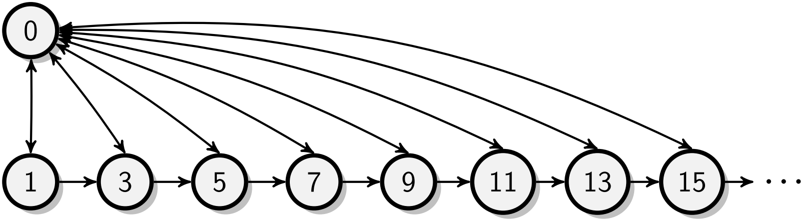


Infinity Makes Things Interesting

Attractor computations may take infinitely many steps.

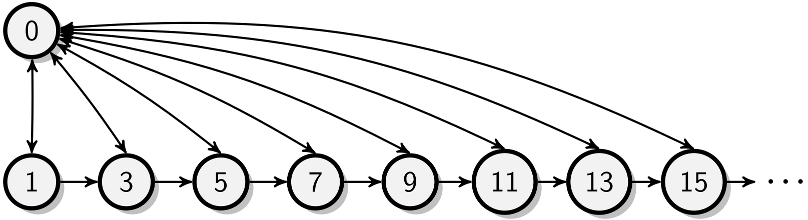


More Interesting Things



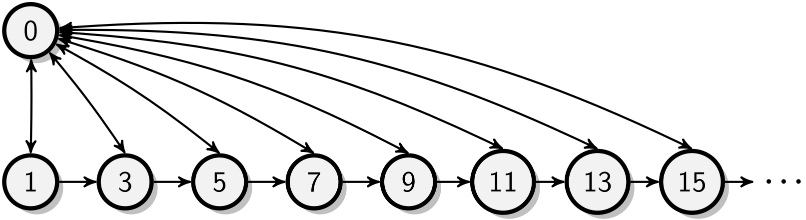
More Interesting Things

- Set of colors seen infinitely often during a play may be empty.



More Interesting Things

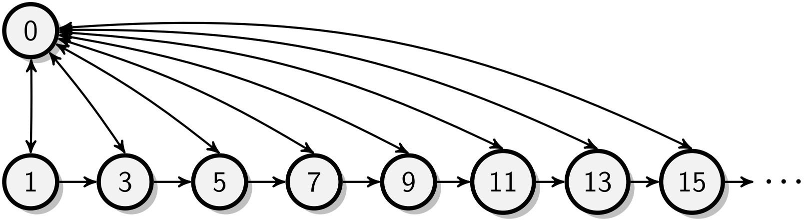
- Set of colors seen infinitely often during a play may be empty.
- Parity games might require infinite-memory strategies.



More Interesting Things

- Set of colors seen infinitely often during a play may be empty.
- Parity games might require infinite-memory strategies.

Hence, we will only consider parity games with finitely many colors.
In such games, both players have positional winning strategies, even if the game graph is infinite.



Our Goal

- Yesterday, (almost) all game graphs were finite.
- Today, we want to solve games on infinite graphs.

Our Goal

- Yesterday, (almost) all game graphs were finite.
- Today, we want to solve games on infinite graphs.
- But how do we represent infinite graphs finitely so that they can serve as an input to a solution algorithm?
- General idea: Use finite "machines" to encode graphs.

Our Goal

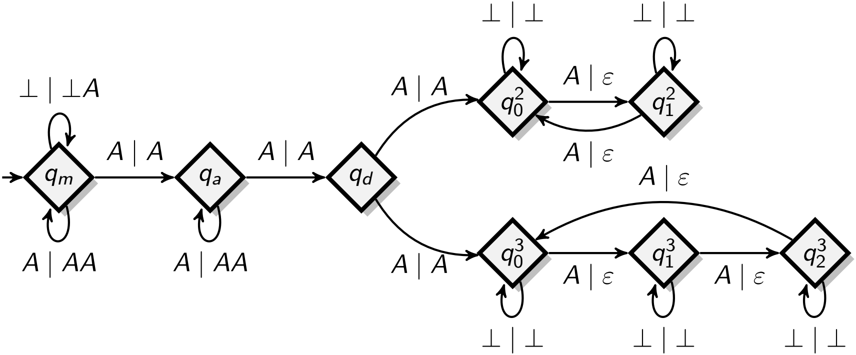
- Yesterday, (almost) all game graphs were finite.
- Today, we want to solve games on infinite graphs.
- But how do we represent infinite graphs finitely so that they can serve as an input to a solution algorithm?
- General idea: Use finite "machines" to encode graphs.

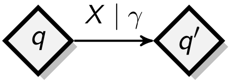
"Cautionary" Tale [Thomas 1995]

There is a recursive Büchi game that has no recursive winning strategies.

- So, Turing-complete machines are too strong. Therefore, we focus on weaker models!

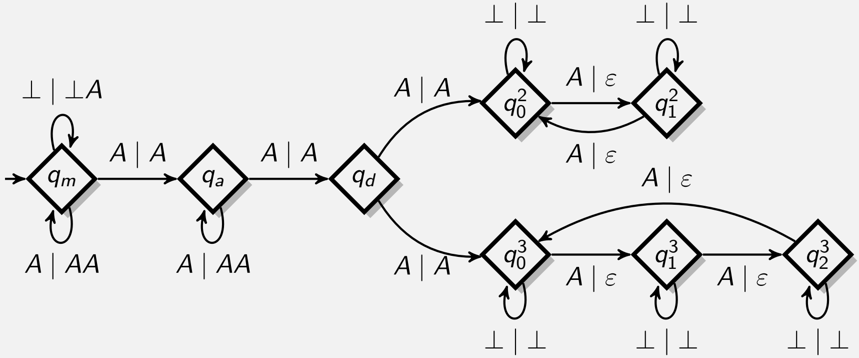
Pushdown Systems



Semantics of a transition  :

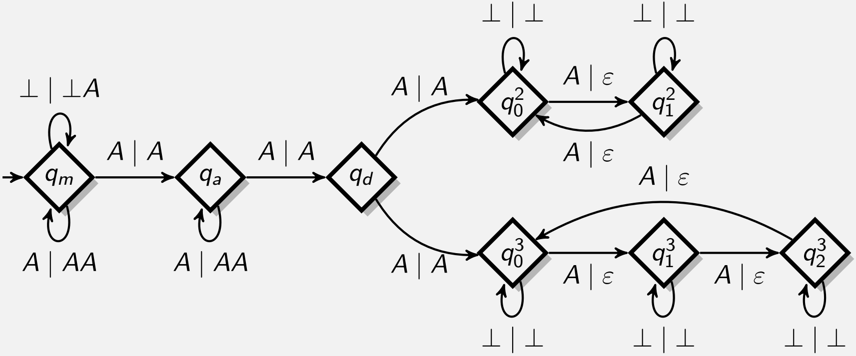
In state q with stack symbol X on the top of the stack, transition to state q' and replace X by sequence γ of stack symbols.

Configuration Graphs



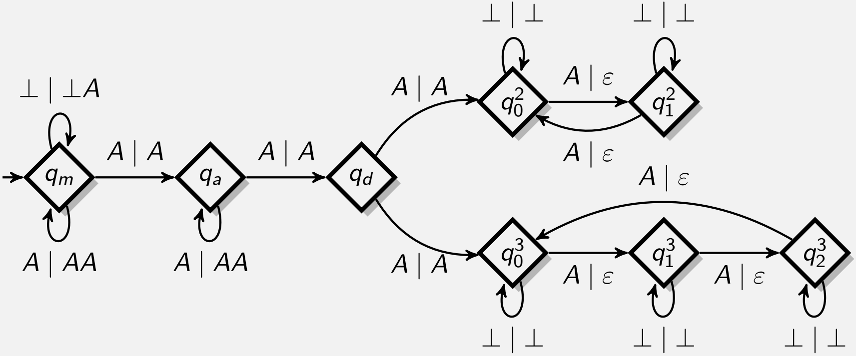
Configuration Graphs

$\rightarrow (q_m, \perp)$



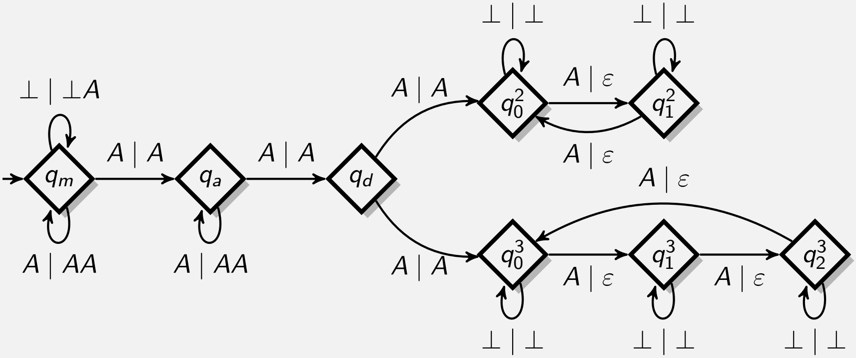
Configuration Graphs

$\rightarrow (q_m, \perp) \longrightarrow (q_m, \perp A^1)$



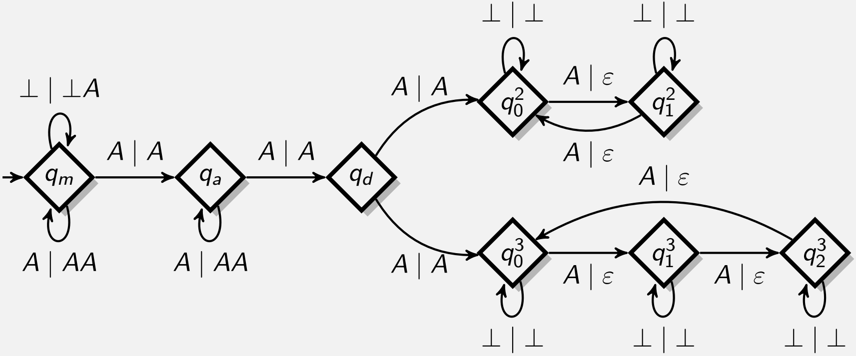
Configuration Graphs

$\rightarrow (q_m, \perp) \longrightarrow (q_m, \perp A^1) \rightarrow (q_m, \perp A^2)$



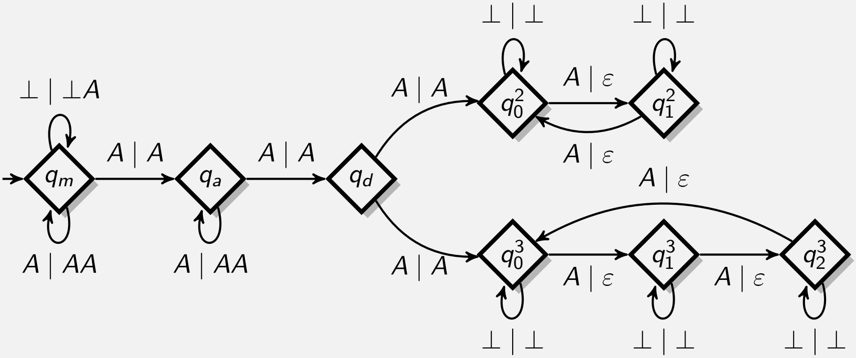
Configuration Graphs

$\rightarrow (q_m, \perp) \longrightarrow (q_m, \perp A^1) \rightarrow (q_m, \perp A^2) \rightarrow (q_m, \perp A^3)$



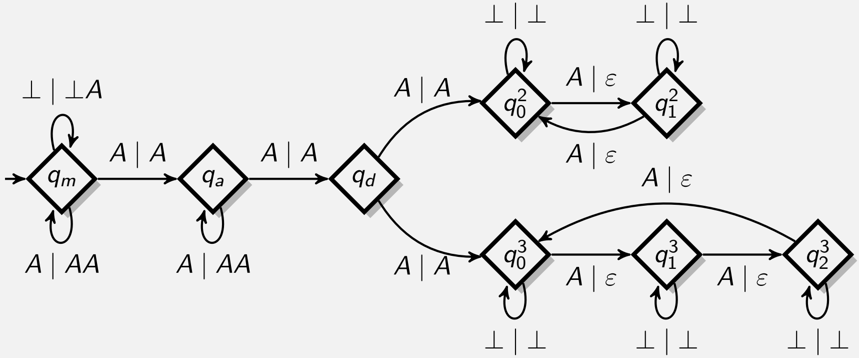
Configuration Graphs

$$\rightarrow (q_m, \bot) \longrightarrow (q_m, \bot A^1) \rightarrow (q_m, \bot A^2) \rightarrow (q_m, \bot A^3) \longrightarrow \cdots$$



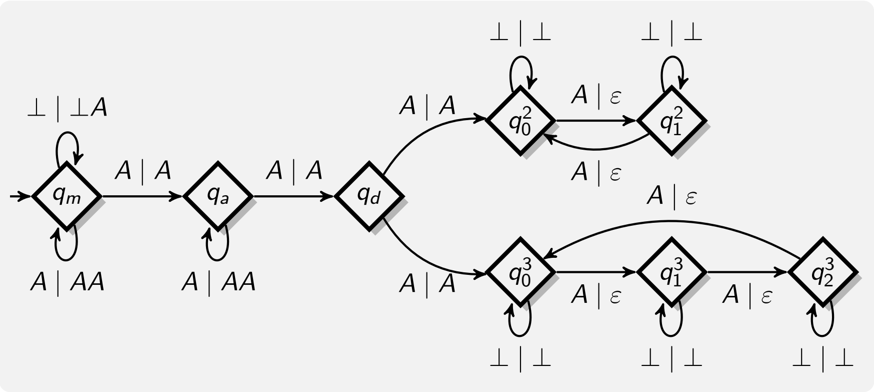
Configuration Graphs

$$\begin{tikzcd} {} \arrow[r] & (q_m, \perp) \arrow[r] & (q_m, \perp A^1) \arrow[r] \arrow[d] & (q_m, \perp A^2) \arrow[r] \arrow[d] & (q_m, \perp A^3) \arrow[r] \arrow[d] & \cdots \\ & & (q_a, \perp A^1) & (q_a, \perp A^2) & (q_a, \perp A^3) & \cdots \end{tikzcd}$$



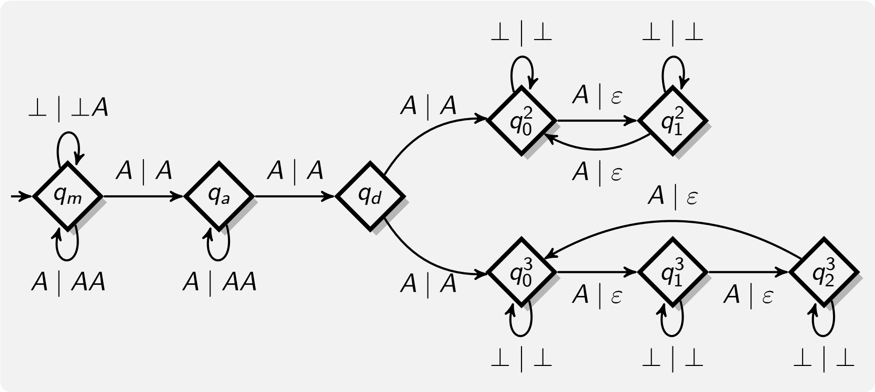
Configuration Graphs

$$
\begin{tikzcd}
{} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \bot) & (q_a, \bot A^1) & (q_a, \bot A^2) & (q_a, \bot A^3) & \cdots
\end{tikzcd}
$$



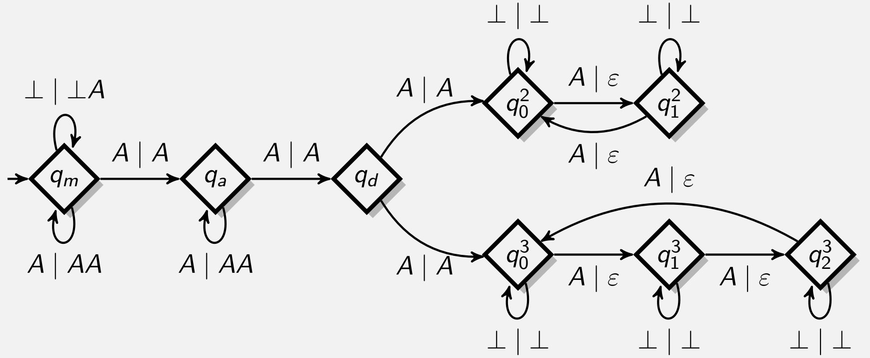
Configuration Graphs

$$\begin{tikzcd} {} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\ & (q_a, \bot) & (q_a, \bot A^1) \arrow[r] & (q_a, \bot A^2) \arrow[r] & (q_a, \bot A^3) \arrow[r] & \cdots \end{tikzcd}$$



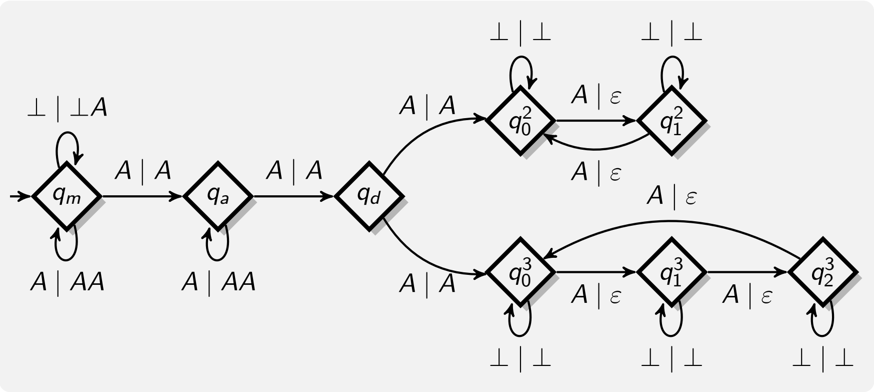
Configuration Graphs

$$
\begin{tikzcd}
{} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \bot) & (q_a, \bot A^1) \arrow[r] \arrow[d] & (q_a, \bot A^2) \arrow[r] \arrow[d] & (q_a, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& & (q_d, \bot A^1) & (q_d, \bot A^2) & (q_d, \bot A^3) & \cdots
\end{tikzcd}
$$

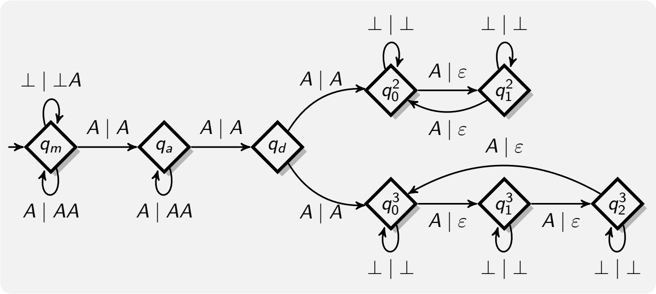


Configuration Graphs

$$\begin{tikzcd}
{} \arrow[r] & (q_m, \perp) \arrow[r] & (q_m, \perp A^1) \arrow[r] \arrow[d] & (q_m, \perp A^2) \arrow[r] \arrow[d] & (q_m, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \perp) & (q_a, \perp A^1) \arrow[r] \arrow[d] & (q_a, \perp A^2) \arrow[r] \arrow[d] & (q_a, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \perp) & (q_d, \perp A^1) & (q_d, \perp A^2) & (q_d, \perp A^3) & \cdots
\end{tikzcd}$$

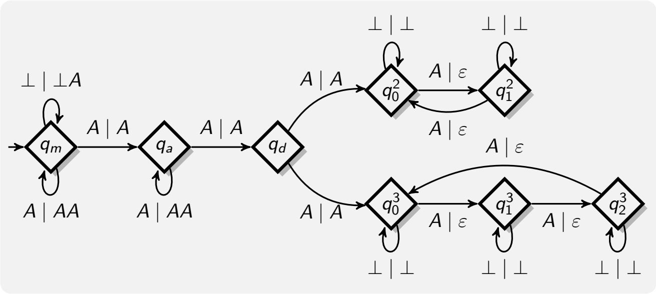


$$
\begin{tikzcd}
\arrow[r] & (q_m, \perp) \arrow[r] & (q_m, \perp A^1) \arrow[r] \arrow[d] & (q_m, \perp A^2) \arrow[r] \arrow[d] & (q_m, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \perp) & (q_a, \perp A^1) \arrow[r] \arrow[d] & (q_a, \perp A^2) \arrow[r] \arrow[d] & (q_a, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \perp) & (q_d, \perp A^1) & (q_d, \perp A^2) & (q_d, \perp A^3) & \cdots
\end{tikzcd}
$$



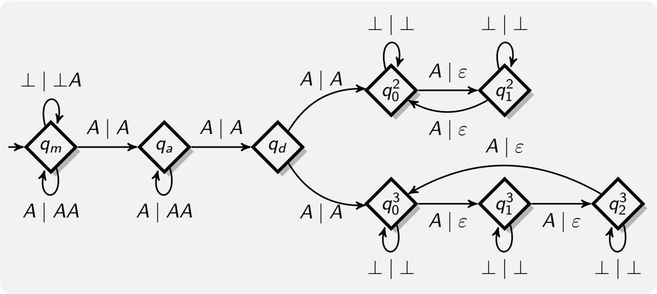
Configuration Graphs

$$\begin{tikzcd}
{} \arrow[r] & (q_m, \perp) \arrow[r] & (q_m, \perp A^1) \arrow[r] \arrow[d] & (q_m, \perp A^2) \arrow[r] \arrow[d] & (q_m, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \perp) & (q_a, \perp A^1) \arrow[r] \arrow[d] & (q_a, \perp A^2) \arrow[r] \arrow[d] & (q_a, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \perp) & (q_d, \perp A^1) \arrow[d] & (q_d, \perp A^2) \arrow[d] & (q_d, \perp A^3) \arrow[d] & \cdots \\
& & (q_0^2, \perp A^1) & (q_0^2, \perp A^2) & (q_0^2, \perp A^3) & \cdots
\end{tikzcd}$$



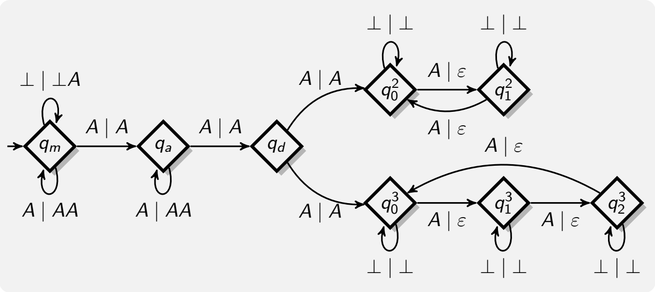
Configuration Graphs

$$
\begin{tikzcd}
{} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \bot) & (q_a, \bot A^1) \arrow[r] \arrow[d] & (q_a, \bot A^2) \arrow[r] \arrow[d] & (q_a, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \bot) & (q_d, \bot A^1) \arrow[d] & (q_d, \bot A^2) \arrow[d] & (q_d, \bot A^3) \arrow[d] & \cdots \\
& & (q_0^2, \bot A^1) \arrow[dl] & (q_0^2, \bot A^2) \arrow[dl] & (q_0^2, \bot A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_1^2, \bot) & (q_1^2, \bot A^1) & (q_1^2, \bot A^2) & (q_1^2, \bot A^3) & \cdots
\end{tikzcd}
$$



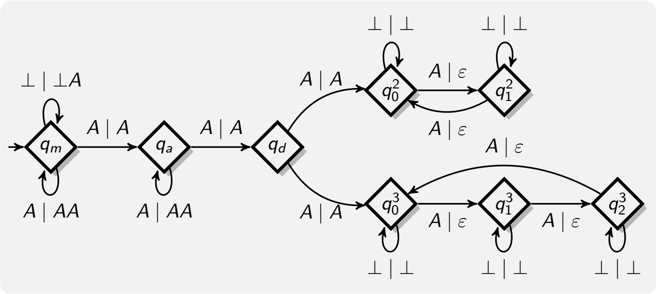
Configuration Graphs

$$\begin{tikzcd}
{} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \bot) & (q_a, \bot A^1) \arrow[r] \arrow[d] & (q_a, \bot A^2) \arrow[r] \arrow[d] & (q_a, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \bot) & (q_d, \bot A^1) \arrow[d] & (q_d, \bot A^2) \arrow[d] & (q_d, \bot A^3) \arrow[d] & \cdots \\
& (q_0^2, \bot) & (q_0^2, \bot A^1) \arrow[dl] & (q_0^2, \bot A^2) \arrow[dl] & (q_0^2, \bot A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_1^2, \bot) & (q_1^2, \bot A^1) \arrow[ul] & (q_1^2, \bot A^2) \arrow[ul] & (q_1^2, \bot A^3) \arrow[ul] & \cdots \arrow[ul]
\end{tikzcd}$$



Configuration Graphs

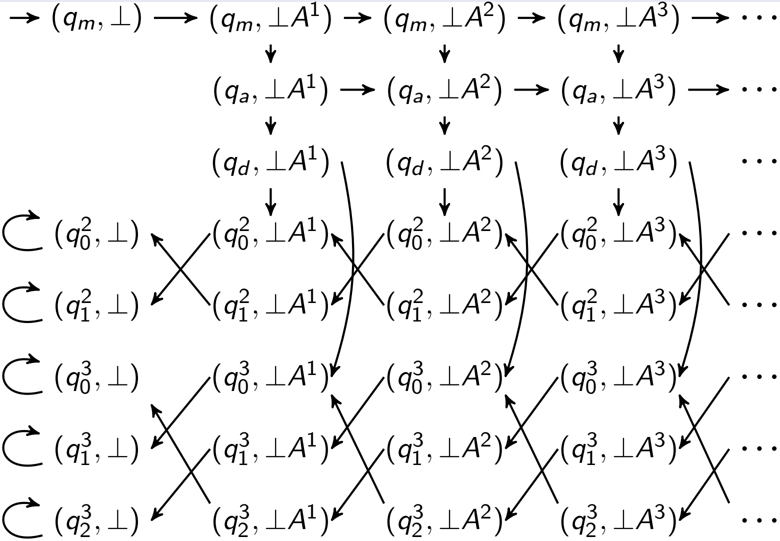
$$
\begin{tikzcd}
{} \arrow[r] & (q_m, \perp) \arrow[r] & (q_m, \perp A^1) \arrow[r] \arrow[d] & (q_m, \perp A^2) \arrow[r] \arrow[d] & (q_m, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \perp) & (q_a, \perp A^1) \arrow[r] \arrow[d] & (q_a, \perp A^2) \arrow[r] \arrow[d] & (q_a, \perp A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \perp) & (q_d, \perp A^1) \arrow[d] & (q_d, \perp A^2) \arrow[d] & (q_d, \perp A^3) \arrow[d] & \cdots \\
& (q_0^2, \perp) \arrow[loop left] & (q_0^2, \perp A^1) \arrow[dl] & (q_0^2, \perp A^2) \arrow[dl] & (q_0^2, \perp A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_1^2, \perp) \arrow[loop left] & (q_1^2, \perp A^1) \arrow[ul] & (q_1^2, \perp A^2) \arrow[ul] & (q_1^2, \perp A^3) \arrow[ul] & \cdots \arrow[ul]
\end{tikzcd}
$$



Configuration Graphs

$$
\begin{tikzcd}
{} \arrow[r] & (q_m, \bot) \arrow[r] & (q_m, \bot A^1) \arrow[r] \arrow[d] & (q_m, \bot A^2) \arrow[r] \arrow[d] & (q_m, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_a, \bot) & (q_a, \bot A^1) \arrow[r] \arrow[d] & (q_a, \bot A^2) \arrow[r] \arrow[d] & (q_a, \bot A^3) \arrow[r] \arrow[d] & \cdots \\
& (q_d, \bot) & (q_d, \bot A^1) \arrow[d] \arrow[ddd, bend left] & (q_d, \bot A^2) \arrow[d] \arrow[ddd, bend left] & (q_d, \bot A^3) \arrow[d] \arrow[ddd, bend left] & \cdots \\
& (q_0^2, \bot) \arrow[loop left] & (q_0^2, \bot A^1) \arrow[dl] & (q_0^2, \bot A^2) \arrow[dl] & (q_0^2, \bot A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_1^2, \bot) \arrow[loop left] & (q_1^2, \bot A^1) \arrow[ul] & (q_1^2, \bot A^2) \arrow[ul] & (q_1^2, \bot A^3) \arrow[ul] & \cdots \arrow[ul] \\
& (q_0^3, \bot) \arrow[loop left] & (q_0^3, \bot A^1) \arrow[dl] & (q_0^3, \bot A^2) \arrow[dl] & (q_0^3, \bot A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_1^3, \bot) \arrow[loop left] & (q_1^3, \bot A^1) \arrow[dl] & (q_1^3, \bot A^2) \arrow[dl] & (q_1^3, \bot A^3) \arrow[dl] & \cdots \arrow[dl] \\
& (q_2^3, \bot) \arrow[loop left] & (q_2^3, \bot A^1) \arrow[uul] & (q_2^3, \bot A^2) \arrow[uul] & (q_2^3, \bot A^3) \arrow[uul] & \cdots \arrow[uul]
\end{tikzcd}
$$

Configuration Graphs



Oftentimes we only consider vertices reachable from the initial vertex.

A Bit More Formal: Syntax

A pushdown system (PDS) $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$ consists of

- a finite set Q of states,
- a stack alphabet Γ,
- an initial state $q_I \in Q$, and
- a transition relation $\Delta \subseteq Q \times \Gamma_\perp \times Q \times \Gamma_\perp^{\leq 2}$,

where $\perp \notin \Gamma$ is a designated stack bottom symbol and $\Gamma_\perp = \Gamma \cup \{\perp\}$.

A Bit More Formal: Syntax

A pushdown system (PDS) $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$ consists of

- a finite set Q of states,
- a stack alphabet Γ,
- an initial state $q_I \in Q$, and
- a transition relation $\Delta \subseteq Q \times \Gamma_\perp \times Q \times \Gamma_\perp^{\leq 2}$,

where $\perp \notin \Gamma$ is a designated stack bottom symbol and $\Gamma_\perp = \Gamma \cup \{\perp\}$.

Assumptions

1. $\perp$ is neither written nor deleted from the stack. Formally:
   - If $(q, \perp, q', \gamma) \in \Delta$, then $\gamma \in \perp \cdot (\Gamma \cup \{\varepsilon\})$, and
   - if $(q, X, q', \gamma) \in \Delta$ for $X \neq \perp$, then $\gamma \in \Gamma^{\leq 2}$.
2. Deadlock freedom: For all $q \in Q$ and all $X \in \Gamma_\perp$, there is a transition $(q, X, q', \gamma) \in \Delta$.

A Bit More Formal: Semantics

- **Stack content**: a finite word in $\bot\Gamma^*$ (i.e., stacks grow to the right).
- **Configuration**: $c = (q, \gamma)$ consisting of a state $q \in Q$ and a stack content γ.
- **Initial configuration**: $(q_I, \bot)$.
- Transition $\tau = (q, X, q', \gamma') \in \Delta$ is **enabled** in a configuration c: if $c = (q, \gamma X)$ for some $\gamma \in \Gamma_\bot^*$. In this case, write $(q, \gamma X) \xrightarrow{\tau} (q', \gamma\gamma')$.

A Bit More Formal: Semantics

- **Stack content**: a finite word in $\bot\Gamma^*$ (i.e., stacks grow to the right).
- **Configuration**: $c = (q, \gamma)$ consisting of a state $q \in Q$ and a stack content γ.
- **Initial configuration**: $(q_I, \bot)$.
- Transition $\tau = (q, X, q', \gamma') \in \Delta$ is **enabled** in a configuration c: if $c = (q, \gamma X)$ for some $\gamma \in \Gamma_\bot^*$. In this case, write $(q, \gamma X) \xrightarrow{\tau} (q', \gamma\gamma')$.

Configuration graph of PDS $\mathcal{P} = (Q, \Gamma, q_I, \Delta)$:

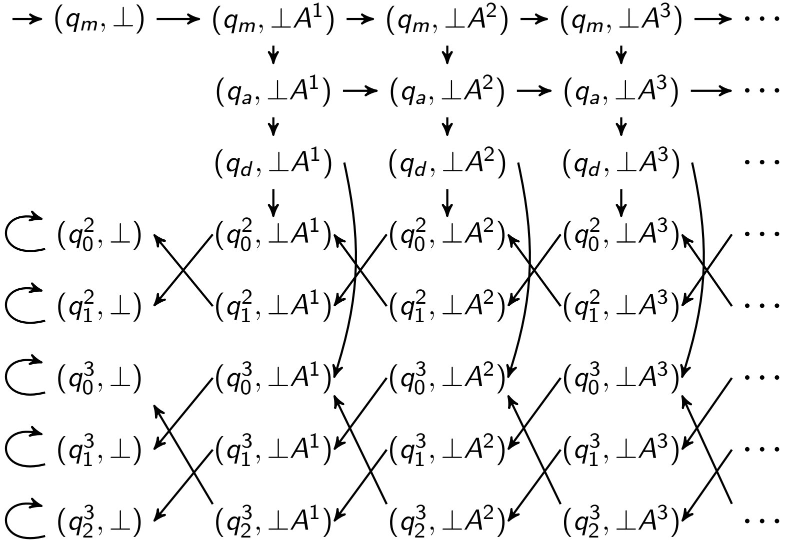
- Vertices: all configurations of $\mathcal{P}$.
- Edges: $(c, c') \in E$ if and only if $c \xrightarrow{\tau} c'$ for some transition τ.

Let’s Play

We want to play parity games on configuration graphs of PDS’s (so-called pushdown graphs).

Let's Play

We want to play parity games on configuration graphs of PDS's (so-called pushdown graphs).



Let's Play

We want to play parity games on configuration graphs of PDS's (so-called pushdown graphs).

- We need to specify a partition of the vertices into the positions of the players and a coloring of the vertices.
- For simplicity, both only depend on the state of a configuration (more general definitions are possible).

Let's Play

We want to play parity games on configuration graphs of PDS's (so-called pushdown graphs).

- We need to specify a partition of the vertices into the positions of the players and a coloring of the vertices.
- For simplicity, both only depend on the state of a configuration (more general definitions are possible).

Definition

A PDS (Q, Γ, q_I, Δ), a partition $Q = Q_0 \cup Q_1$, and a coloring $\Omega' \colon Q \to \mathbb{N}$ induce the parity game (V, V_0, V_1, E, Ω) where

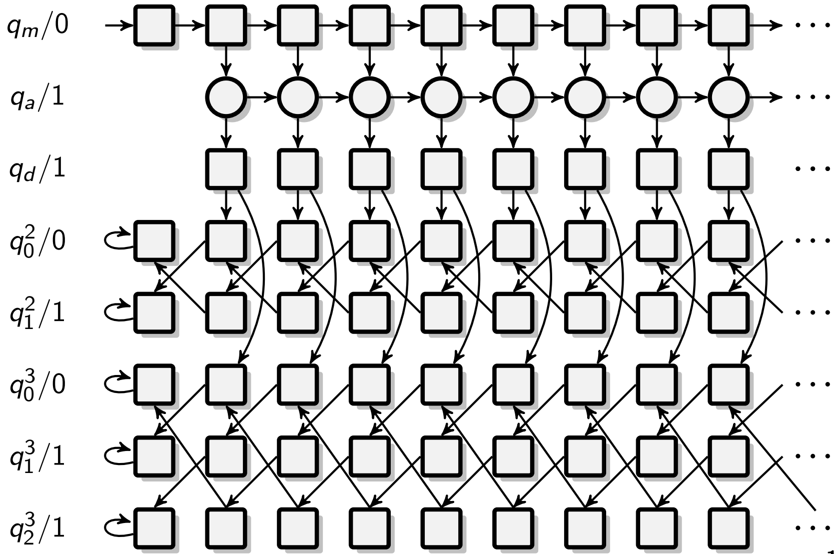
- (V, E) is the configuration graph induced by the PDS,
- $V_i = \{(q, \gamma) \in V \mid q \in Q_i\}$, and
- $\Omega(q, \gamma) = \Omega'(q)$.

Let's Play

Consider $Q_0 = \{q_a\}$, $Q_1 = Q \setminus Q_0$, $\Omega'(q_m) = \Omega(q_0^2) = \Omega(q_0^3) = 0$ and $\Omega'(q) = 1$ for all other states q.

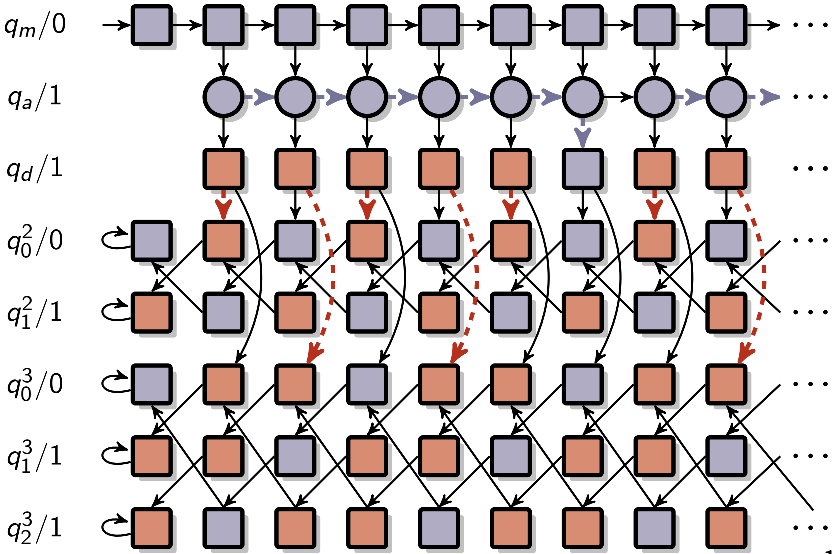
Let's Play

Consider $Q_0 = \{q_a\}$, $Q_1 = Q \setminus Q_0$, $\Omega'(q_m) = \Omega(q_0^2) = \Omega(q_0^3) = 0$ and $\Omega'(q) = 1$ for all other states q.



Let's Play

Consider $Q_0 = \{q_a\}$, $Q_1 = Q \setminus Q_0$, $\Omega'(q_m) = \Omega(q_0^2) = \Omega(q_0^3) = 0$ and $\Omega'(q) = 1$ for all other states q.



Our Goal

Solving parity games on pushdown graphs: given a PDS, and a partition and a coloring of its states, determine whether Player 0 *wins the induced parity game from the initial configuration.*

Our Goal

Solving parity games on pushdown graphs: given a PDS, and a partition and a coloring of its states, determine whether Player 0 *wins the induced parity game from the initial configuration.*

All inputs are finite objects, but the induced parity game has

- infinitely many vertices,
- finitely many colors, and
- finite branching.

An (Abridged) History

- **Thomas 1995**: Can winning strategies for parity games in pushdown games be finitely represented?

An (Abridged) History

- **Thomas 1995**: Can winning strategies for parity games in pushdown games be finitely represented?
- Solving parity games on pushdown graphs is decidable by a reduction to MSO satisfiability over pushdown graphs **[Rabin 1969, Muller, Schupp 1985]**. However, this does not yield (tight) complexity bounds.

An (Abridged) History

- **Thomas 1995**: Can winning strategies for parity games in pushdown games be finitely represented?
- Solving parity games on pushdown graphs is decidable by a reduction to MSO satisfiability over pushdown graphs **[Rabin 1969, Muller, Schupp 1985]**. However, this does not yield (tight) complexity bounds.
- **Walukiewicz 1996**: Solving parity games on pushdown graphs is EXPTIME-complete.

An (Abridged) History

- **Thomas 1995**: Can winning strategies for parity games in pushdown games be finitely represented?
- Solving parity games on pushdown graphs is decidable by a reduction to MSO satisfiability over pushdown graphs **[Rabin 1969, Muller, Schupp 1985]**. However, this does not yield (tight) complexity bounds.
- **Walukiewicz 1996**: Solving parity games on pushdown graphs is $\textsc{ExpTime}$-complete.
- **Kupferman and Vardi 2000**: An alternative proof (with optimal complexity) via alternating two-way tree automata.

An (Abridged) History

- **Thomas 1995**: Can winning strategies for parity games in pushdown games be finitely represented?
- Solving parity games on pushdown graphs is decidable by a reduction to MSO satisfiability over pushdown graphs **[Rabin 1969, Muller, Schupp 1985]**. However, this does not yield (tight) complexity bounds.
- **Walukiewicz 1996**: Solving parity games on pushdown graphs is EXPTIME-complete.
- **Kupferman and Vardi 2000**: An alternative proof (with optimal complexity) via alternating two-way tree automata.
- Thereafter: Extensions, refinements, simplifications, etc.

Some Terminology

- The **stack height** of a configuration (q, γ) is $|\gamma| - 1 \in \mathbb{N}$.

Some Terminology

- The **stack height** of a configuration (q, γ) is $|\gamma| - 1 \in \mathbb{N}$.
- We classify transitions (q, X, q', γ) of a PDS into three types:
  - **push**: if $|\gamma| = 2$, i.e., a symbol is pushed onto the stack and the stack height increases by one.
  - **skip**: if $|\gamma| = 1$, i.e., the topmost stack symbol is replaced and the stack height does not change.
  - **pop**: if $|\gamma| = 0$, i.e., a symbol is popped off the stack and the stack height decreases by one.

Walukiewicz's Reduction: Intuition

Given a parity game $\mathcal{G}$ induced by a PDS (Q, Γ, q_I, Δ), a partition (Q_0, Q_1), and a coloring Ω, we construct a finite parity game $\mathcal{G}'$ such that Player i wins $\mathcal{G}$ from $(q_I, \perp)$ if and only if Player i wins $\mathcal{G}'$ from some designated vertex v.

Walukiewicz's Reduction: Intuition

Given a parity game $\mathcal{G}$ induced by a PDS (Q, Γ, q_I, Δ), a partition (Q_0, Q_1), and a coloring Ω, we construct a finite parity game $\mathcal{G}'$ such that Player i wins $\mathcal{G}$ from $(q_I, \perp)$ if and only if Player i wins $\mathcal{G}'$ from some designated vertex v.

Idea

- Simulate plays in $\mathcal{G}$ by plays in $\mathcal{G}'$ by only storing the state and the topmost stack symbol.

Walukiewicz's Reduction: Intuition

Given a parity game $\mathcal{G}$ induced by a PDS (Q, Γ, q_I, Δ), a partition (Q_0, Q_1), and a coloring Ω, we construct a finite parity game $\mathcal{G}'$ such that Player i wins $\mathcal{G}$ from $(q_I, \perp)$ if and only if Player i wins $\mathcal{G}'$ from some designated vertex v.

Idea

- Simulate plays in $\mathcal{G}$ by plays in $\mathcal{G}'$ by only storing the state and the topmost stack symbol.
- This works for push- and skip-, but not for pop-transitions.

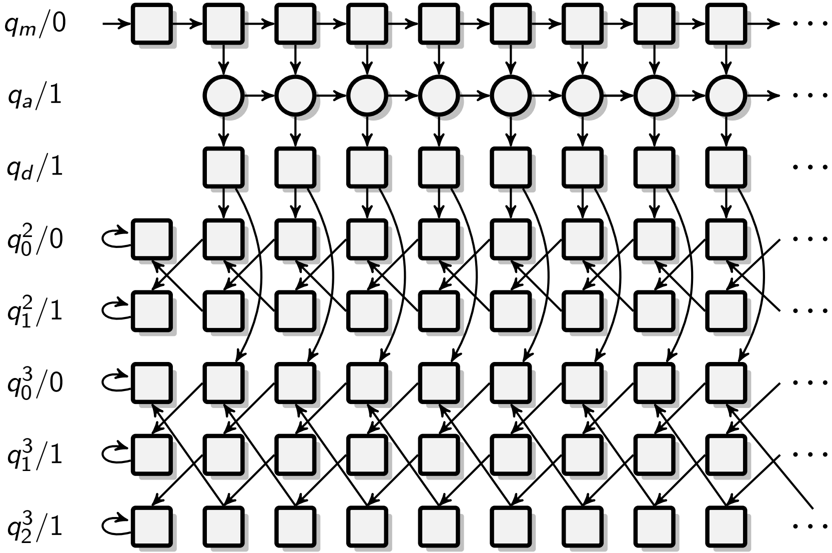
Walukiewicz's Reduction: Intuition

Given a parity game $\mathcal{G}$ induced by a PDS (Q, Γ, q_I, Δ), a partition (Q_0, Q_1), and a coloring Ω, we construct a finite parity game $\mathcal{G}'$ such that Player i wins $\mathcal{G}$ from $(q_I, \perp)$ if and only if Player i wins $\mathcal{G}'$ from some designated vertex v.

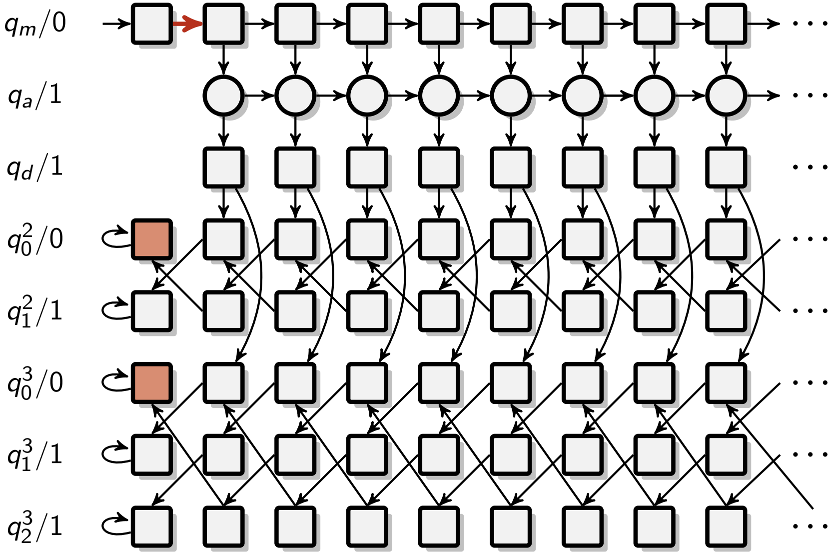
Idea

- Simulate plays in $\mathcal{G}$ by plays in $\mathcal{G}'$ by only storing the state and the topmost stack symbol.
- This works for push- and skip-, but not for pop-transitions.
- So, we add a mechanism for Player 0 to make predictions about the possible states a play is in the next time the current stack height is reached again (if it is at all).
- Player 1 verifies these predictions and can "jump" over parts of the play.

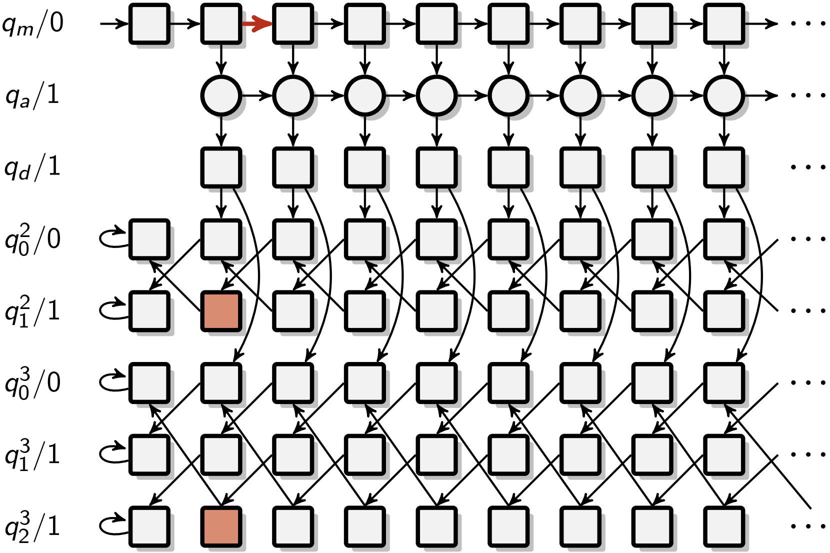
Intuition



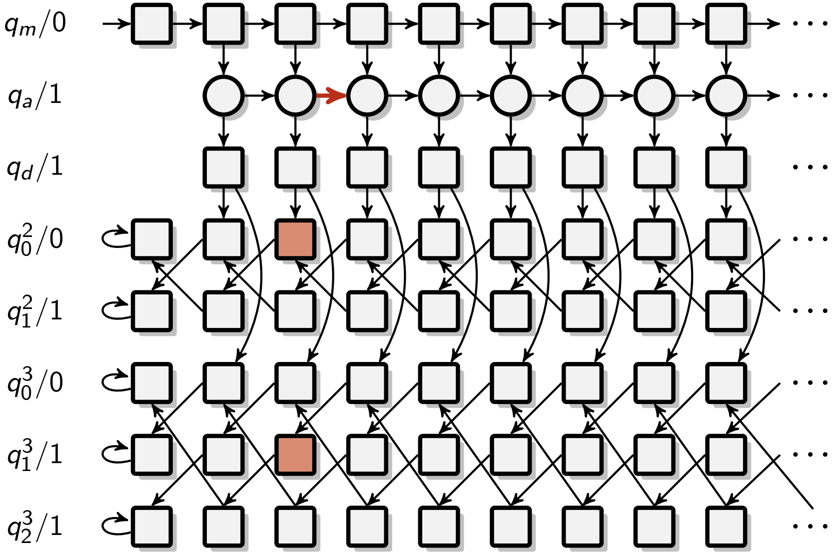
Intuition



Intuition



Intuition



Predictions

We do not only need to consider states in a prediction, but also the colors seen along the play infixes leading to these states.

Predictions

We do not only need to consider states in a prediction, but also the colors seen along the play infixes leading to these states.

Definition

Let $C = \Omega(Q)$ be the set of colors of the game $\mathcal{G}$.

- A prediction is a tuple $(P_c)_{c \in C}$ s.t. each P_c is a subset of Q.
- $\mathfrak{P}$ is the set of all predictions.

Predictions

We do not only need to consider states in a prediction, but also the colors seen along the play infixes leading to these states.

Definition

Let $C = \Omega(Q)$ be the set of colors of the game $\mathcal{G}$.

- A prediction is a tuple $(P_c)_{c \in C}$ s.t. each P_c is a subset of Q.
- $\mathfrak{P}$ is the set of all predictions.

Remark

There are exponentially many predictions.

Predictions: Intuition

- With every simulated push-transition, Player 0 makes a prediction $(P_c)_{c \in C}$ about the states that are reached when the current stack height s (before the push-transition is executed) is reached for the first time again (if it is reached at all).

Predictions: Intuition

- With every simulated push-transition, Player 0 makes a prediction $(P_c)_{c \in C}$ about the states that are reached when the current stack height s (before the push-transition is executed) is reached for the first time again (if it is reached at all).
- $q \in P_c$ if and only if state q is reached and the maximal color seen between the push-transition and the pop-transition bringing the play to stack height s for the first time is c.

Predictions: Intuition

- With every simulated push-transition, Player 0 makes a prediction $(P_c)_{c \in C}$ about the states that are reached when the current stack height s (before the push-transition is executed) is reached for the first time again (if it is reached at all).
- $q \in P_c$ if and only if state q is reached and the maximal color seen between the push-transition and the pop-transition bringing the play to stack height s for the first time is c.
- Player 1 has two choices to react:
  1. Accept $(P_c)_{c \in C}$ by jumping to some $q \in \bigcup_{c \in C} P_c$ and continue simulation there.

Predictions: Intuition

- With every simulated push-transition, Player 0 makes a prediction $(P_c)_{c \in C}$ about the states that are reached when the current stack height s (before the push-transition is executed) is reached for the first time again (if it is reached at all).
- $q \in P_c$ if and only if state q is reached and the maximal color seen between the push-transition and the pop-transition bringing the play to stack height s for the first time is c.
- Player 1 has two choices to react:
  1. Accept $(P_c)_{c \in C}$ by jumping to some $q \in \bigcup_{c \in C} P_c$ and continue simulation there.
  2. Verify correctness of $(P_c)_{c \in C}$: simulate push-transition. When the top of the stack is eventually popped, correctness of $(P_c)_{c \in C}$ can be checked.
     - If prediction is correct, simulation ends and Player 0 wins.
     - If prediction is incorrect, simulation ends and Player 1 wins.

The Finite Parity Game $\mathcal{G}'$: Vertices

Let $q \in Q$, $A, B \in \Gamma$, $c, d \in C$, and $P, R \in \mathfrak{P}$. $\mathcal{G}'$ contains the following vertices:

- $\mathrm{Check}[q, A, P, c, d]$: Encode configuration of $\mathcal{G}$.
- $\mathrm{Push}[P, c, q, AB]$: signal intent to perform a push-transition.
- $\mathrm{Claim}[P, c, q, AB, R]$: to make a new prediction.
- $\mathrm{Jump}[q, A, P, c, d]$: Jump over part of simulated play.
- $\mathrm{Win}_0[q]$ and $\mathrm{Win}_1[q]$: sink vertices reached when prediction is checked.

The Finite Parity Game $\mathcal{G}'$: Vertices

Let $q \in Q$, $A, B \in \Gamma$, $c, d \in C$, and $P, R \in \mathfrak{P}$. $\mathcal{G}'$ contains the following vertices:

- $\text{Check}[q, A, P, c, d]$: Encode configuration of $\mathcal{G}$.
- $\text{Push}[P, c, q, AB]$: signal intent to perform a push-transition.
- $\text{Claim}[P, c, q, AB, R]$: to make a new prediction.
- $\text{Jump}[q, A, P, c, d]$: Jump over part of simulated play.
- $\text{Win}_0[q]$ and $\text{Win}_1[q]$: sink vertices reached when prediction is checked.

- All Push-vertices and all $\text{Check}[q, \ldots]$-vertices with $q \in Q_0$ belong to Player 0 in $\mathcal{G}'$.
- All other vertices belong to Player 1 in $\mathcal{G}'$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **skip-transitions** $(q, A, q', B) \in \Delta$, $\mathcal{G}'$ has the edge

$$\text{Check}[q, A, P, c, d] \to \text{Check}[q', B, P, \max\{c, \Omega(q')\}, \Omega(q')]$$

for all $P \in \mathfrak{P}$ and all $c, d \in C$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

$$\mathsf{Check}[q, A, P, c, d] \to \mathsf{Push}[P, c, q', BC],$$

for all $P, R \in \mathfrak{P}$ and all $c, c', c_j, d \in C$, and all $q_j \in R_{c_j}$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

$$\mathsf{Check}[q, A, P, c, d] \to \mathsf{Push}[P, c, q', BC],$$

$$\mathsf{Push}[P, c, q', BC] \to \mathsf{Claim}[P, c, q', BC, R],$$

for all $P, R \in \mathfrak{P}$ and all $c, c', c_j, d \in C$, and all $q_j \in R_{c_j}$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Push}[P, c, q', BC],$$

$$\mathrm{Push}[P, c, q', BC] \to \mathrm{Claim}[P, c, q', BC, R],$$

$$\mathrm{Claim}[P, c, q', BC, R] \to \mathrm{Check}[q', C, R, \Omega(q'), \Omega(q')],$$

for all $P, R \in \mathfrak{P}$ and all $c, c', c_j, d \in C$, and all $q_j \in R_{c_j}$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Push}[P, c, q', BC],$$

$$\mathrm{Push}[P, c, q', BC] \to \mathrm{Claim}[P, c, q', BC, R],$$

$$\mathrm{Claim}[P, c, q', BC, R] \to \mathrm{Check}[q', C, R, \Omega(q'), \Omega(q')],$$

$$\mathrm{Claim}[P, c, q', BC, R] \to \mathrm{Jump}[q_j, B, P, c, c_j], \text{ and}$$

for all $P, R \in \mathfrak{P}$ and all $c, c', c_j, d \in C$, and all $q_j \in R_{c_j}$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **push-transitions** $(q, A, q', BC) \in \Delta$, $\mathcal{G}$ has the edges

$$\mathsf{Check}[q, A, P, c, d] \to \mathsf{Push}[P, c, q', BC],$$

$$\mathsf{Push}[P, c, q', BC] \to \mathsf{Claim}[P, c, q', BC, R],$$

$$\mathsf{Claim}[P, c, q', BC, R] \to \mathsf{Check}[q', C, R, \Omega(q'), \Omega(q')],$$

$$\mathsf{Claim}[P, c, q', BC, R] \to \mathsf{Jump}[q_j, B, P, c, c_j], \text{ and}$$

$$\mathsf{Jump}[q_j, B, P, c, c_j] \to \mathsf{Check}[q_j, B, P, \max\{c, c_j, \Omega(q')\}, \max\{c_j, \Omega(q')\}]$$

for all $P, R \in \mathfrak{P}$ and all $c, c', c_j, d \in C$, and all $q_j \in R_{c_j}$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **pop-transitions** $(q, A, q', \varepsilon) \in \Delta$, $\mathcal{G}$ has the edge

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Win}_0[q']$$

if $q' \in P_c$ and

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Win}_1[q']$$

if $q' \notin P_c$.

The Finite Parity Game $\mathcal{G}'$: Edges

For all **pop-transitions** $(q, A, q', \varepsilon) \in \Delta$, $\mathcal{G}$ has the edge

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Win}_0[q']$$

if $q' \in P_c$ and

$$\mathrm{Check}[q, A, P, c, d] \to \mathrm{Win}_1[q']$$

if $q' \notin P_c$.

Finally, $\mathcal{G}$ has the edges

$$\mathrm{Win}_i[q] \to \mathrm{Win}_i[q]$$

for all $i \in \{0, 1\}$ and all $q \in Q$.

The Finite Parity Game $\mathcal{G}'$: Colors

- Check$[\ldots, d]$ has color d.
- $\text{Win}_i[q]$ has color i.
- All other vertices have color 0 (which is neutral, i.e., it never determines the winner).

Finishing Touches

1. $\mathcal{G}'$ has exponentially many vertices in $|Q| + |\Gamma|$ and at most two more colors than $\mathcal{G}$.
2. It can be solved in exponential time (in $|Q| + |\Gamma|$).

Finishing Touches

1. $\mathcal{G}'$ has exponentially many vertices in $|Q| + |\Gamma|$ and at most two more colors than $\mathcal{G}$.
2. It can be solved in exponential time (in $|Q| + |\Gamma|$).

Lemma (Walukiewicz 1996)

Player 0 *wins* $\mathcal{G}$ *from* $(q_I, \perp)$ *if and only if Player* 0 *wins* $\mathcal{G}'$ *from* $\mathrm{Check}[q_I, \perp, (\emptyset)_{c \in C}, \Omega(q_I), \Omega(q_I)]$.

Proof Sketch: From $\mathcal{G}$ to $\mathcal{G}'$

- Let σ be a positional winning strategy for Player 0 from $(q_I, \perp)$.
- Use σ to play $\mathcal{G}'$: use it to pick successors and to make predictions using continuations of plays that are consistent with σ.

Proof Sketch: From $\mathcal{G}$ to $\mathcal{G}'$

- Let σ be a positional winning strategy for Player 0 from $(q_I, \perp)$.
- Use σ to play $\mathcal{G}'$: use it to pick successors and to make predictions using continuations of plays that are consistent with σ.
- Such predictions are always correct, i.e., the bad sink state is never visited.
- Hence, infinitely many Check-vertices are visited.

Proof Sketch: From $\mathcal{G}$ to $\mathcal{G}'$

- Let σ be a positional winning strategy for Player 0 from $(q_I, \perp)$.
- Use σ to play $\mathcal{G}'$: use it to pick successors and to make predictions using continuations of plays that are consistent with σ.
- Such predictions are always correct, i.e., the bad sink state is never visited.
- Hence, infinitely many Check-vertices are visited.
- The sequence of colors at these Check-vertices “corresponds” to the sequence of colors of a play in $\mathcal{G}$ that is consistent with σ. Hence, it satisfies the parity condition.

Proof Sketch: From $\mathcal{G}'$ to $\mathcal{G}$

- Let σ' be a positional winning strategy for Player 0 from $\mathrm{Check}[q_I, \bot, (\emptyset)_{c \in C}, \Omega(q_I), \Omega(q_I)]$.
- Use σ' to play in $\mathcal{G}$, assuming Player 1 never "jumps". This works until a pop-transition is simulated, which leads to a sink vertex $\mathrm{Win}_0[q]$ in $\mathcal{G}'$.

Proof Sketch: From $\mathcal{G}'$ to $\mathcal{G}$

- Let σ' be a positional winning strategy for Player 0 from $\mathrm{Check}[q_I, \perp, (\emptyset)_{c \in C}, \Omega(q_I), \Omega(q_I)]$.
- Use σ' to play in $\mathcal{G}$, assuming Player 1 never "jumps". This works until a pop-transition is simulated, which leads to a sink vertex $\mathrm{Win}_0[q]$ in $\mathcal{G}'$.
- To continue the simulation, backtrack to the corresponding Claim-vertex where the prediction was made and continue like σ' would do if Player 1 had chosen to jump to q.

Proof Sketch: From $\mathcal{G}'$ to $\mathcal{G}$

- Let σ' be a positional winning strategy for Player 0 from $\mathrm{Check}[q_I, \perp, (\emptyset)_{c \in C}, \Omega(q_I), \Omega(q_I)]$.
- Use σ' to play in $\mathcal{G}$, assuming Player 1 never "jumps". This works until a pop-transition is simulated, which leads to a sink vertex $\mathrm{Win}_0[q]$ in $\mathcal{G}'$.
- To continue the simulation, backtrack to the corresponding Claim-vertex where the prediction was made and continue like σ' would do if Player 1 had chosen to jump to q.
- One can again relate the sequence of colors visited by the play in $\mathcal{G}$ with those of some play in $\mathcal{G}'$ that is consistent with σ'. Hence, it satisfies the parity condition.

Main Theorem

Theorem (Walukiewicz 1996)

Solving parity games on pushdown graphs is EXPTIME*-complete.*

Proof.

Upper bound:

- The finite parity game we have just constructed is of exponential size with polynomially many colors, and can therefore be solved in exponential time. It has the same winner as the original game.
- Lower bound: encode alternating polynomial-time Turing machines. □

Some Further Results

- The reduction to finite parity games even yields finite representations of winning strategies (via pushdown transducers).

Some Further Results

- The reduction to finite parity games even yields finite representations of winning strategies (via pushdown transducers).
- The winning regions of both players are regular subsets of $Q{\perp}\Gamma^*$ **[Cachat 2002, Serre 2003]**.

Some Further Results

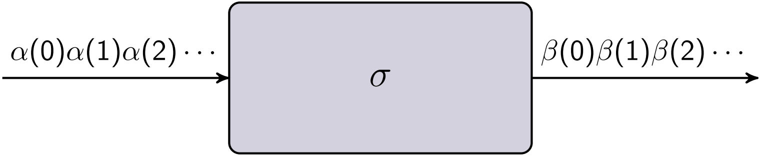
- The reduction to finite parity games even yields finite representations of winning strategies (via pushdown transducers).
- The winning regions of both players are regular subsets of $Q\perp\Gamma^*$ **[Cachat 2002, Serre 2003]**.
- Solving one-counter parity games (induced by PDS's with a single stack symbol) is only PSPACE-complete. **[Serre 2006, Jancar, Sawa 2007]**

Some Further Results

- The reduction to finite parity games even yields finite representations of winning strategies (via pushdown transducers).
- The winning regions of both players are regular subsets of $Q\bot\Gamma^*$ **[Cachat 2002, Serre 2003]**.
- Solving one-counter parity games (induced by PDS's with a single stack symbol) is only $\mathrm{PS}\textsc{pace}$-complete. **[Serre 2006, Jancar, Sawa 2007]**
- In the other direction, the result has been extended to higher-order pushdown systems (having stacks of stacks of stacks....): Solving parity games induced by order-k pushdown automata is in k-$\textsc{ExpTime}$ **[Cachat 2003]**.

Reminder: Gale-Stewart Games

In a Gale-Stewart game $\mathcal{G}(R)$, Player I (input) and Player O (output) produce sequences $\alpha(0)\alpha(1)\alpha(2)\cdots\Sigma_I^\omega$ and $\beta(0)\beta(1)\beta(2)\cdots\Sigma_O^\omega$ of input and output symbols, a relation $R \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$ determines the winner.



We are looking for a (letter-to-letter) strategy σ such that

$$(\alpha, \sigma(\alpha)) \in R$$

for every possible input $\alpha \in \Sigma_I^\omega$.

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a,b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:

Pl. O:

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a
Pl. O:	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a
Pl. O:	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a
Pl. O:	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a
Pl. O:	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a
Pl. O:	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a
Pl. O:	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b
Pl. O:	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b
Pl. O:	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a
Pl. O:	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a
Pl. O:	a	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a,b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b
Pl. O:	a	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b
Pl. O:	a	a	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a
Pl. O:	a	a	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a
Pl. O:	a	a	a	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b
Pl. O:	a	a	a	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b
Pl. O:	a	a	a	a	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a
Pl. O:	a	a	a	a	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a
Pl. O:	a	a	a	a	a	a	a	a	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b
Pl. O:	a	a	a	a	a	a	a	a	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b
Pl. O:	a	a	a	a	a	a	a	a	a	b

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a,b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b	b
Pl. O:	a	a	a	a	a	a	a	a	a	b	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b	b
Pl. O:	a	a	a	a	a	a	a	a	a	b	a

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b	b	a
Pl. O:	a	a	a	a	a	a	a	a	a	b	a	

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b	b	a
Pl. O:	a	a	a	a	a	a	a	a	a	b	a	b

Example

Consider

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

and let's play.

Pl. I:	a	a	a	b	a	b	a	b	a	b	b	a	$\cdots$
Pl. O:	a	a	a	a	a	a	a	a	a	b	a	b	$\cdots$

Player *O* wins!

Our Goal

- A **strategy** for Player O: mapping $\sigma \colon \Sigma_I^* \to \Sigma_O$ mapping a finite sequence of input letters to one output letter.
- (α, β) is **consistent** with σ: $\beta(n) = \sigma(\alpha(0) \cdots \alpha(n))$ for all n.
- σ is **winning** in $\mathcal{G}(R)$ with $R \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$: all consistent pairs (α, β) are in R.

Our Goal

- A **strategy** for Player O: mapping $\sigma\colon \Sigma_I^* \to \Sigma_O$ mapping a finite sequence of input letters to one output letter.
- (α, β) is **consistent** with σ: $\beta(n) = \sigma(\alpha(0) \cdots \alpha(n))$ for all n.
- σ is **winning** in $\mathcal{G}(R)$ with $R \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$: all consistent pairs (α, β) are in R.

Example

$$\sigma(w) = \begin{cases} a & \text{if } w \in a^*, \\ a & \text{if } w = a^n b w' \text{ with } |w'| \leq 2n-1, \\ b & \text{if } w = a^n b w' \text{ with } |w'| = 2n, \\ a & \text{otherwise.} \end{cases}$$

is a winning strategy for $\mathcal{G}(R)$ with

$$R = \{(a^n b w, a^{3n} b w') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}.$$

Our Goal

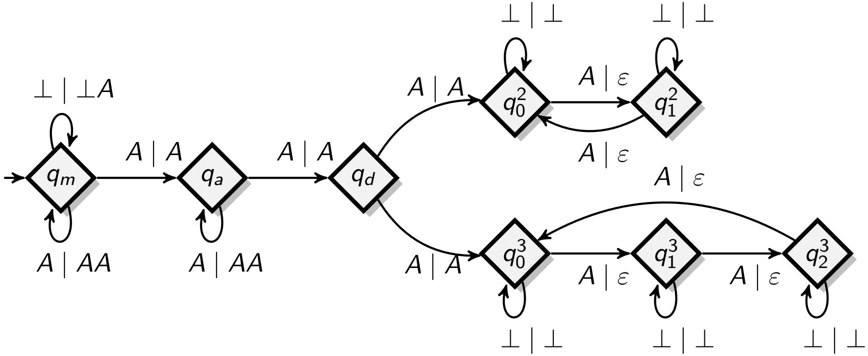
- A **strategy** for Player O: mapping $\sigma \colon \Sigma_I^* \to \Sigma_O$ mapping a finite sequence of input letters to one output letter.
- (α, β) is **consistent** with σ: $\beta(n) = \sigma(\alpha(0) \cdots \alpha(n))$ for all n.
- σ is **winning** in $\mathcal{G}(R)$ with $R \subseteq \Sigma_I^\omega \times \Sigma_O^\omega$: all consistent pairs (α, β) are in R.

Yesterday, we have seen how to solve Gale-Stewart games with ω-regular winning conditions. Today, we consider the following problem:

> *Solving Gale-Stewart games with ω-contextfree winning conditions: Given an ω-contextfree relation R, does Player O have a winning strategy for $\mathcal{G}(R)$?*

Pushdown Automata

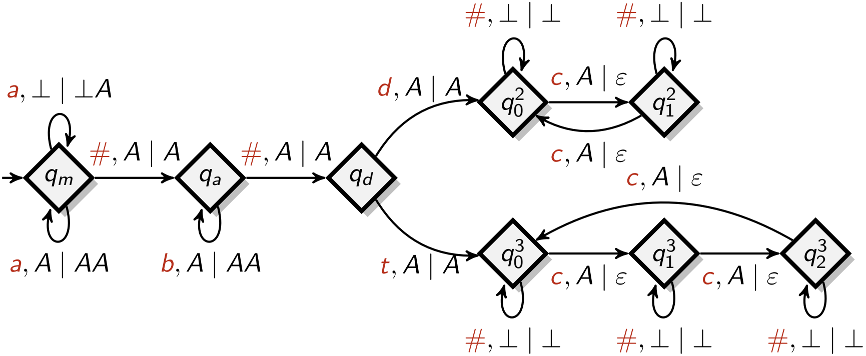
We have seen pushdown systems and pushdown graphs. Now, we want to accept ω-languages with pushdown automata.



Pushdown Automata

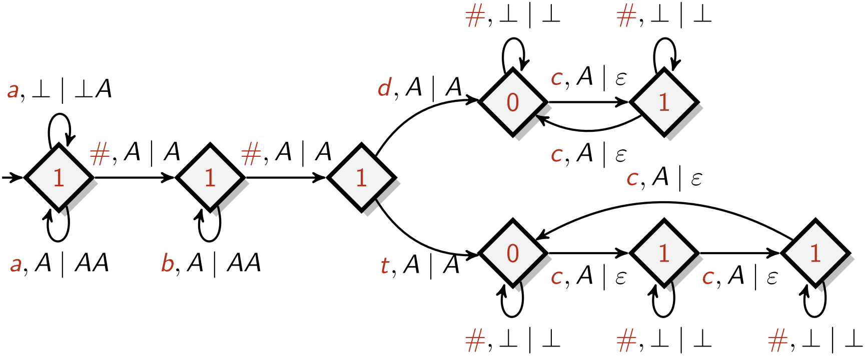
We have seen pushdown systems and pushdown graphs. Now, we want to accept ω-languages with pushdown automata. To this end, we

- equip the transitions of pushdown systems with letters they process (we allow ε-transitions), and



We have seen pushdown systems and pushdown graphs. Now, we want to accept ω-languages with pushdown automata. To this end, we

- equip the transitions of pushdown systems with letters they process (we allow ε-transitions), and
- add an acceptance condition: we again use parity.



More Formally: Syntax and Semantics

An ω-pushdown (parity) automaton (ω-PDA) $\mathcal{P} = (\mathcal{S}, \Sigma, \ell, \Omega)$ consists of

- a PDS $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$,
- an input alphabet Σ,
- a labeling $\ell \colon \Delta \to \Sigma \cup \{\varepsilon\}$ of the transitions by letters or ε (we say τ is an $\ell(\tau)$-transition), and
- a coloring $\Omega \colon Q \to \mathbb{N}$ of the states.

More Formally: Syntax and Semantics

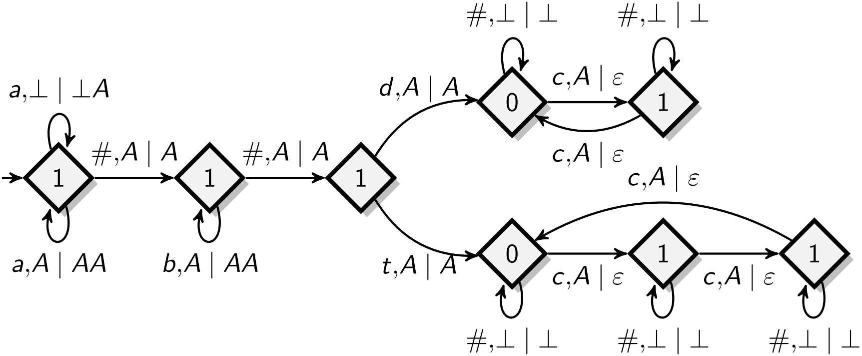
An ω-pushdown (parity) automaton (ω-PDA) $\mathcal{P} = (\mathcal{S}, \Sigma, \ell, \Omega)$ consists of

- a PDS $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$,
- an input alphabet Σ,
- a labeling $\ell \colon \Delta \to \Sigma \cup \{\varepsilon\}$ of the transitions by letters or ε (we say τ is an $\ell(\tau)$-transition), and
- a coloring $\Omega \colon Q \to \mathbb{N}$ of the states.

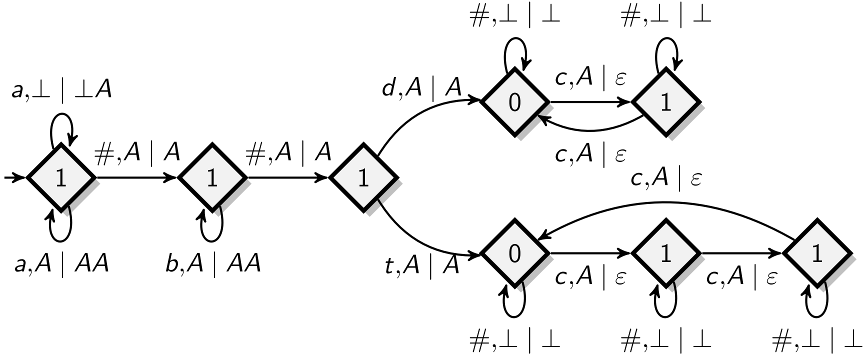
An ω-word $w_0 w_1 w_2 \cdots \in \Sigma^\omega$ is in the language $L(\mathcal{P})$ of $\mathcal{P}$ if there is a path $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ through the configuration graph of $\mathcal{S}$ such that

- $c_0 = (q_I, \bot)$,
- $\ell(\tau_0)\ell(\tau_1)\ell(\tau_2) \cdots = w_0 w_1 w_2 \cdots$, and
- $\Omega(q_0)\Omega(q_1)\Omega(q_2) \cdots$ satisfies the parity condition, where q_n is the state of c_n.

Back to the Example



Back to the Example



$$L(\mathcal{P}) = \{a^m\#b^n\#dc^{m+n}\#^\omega \mid m > 0 \text{ and } m+n \bmod 2 = 0\} \cup$$
$$\{a^m\#b^n\#tc^{m+n}\#^\omega \mid m > 0 \text{ and } m+n \bmod 3 = 0\}$$

Technicalities

Let $\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma_I^\omega$ and $\beta = \beta(0)\beta(1)\beta(2)\cdots \in \Sigma_O^\omega$ be two ω-words. We want to combine these two words into a single ω-word:

Technicalities

Let $\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma_I^\omega$ and $\beta = \beta(0)\beta(1)\beta(2)\cdots \in \Sigma_O^\omega$ be two ω-words. We want to combine these two words into a single ω-word:

- Yesterday:

$$\alpha\hat{}\beta = \alpha(0)\beta(0)\alpha(1)\beta(1)\alpha(2)\beta(2)\cdots \in (\Sigma_I \cdot \Sigma_O)^\omega.$$

Technicalities

Let $\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma_I^\omega$ and $\beta = \beta(0)\beta(1)\beta(2)\cdots \in \Sigma_O^\omega$ be two ω-words. We want to combine these two words into a single ω-word:

- Yesterday:

$$\alpha\hat{}\beta = \alpha(0)\beta(0)\alpha(1)\beta(1)\alpha(2)\beta(2)\cdots \in (\Sigma_I \cdot \Sigma_O)^\omega.$$

- Today:

$$\binom{\alpha}{\beta} = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots \in (\Sigma_I \times \Sigma_O)^\omega.$$

Technicalities

Let $\alpha = \alpha(0)\alpha(1)\alpha(2)\cdots \in \Sigma_I^\omega$ and $\beta = \beta(0)\beta(1)\beta(2)\cdots \in \Sigma_O^\omega$ be two ω-words. We want to combine these two words into a single ω-word:

- Yesterday:

$$\alpha\hat{\ }\beta = \alpha(0)\beta(0)\alpha(1)\beta(1)\alpha(2)\beta(2)\cdots \in (\Sigma_I \cdot \Sigma_O)^\omega.$$

- Today:

$$\binom{\alpha}{\beta} = \binom{\alpha(0)}{\beta(0)}\binom{\alpha(1)}{\beta(1)}\binom{\alpha(2)}{\beta(2)}\cdots \in (\Sigma_I \times \Sigma_O)^\omega.$$

In the following, we consider winning conditions $L \subseteq (\Sigma_I \times \Sigma_O)^\omega$, which represent the relation

$$R_L = \left\{ (\alpha, \beta) \in \Sigma_I^\omega \times \Sigma_O^\omega \;\middle|\; \binom{\alpha}{\beta} \in L \right\}.$$

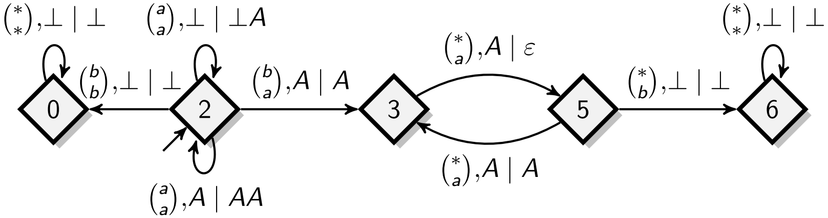
Back to the Example

$$R = \{(a^n bw, a^{3n} bw') \mid n \geq 0, w, w' \in \{a, b\}^\omega\} \cup \{(a^\omega, a^\omega)\}$$

is encoded by ($*$ denotes an arbitrary letter)

$$\left\{ \binom{a}{a}^n \binom{b}{a} \binom{*}{a}^{2n-1} \binom{*}{b} \binom{*}{*}^\omega \;\middle|\; n \geq 1 \right\} \cup \left\{ \binom{b}{b} \binom{*}{*}^\omega, \binom{a}{a}^\omega \right\},$$

which is accepted by the ω-PDA (state name = color)



Bad News

Theorem

Solving Gale-Stewart games with ω-contextfree winning conditions is undecidable.

Theorem

Solving Gale-Stewart games with ω-contextfree winning conditions is undecidable.

Proof.

Given an ω-language $L \subseteq \Sigma^\omega$, let $L_=$ be the ω-language

$$L_= = \left\{ \binom{\alpha(0)}{\alpha(0)} \binom{\alpha(1)}{\alpha(1)} \binom{\alpha(2)}{\alpha(2)} \cdots \;\middle|\; \alpha(0)\alpha(1)\alpha(2)\cdots \in L \right\}.$$

Bad News

Theorem

Solving Gale-Stewart games with ω-contextfree winning conditions is undecidable.

Proof.

Given an ω-language $L \subseteq \Sigma^\omega$, let $L_=$ be the ω-language

$$L_= = \left\{ \binom{\alpha(0)}{\alpha(0)} \binom{\alpha(1)}{\alpha(1)} \binom{\alpha(2)}{\alpha(2)} \cdots \;\middle|\; \alpha(0)\alpha(1)\alpha(2)\cdots \in L \right\}.$$

- Given an ω-PDA $\mathcal{P}$ accepting a language L, one can effectively construct an ω-PDA $\mathcal{P}_=$ for $L_=$.

Theorem

Solving Gale-Stewart games with ω-contextfree winning conditions is undecidable.

Proof.

Given an ω-language $L \subseteq \Sigma^\omega$, let $L_=$ be the ω-language

$$L_= = \left\{ \binom{\alpha(0)}{\alpha(0)} \binom{\alpha(1)}{\alpha(1)} \binom{\alpha(2)}{\alpha(2)} \cdots \;\middle|\; \alpha(0)\alpha(1)\alpha(2)\cdots \in L \right\}.$$

- Given an ω-PDA $\mathcal{P}$ accepting a language L, one can effectively construct an ω-PDA $\mathcal{P}_=$ for $L_=$.
- L is universal if and only if Player O has a winning strategy for $\mathcal{G}(L_=)$.

Bad News

Theorem

Solving Gale-Stewart games with ω-contextfree winning conditions is undecidable.

Proof.

Given an ω-language $L \subseteq \Sigma^\omega$, let $L_=$ be the ω-language

$$L_= = \left\{ \binom{\alpha(0)}{\alpha(0)}\binom{\alpha(1)}{\alpha(1)}\binom{\alpha(2)}{\alpha(2)} \cdots \;\middle|\; \alpha(0)\alpha(1)\alpha(2)\cdots \in L \right\}.$$

- Given an ω-PDA $\mathcal{P}$ accepting a language L, one can effectively construct an ω-PDA $\mathcal{P}_=$ for $L_=$.
- L is universal if and only if Player O has a winning strategy for $\mathcal{G}(L_=)$.
- Universality for ω-PDA is undecidable, as it is undecidable for PDA over finite words. $\square$

What about Deterministic ω-PDA?

Universality is only undecidable for nondeterministic (ω-) PDA, but decidable for deterministic (ω-) PDA. Thus, solving Gale-Stewart games with deterministic contextfree winning conditions could still be decidable as well.

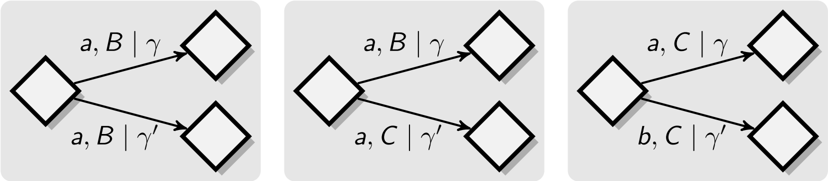
What about Deterministic ω-PDA?

Universality is only undecidable for nondeterministic (ω-) PDA, but decidable for deterministic (ω-) PDA. Thus, solving Gale-Stewart games with deterministic contextfree winning conditions could still be decidable as well.

$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

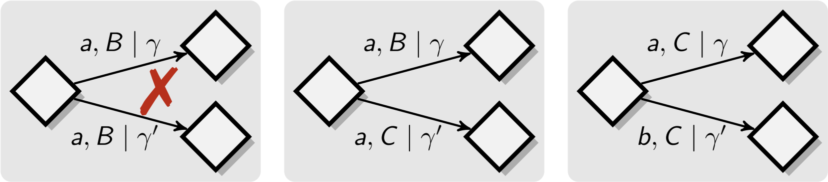
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

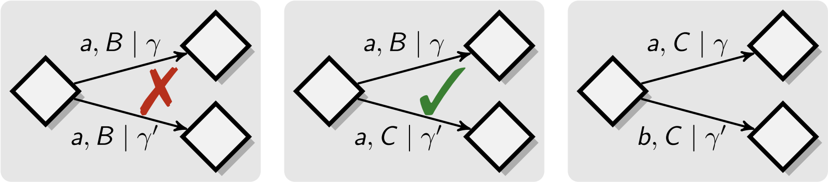
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

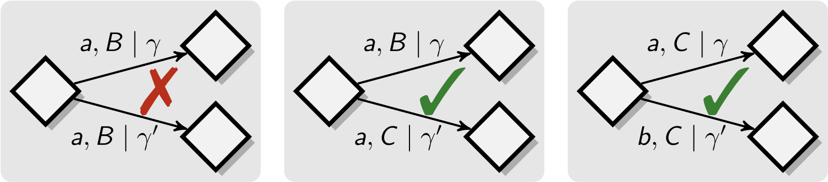
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

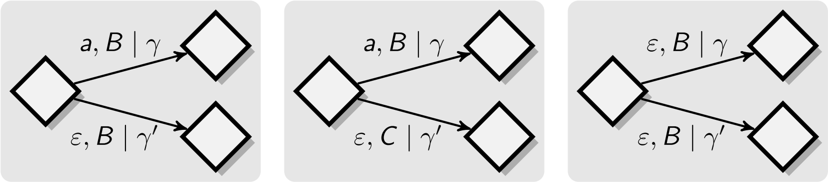
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

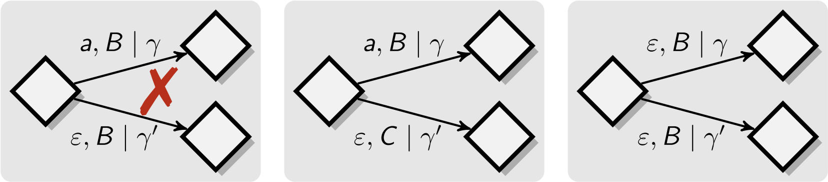
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

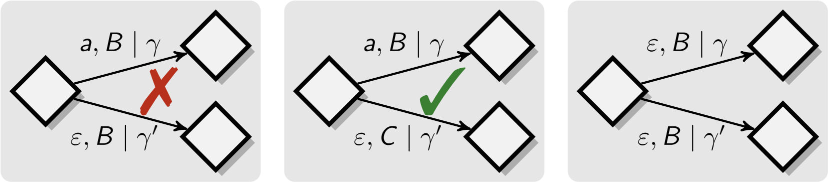
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

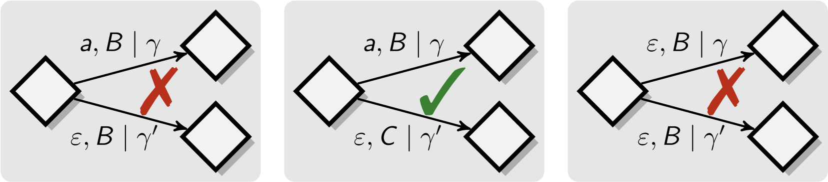
What about Deterministic ω-PDA?



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

What about Deterministic ω-PDA?

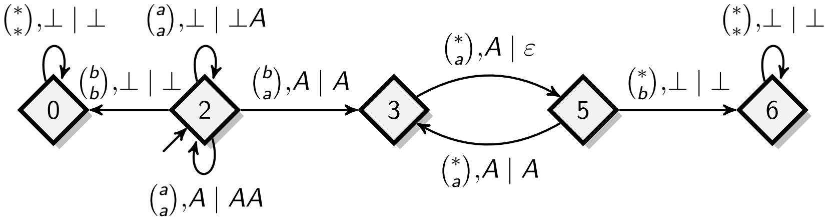


$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\perp$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\perp$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

What about Deterministic ω-PDA?

Example



$\mathcal{P}$ is deterministic if

- for every $q \in Q$, every $A \in \Gamma_\bot$, and every $a \in \Sigma \cup \{\varepsilon\}$, there is at most one a-transition of the form $(q, A, q', \gamma) \in \Delta$ for some q' and some γ, and
- for every $q \in Q$ and every $A \in \Gamma_\bot$, if there is an ε-transition $(q, A, q_1, \gamma_1) \in \Delta$ for some q_1 and some γ_1, then there is no $a \in \Sigma$ such that there is an a-transition $(q, A, q_2, \gamma_2) \in \Delta$ for some q_2 and some γ_2.

Good News

Corollary (Walukiewicz 1996)

Solving Gale-Stewart games with deterministic ω-contextfree winning conditions is $\textsc{ExpTime}$*-complete.*

Good News

Corollary (Walukiewicz 1996)

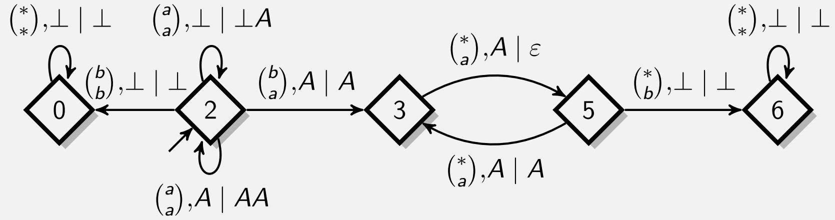
Solving Gale-Stewart games with deterministic ω-contextfree winning conditions is $\textsc{ExpTime}$*-complete.*

Proof Sketch

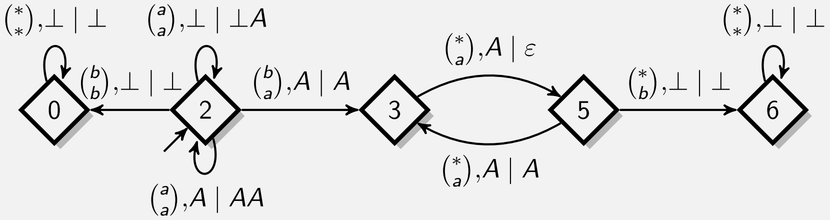
We show that Gale-Stewart games with ω-contextfree winning conditions can be simulated by parity games on pushdown graphs and vice versa (with a polynomial blowup).

1. Given an ω-PDA $\mathcal{P}$, we construct a polynomial-sized PDS $\mathcal{S}'$ such that Player O wins $\mathcal{G}(L(\mathcal{P}))$ if and only if Player 0 wins the parity game induced by $\mathcal{S}'$.
2. Given a PDS $\mathcal{S}'$, we construct a polynomial-sized ω-PDA $\mathcal{P}$ such that Player 0 wins the parity game induced by $\mathcal{S}'$ if and only if Player O wins $\mathcal{G}(L(\mathcal{P}))$.

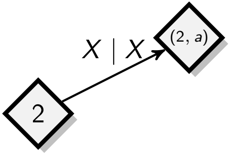
From Gale-Stewart to Parity: Intuition

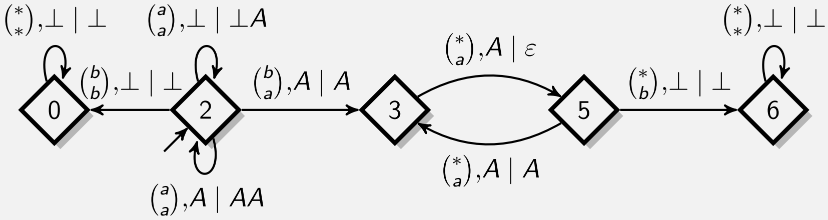


From Gale-Stewart to Parity: Intuition

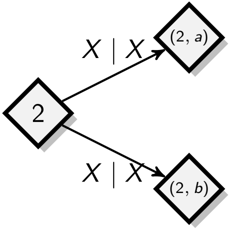


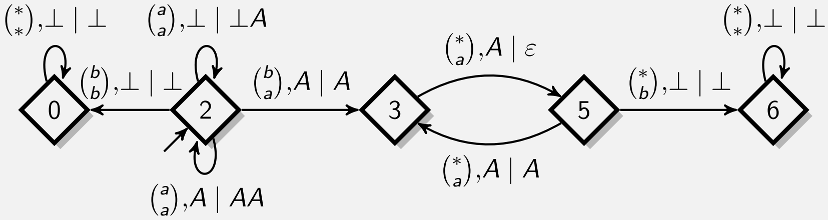
From Gale-Stewart to Parity: Intuition



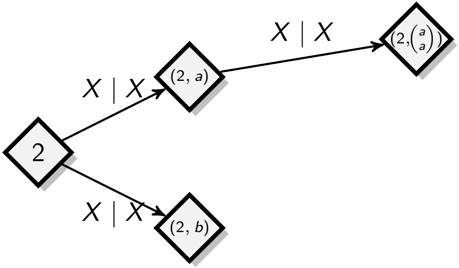


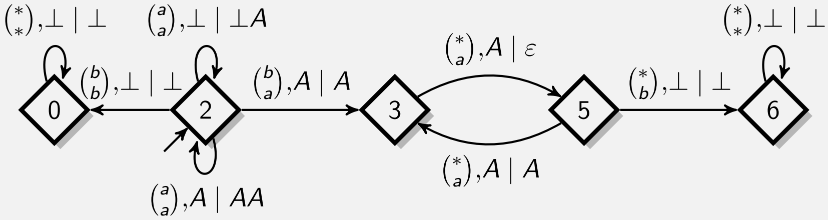
From Gale-Stewart to Parity: Intuition



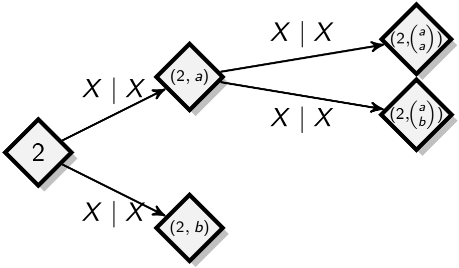


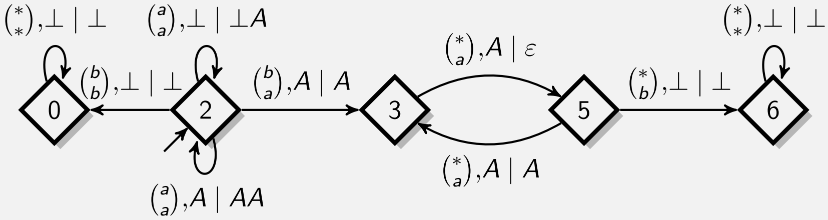
From Gale-Stewart to Parity: Intuition



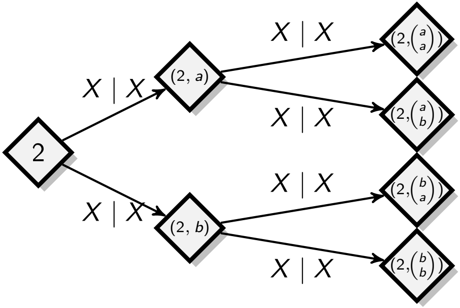


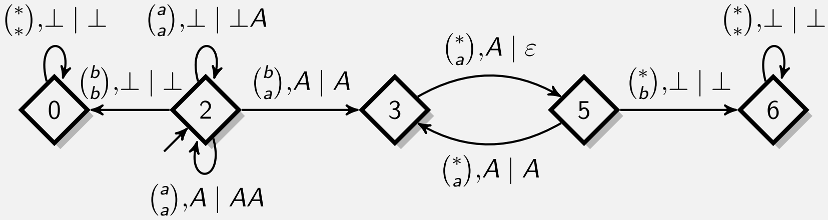
From Gale-Stewart to Parity: Intuition



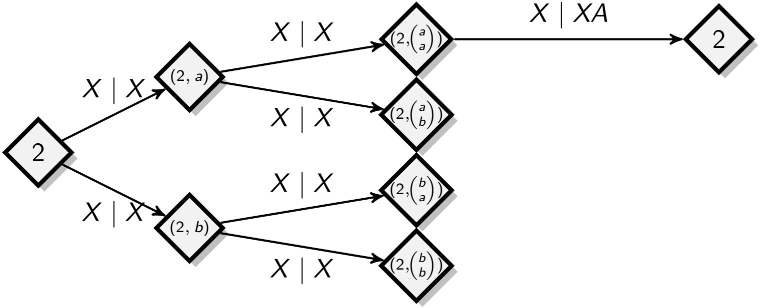


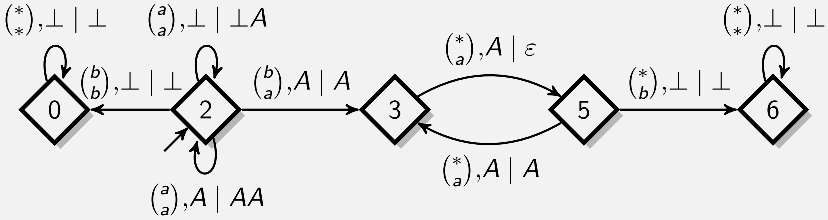
From Gale-Stewart to Parity: Intuition



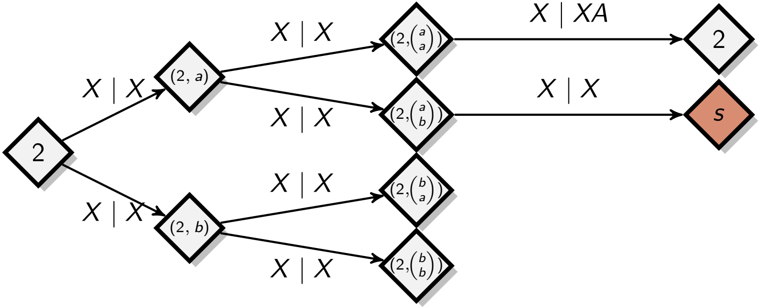


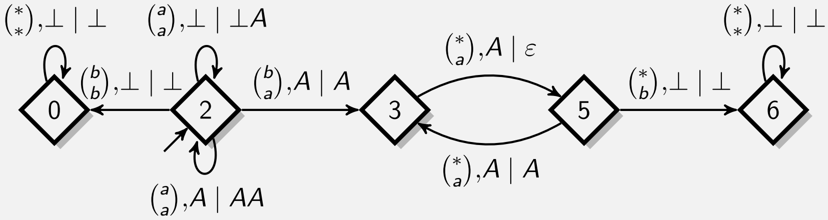
From Gale-Stewart to Parity: Intuition



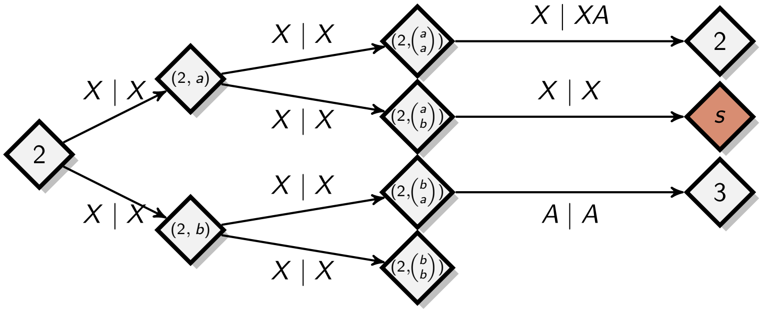


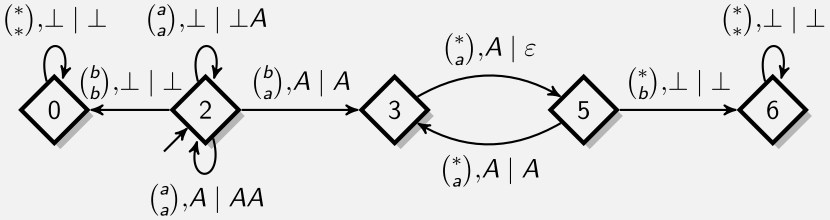
From Gale-Stewart to Parity: Intuition



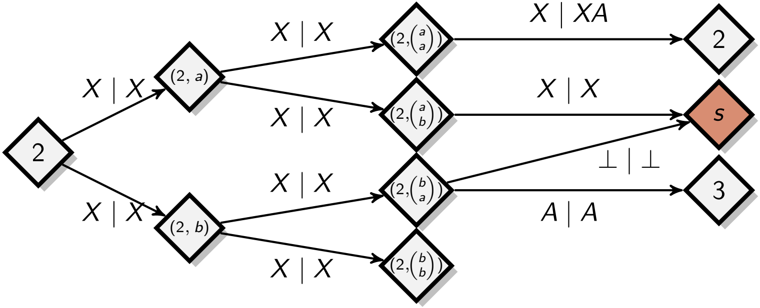


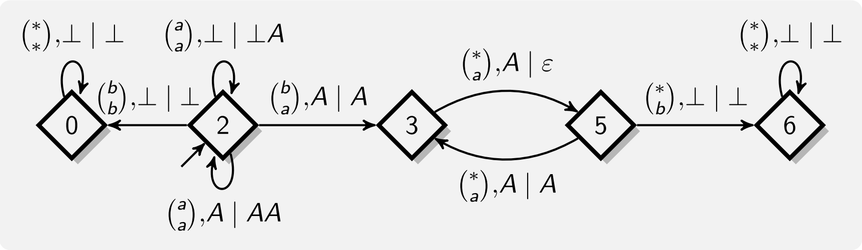
From Gale-Stewart to Parity: Intuition



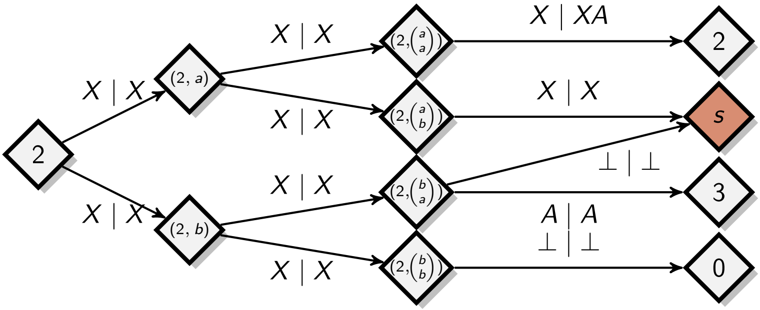


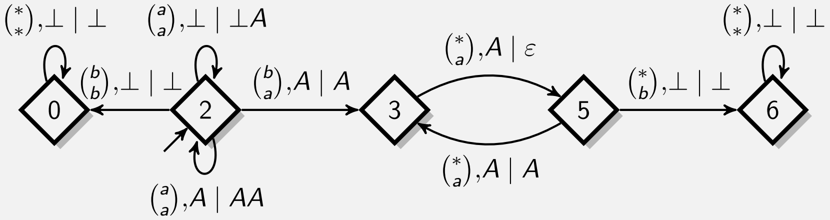
From Gale-Stewart to Parity: Intuition



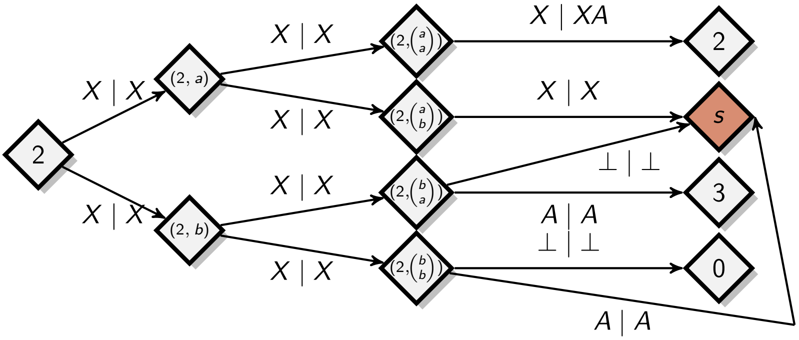


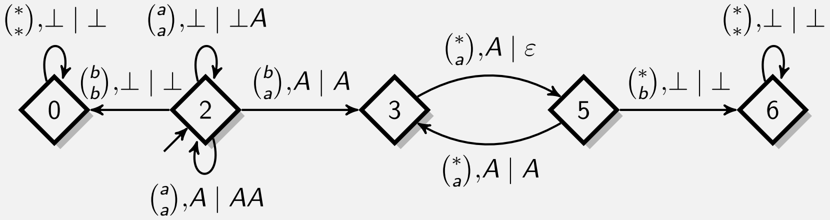
From Gale-Stewart to Parity: Intuition





From Gale-Stewart to Parity: Intuition





(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

- $\mathcal{S}'$ has all states of $\mathcal{S}$, plus some auxiliary ones:
  - $\{(q, a), (q, \binom{a}{b}) \mid q \in Q, a \in \Sigma_I, b \in \Sigma_O\}$ to mimic picking letters, and
  - a fresh sink state s.

(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

- $\mathcal{S}'$ has all states of $\mathcal{S}$, plus some auxiliary ones:
  - $\{(q, a), (q, \binom{a}{b}) \mid q \in Q, a \in \Sigma_I, b \in \Sigma_O\}$ to mimic picking letters, and
  - a fresh sink state s.
- The initial state and the stack alphabet of $\mathcal{S}$ are the same as in $\mathcal{S}'$.

(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

- $\mathcal{S}'$ has all states of $\mathcal{S}$, plus some auxiliary ones:
  - $\{(q, a), (q, \binom{a}{b}) \mid q \in Q, a \in \Sigma_I, b \in \Sigma_O\}$ to mimic picking letters, and
  - a fresh sink state s.
- The initial state and the stack alphabet of $\mathcal{S}$ are the same as in $\mathcal{S}'$.
- The transitions of $\mathcal{S}'$ are defined to implement the picking of letters and then the deterministic simulation of $\mathcal{P}$ (might involve ε-transitions).

(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

- $\mathcal{S}'$ has all states of $\mathcal{S}$, plus some auxiliary ones:
  - $\{(q, a), (q, \binom{a}{b}) \mid q \in Q, a \in \Sigma_I, b \in \Sigma_O\}$ to mimic picking letters, and
  - a fresh sink state s.
- The initial state and the stack alphabet of $\mathcal{S}$ are the same as in $\mathcal{S}'$.
- The transitions of $\mathcal{S}'$ are defined to implement the picking of letters and then the deterministic simulation of $\mathcal{P}$ (might involve ε-transitions).
- Player 0 moves at states of the form $q \in Q$, Player 1 at states of the form $(q, a) \in Q \times \Sigma_I$ and, for completeness, at all other states (irrelevant, as these are "deterministic").

(Slightly) More Formally

Construct PDS $\mathcal{S}'$ simulating $\mathcal{G}(L(\mathcal{P}))$ for $\mathcal{P} = (\mathcal{S}, \Sigma_I \times \Sigma_O, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$:

- $\mathcal{S}'$ has all states of $\mathcal{S}$, plus some auxiliary ones:
  - $\{(q, a), (q, \binom{a}{b}) \mid q \in Q, a \in \Sigma_I, b \in \Sigma_O\}$ to mimic picking letters, and
  - a fresh sink state s.
- The initial state and the stack alphabet of $\mathcal{S}$ are the same as in $\mathcal{S}'$.
- The transitions of $\mathcal{S}'$ are defined to implement the picking of letters and then the deterministic simulation of $\mathcal{P}$ (might involve ε-transitions).
- Player 0 moves at states of the form $q \in Q$, Player 1 at states of the form $(q, a) \in Q \times \Sigma_I$ and, for completeness, at all other states (irrelevant, as these are "deterministic").
- Colors of states from Q are inherited from $\mathcal{P}$ (and are w.l.o.g. ≥ 2), all auxiliary states are colored by 1.

Lemma

Player O wins $\mathcal{G}(L(\mathcal{P}))$ if and only if Player 0 wins the parity game induced by $\mathcal{S}'$ from its initial vertex.

Correctness

Lemma

Player O wins $\mathcal{G}(L(\mathcal{P}))$ if and only if Player 0 wins the parity game induced by $\mathcal{S}'$ from its initial vertex.

Proof.

- Show that winning strategies can be translated from one game to the other.
- In particular, a pushdown transducer implement a winning strategy for Player 0 in the parity game induced by $\mathcal{S}'$ can be effectively turned into a pushdown transducer implement a winning strategy for Player O in $\mathcal{G}(L(\mathcal{P}))$. $\square$

Lemma

Player O wins $\mathcal{G}(L(\mathcal{P}))$ if and only if Player 0 wins the parity game induced by $\mathcal{S}'$ from its initial vertex.

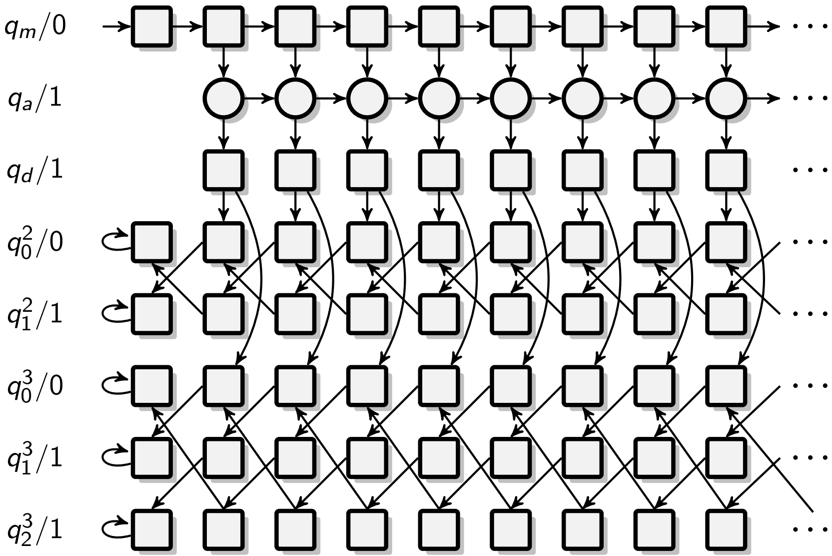
Proof.

- Show that winning strategies can be translated from one game to the other.
- In particular, a pushdown transducer implement a winning strategy for Player 0 in the parity game induced by $\mathcal{S}'$ can be effectively turned into a pushdown transducer implement a winning strategy for Player O in $\mathcal{G}(L(\mathcal{P}))$. $\square$

Corollary

If Player O wins a Gale-Stewart game with deterministic ω-contextfree winning condition, then she has a finitely representable winning strategy (and the representation is computable in exponential time).

From Parity to Gale-Stewart: Intuition



We can identify plays with infinite sequences of transitions.

(Slightly) More Formally

Fix a PDS (Q, Γ, q_I, Δ), a partition $Q = Q_0 \cup Q_1$, and a coloring $\Omega \colon Q \to \mathbb{N}$. We can assume w.l.o.g. that

- the induced game is alternating, i.e., if $(q, X, q', \gamma) \in \Delta$, then $q \in Q_0$ if and only if $q' \in Q_1$, and
- that $q_I \in Q_1$.

Both properties can be satisfied by adding transitions where necessary, while preserving the winner of the induced game.

(Slightly) More Formally

Fix a PDS (Q, Γ, q_I, Δ), a partition $Q = Q_0 \cup Q_1$, and a coloring $\Omega \colon Q \to \mathbb{N}$. We can assume w.l.o.g. that

- the induced game is alternating, i.e., if $(q, X, q', \gamma) \in \Delta$, then $q \in Q_0$ if and only if $q' \in Q_1$, and
- that $q_I \in Q_1$.

Consider the language

$$L = \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3} \cdots \in (\Delta^2)^\omega \;\middle|\; \tau_0\tau_1\tau_2\cdots \text{induces winning play for Pl. } 0 \right\}$$

$$\cup \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3} \cdots \in (\Delta^2)^\omega \;\middle|\; \begin{array}{l}\text{there is an even } n \text{ s.t. } \tau_0\cdots\tau_{n-1} \text{ induces} \\ \text{a play prefix, but } \tau_0\cdots\tau_{n-1}\tau_n \text{ does not}\end{array} \right\}.$$

(Slightly) More Formally

Fix a PDS (Q, Γ, q_I, Δ), a partition $Q = Q_0 \cup Q_1$, and a coloring $\Omega \colon Q \to \mathbb{N}$. We can assume w.l.o.g. that

- the induced game is alternating, i.e., if $(q, X, q', \gamma) \in \Delta$, then $q \in Q_0$ if and only if $q' \in Q_1$, and
- that $q_I \in Q_1$.

Consider the language

$$L = \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3}\cdots \in (\Delta^2)^\omega \;\middle|\; \tau_0\tau_1\tau_2\cdots \text{induces winning play for Pl. } 0 \right\}$$

$$\cup \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3}\cdots \in (\Delta^2)^\omega \;\middle|\; \begin{array}{l}\text{there is an even } n \text{ s.t. } \tau_0\cdots\tau_{n-1} \text{ induces} \\ \text{a play prefix, but } \tau_0\cdots\tau_{n-1}\tau_n \text{ does not}\end{array} \right\}.$$

- L is accepted by a deterministic ω-PDA.

(Slightly) More Formally

Fix a PDS (Q, Γ, q_I, Δ), a partition $Q = Q_0 \cup Q_1$, and a coloring $\Omega\colon Q \to \mathbb{N}$. We can assume w.l.o.g. that

- the induced game is alternating, i.e., if $(q, X, q', \gamma) \in \Delta$, then $q \in Q_0$ if and only if $q' \in Q_1$, and
- that $q_I \in Q_1$.

Consider the language

$$L = \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3} \cdots \in (\Delta^2)^\omega \;\middle|\; \tau_0\tau_1\tau_2 \cdots \text{induces winning play for Pl. 0} \right\}$$

$$\cup \left\{ \binom{\tau_0}{\tau_1}\binom{\tau_2}{\tau_3} \cdots \in (\Delta^2)^\omega \;\middle|\; \begin{array}{l}\text{there is an even } n \text{ s.t. } \tau_0 \cdots \tau_{n-1} \text{ induces} \\ \text{a play prefix, but } \tau_0 \cdots \tau_{n-1}\tau_n \text{ does not}\end{array} \right\}.$$

- L is accepted by a deterministic ω-PDA.
- Player 0 wins the game induced by $\mathcal{S}$ if and only if Player O wins $\mathcal{G}(L)$.

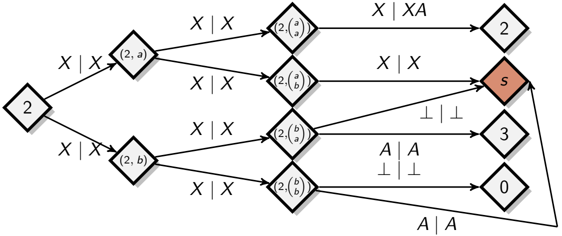
The End of the Story?

- Solving Gale-Stewart games with nondeterministic contextfree winning conditions is undecidable.
- Solving Gale-Stewart games with deterministic contextfree winning conditions is decidable.

The End of the Story?

- Solving Gale-Stewart games with nondeterministic contextfree winning conditions is undecidable.
- Solving Gale-Stewart games with deterministic contextfree winning conditions is decidable.

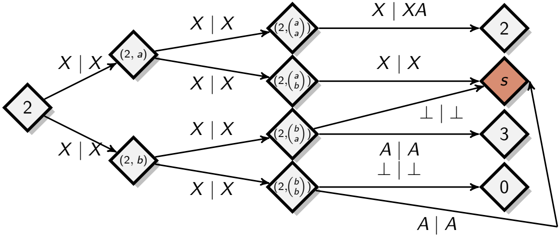
Recall the construction of the PDS $\mathcal{S}'$ simulating the Gale-Stewart game $\mathcal{G}(L(\mathcal{P}))$ for a deterministic ω-PDA $\mathcal{P}$.



The End of the Story?

- Solving Gale-Stewart games with nondeterministic contextfree winning conditions is undecidable.
- Solving Gale-Stewart games with deterministic contextfree winning conditions is decidable.

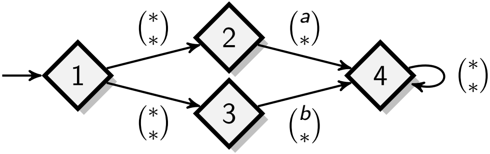
Recall the construction of the PDS $\mathcal{S}'$ simulating the Gale-Stewart game $\mathcal{G}(L(\mathcal{P}))$ for a deterministic ω-PDA $\mathcal{P}$.



- In the configuration graph of $\mathcal{S}'$, every vertex of the form $((q, \binom{a}{b}), \gamma)$ has a unique successor due to determinism of $\mathcal{P}$.
- Why not allow nondeterministic ω-PDA and let Player 0 resolve the nondeterminism (after all, she wins if the run there is an accepting run).

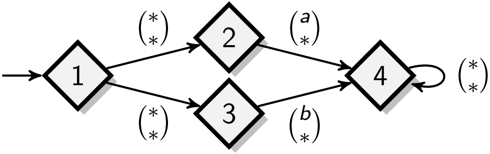
A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.



A Counterexample

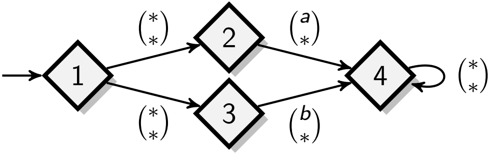
Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.



- Player O wins $\mathcal{G}(U)$.

A Counterexample

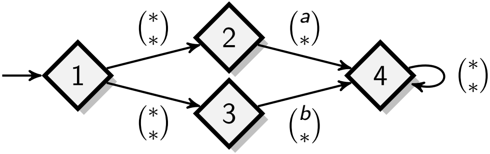
Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.



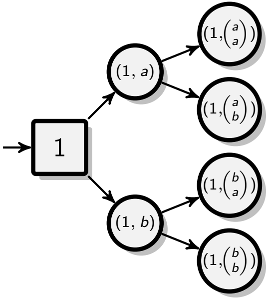
- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

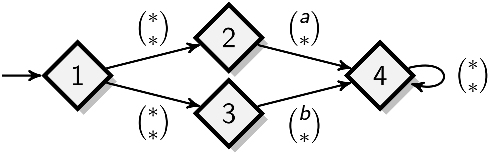


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

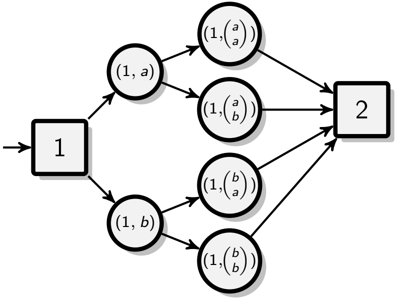


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

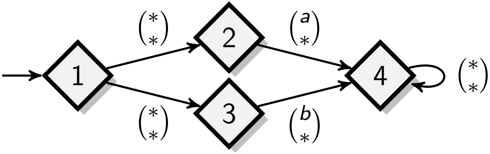


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

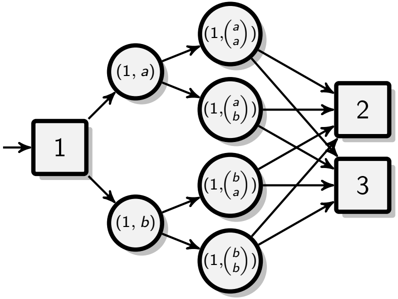


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

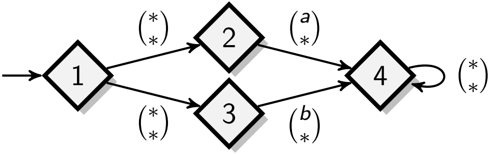


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

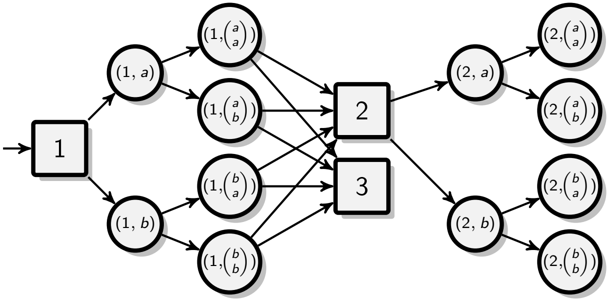


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

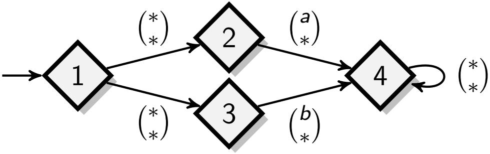


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

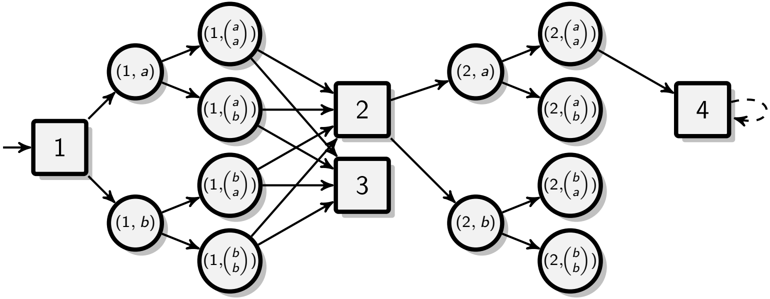


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

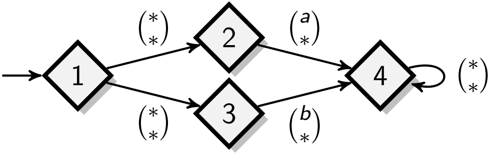


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

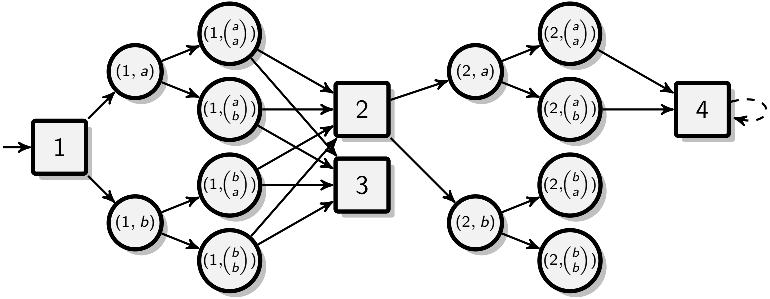


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

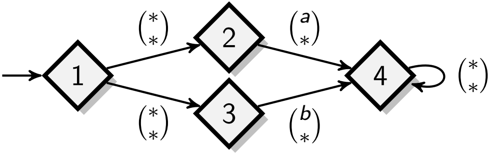


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

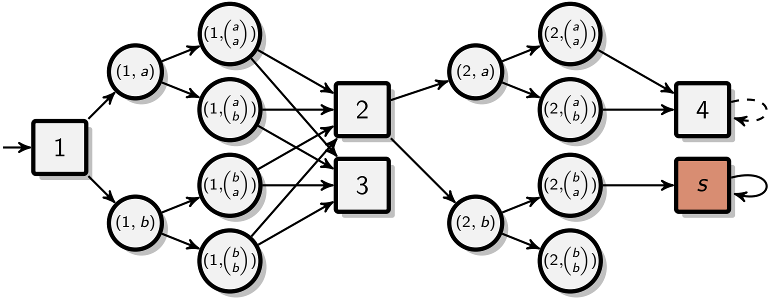


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

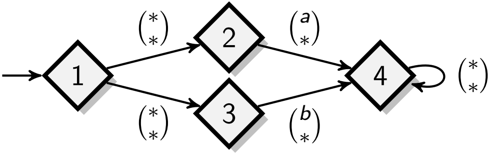


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

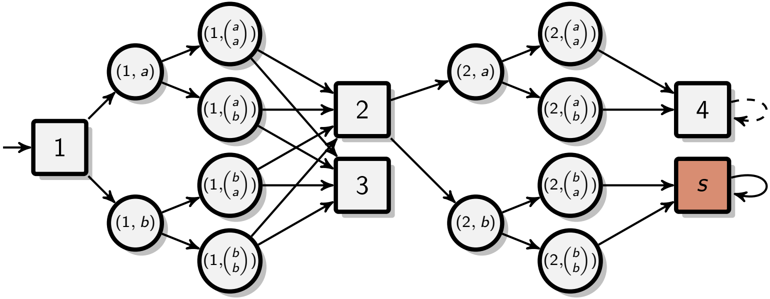


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

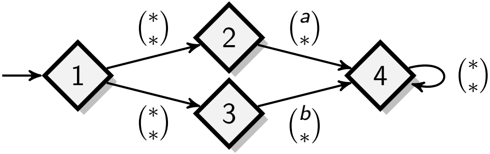


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:

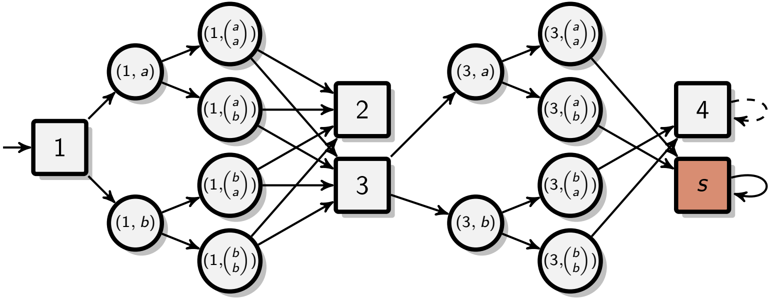


A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.

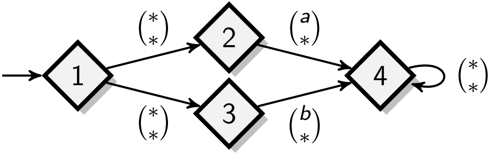


- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph:



A Counterexample

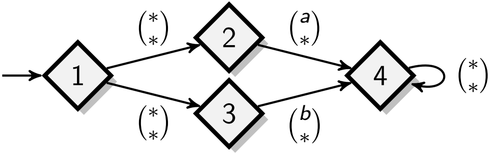
Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.



- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph: It is won by Player 1, i.e., the generalized simulation game is not correct for this automaton!

A Counterexample

Consider the following (admittedly rather contrived) automaton for the language $U = (\{a, b\}^2)^\omega$.



- Player O wins $\mathcal{G}(U)$.
- Let us study the simulation game in the configuration graph: It is won by Player 1, i.e., the generalized simulation game is not correct for this automaton!
- Still not the end of the story! Can we capture the class of automata for which the generalized simulation game is correct?

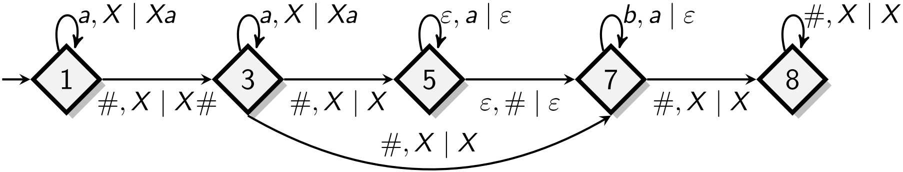
History-determinism: Intuition

An automaton is history-deterministic if the nondeterminism can always be resolved on the fly (during the simulation game).

$$L(\mathcal{P}) = \{a^i \# a^j \# b^k \#^\omega \mid k \leq i \text{ or } k \leq j)\}$$

History-determinism: Intuition

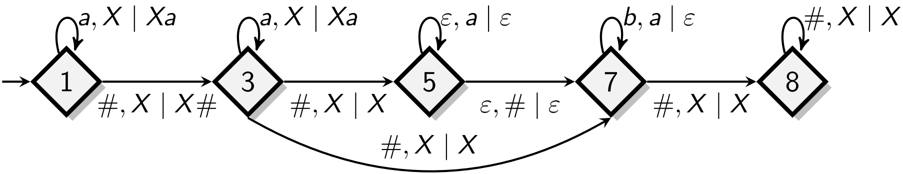
An automaton is history-deterministic if the nondeterminism can always be resolved on the fly (during the simulation game).



$$L(\mathcal{P}) = \{a^i \# a^j \# b^k \#^\omega \mid k \leq i \text{ or } k \leq j)\}$$

History-determinism: Intuition

An automaton is history-deterministic if the nondeterminism can always be resolved on the fly (during the simulation game).

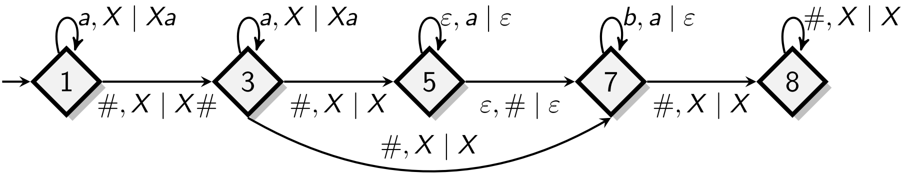


$$L(\mathcal{P}) = \{a^i \# a^j \# b^k \#^\omega \mid k \leq i \text{ or } k \leq j)\}$$

$$(1, \perp) \xrightarrow{a^i \# a^j} (3, \perp a^i \# a^j)$$

History-determinism: Intuition

An automaton is history-deterministic if the nondeterminism can always be resolved on the fly (during the simulation game).

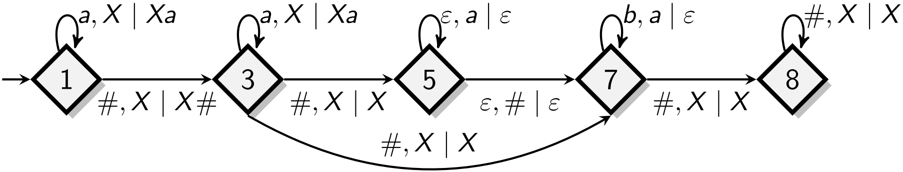


$$L(\mathcal{P}) = \{a^i \# a^j \# b^k \#^\omega \mid k \leq i \text{ or } k \leq j)\}$$

$$(1, \perp) \xrightarrow{a^i \# a^j} (3, \perp a^i \# a^j) \xrightarrow{\#} (7, \perp a^i \# a^j) \xrightarrow{b^k} (7, \perp a^j \# a^{j-k}) \xrightarrow{\#^\omega}$$

History-determinism: Intuition

An automaton is history-deterministic if the nondeterminism can always be resolved on the fly (during the simulation game).



$$L(\mathcal{P}) = \{a^i \# a^j \# b^k \#^\omega \mid k \leq i \text{ or } k \leq j)\}$$

$$\begin{tikzcd}
 & & (7, \perp a^i \# a^j) \arrow[r, "b^k"] & (7, \perp a^j \# a^{j-k}) \arrow[r, "\#^\omega"] & {} \\
(1, \perp) \arrow[r, "a^i \# a^j"] & (3, \perp a^i \# a^j) \arrow[ur, "\#"] \arrow[dr, "\#"'] & & & \\
 & & (5, \perp a^i \# a^j) \arrow[r, "\varepsilon"'] & (7, \perp a^i) \arrow[r, "b^k"'] & (7, \perp a^{i-k}) \arrow[r] & {}
\end{tikzcd}$$

History-determinism Formally

An ω-PDA $\mathcal{P} = (\mathcal{S}, \Sigma, \ell, \Omega)$ with $\mathcal{S} = (Q, \Gamma, q_I, \Delta)$ is history-deterministic, if there is a (nondeterminism) resolver for $\mathcal{P}$, a function $r \colon \Delta^* \times \Sigma \to \Delta$ such that for every $w \in L(\mathcal{P})$ the sequence $\tau_0\tau_1\tau_2 \cdots \in \Delta^\omega$ defined by

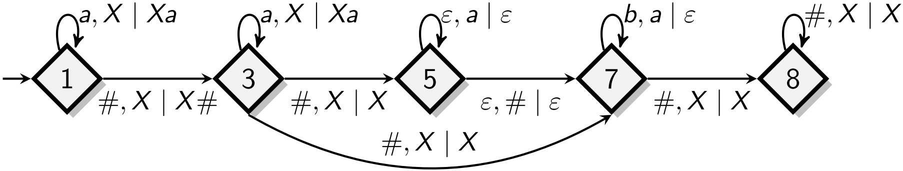
$$\tau_n = r(\tau_0 \cdots \tau_{n-1}, w(|\ell(\tau_0 \cdots \tau_{n-1})|))$$

induces an accepting run of $\mathcal{P}$ on w.

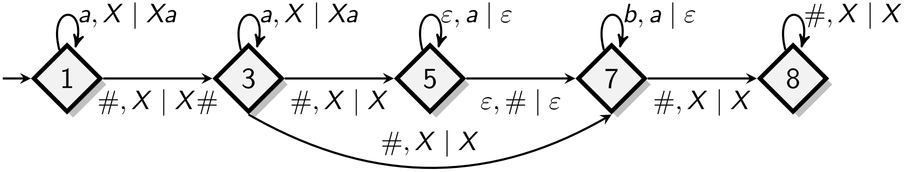
Remark

ω-DCFL $\subseteq$ ω-HD-CFL $\subseteq$ ω-CFL.

Back to the Example



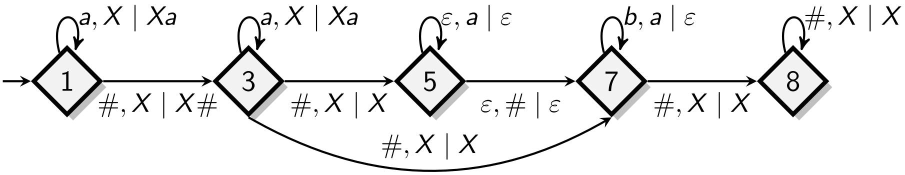
Back to the Example



A resolver for $\mathcal{P}$:

- $r((1,\_,1,\_)^i(1,\_,3,\_)(3,\_,3,\_)^j,\#) = \begin{cases}(3,\_,7,\_) & \text{if } i \leq j,\\ (3,\_,5,\_) & \text{if } i > j.\end{cases}$

Back to the Example



A resolver for $\mathcal{P}$:

- $r((1,\_,1,\_)^i(1,\_,3,\_)(3,\_,3,\_)^j, \#) = \begin{cases} (3,\_,7,\_) & \text{if } i \leq j, \\ (3,\_,5,\_) & \text{if } i > j. \end{cases}$
- For all other run prefixes and letters, there is a unique transition to extend the run to process that letter next.

Many Questions

History-deterministic (a.k.a. good-for-games) automata can often be used in contexts that typically require deterministic automata, e.g., solving games. Much effort has been put into studying history-determinism for various types of automata, e.g., ω-regular, quantitative, timed, etc (see the recent survey by Boker and Lehtinen for an introduction).

Many Questions

History-deterministic (a.k.a. good-for-games) automata can often be used in contexts that typically require deterministic automata, e.g., solving games. Much effort has been put into studying history-determinism for various types of automata, e.g., ω-regular, quantitative, timed, etc (see the recent survey by Boker and Lehtinen for an introduction).

We are interested in the following questions:

1. Are history-deterministic ω-PDA more expressive than deterministic ω-PDA?
2. Are they maybe even as expressive as ω-PDA?
3. Can games with history-deterministic ω-contextfree winning conditions be solved?
4. Can one check whether a ω-PDA is history-deterministic?
5. Closure properties?

Another Language

- Let $I = \{0, +, -\}$ and define the energy level $EL\colon I^* \to \mathbb{Z}$ of a finite word over I as

$$EL(w) = |w|_{+} - |w|_{-},$$

where $|w|_\circ$ is the number of $\circ$ in w, for $\circ \in I$.

Another Language

- Let $I = \{0, +, -\}$ and define the energy level $EL\colon I^* \to \mathbb{Z}$ of a finite word over I as

$$EL(w) = |w|_+ - |w|_-,$$

  where $|w|_\circ$ is the number of $\circ$ in w, for $\circ \in I$.
- A word $w \in I^\omega$ is safe if $EL(w(0) \cdots w(n)) \geq 0$ for every $n \geq 0$.
- A word $w \in I^\omega$ is eventually safe if it has a safe suffix.

Another Language

- Let $I = \{0, +, -\}$ and define the energy level $EL\colon I^* \to \mathbb{Z}$ of a finite word over I as

$$EL(w) = |w|_+ - |w|_-,$$

  where $|w|_\circ$ is the number of $\circ$ in w, for $\circ \in I$.
- A word $w \in I^\omega$ is safe if $EL(w(0) \cdots w(n)) \geq 0$ for every $n \geq 0$.
- A word $w \in I^\omega$ is eventually safe if it has a safe suffix.
- Let $\Sigma = I \times I$ and

$$L_{\mathrm{es}} = \left\{ \binom{w_0}{w_1} \in \Sigma^\omega \;\middle|\; \text{some } w_i \text{ is eventually safe} \right\}.$$

Another Language

- Let $I = \{0, +, -\}$ and define the energy level $EL \colon I^* \to \mathbb{Z}$ of a finite word over I as

$$EL(w) = |w|_+ - |w|_-,$$

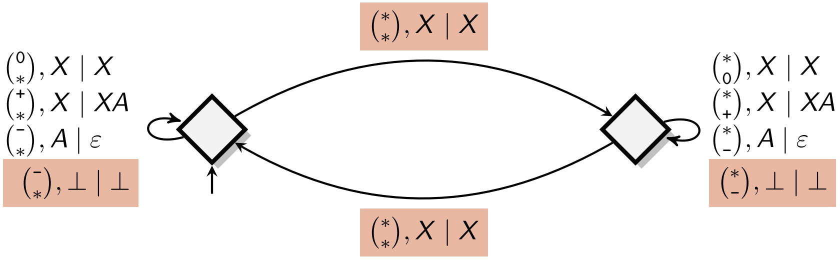
  where $|w|_\circ$ is the number of $\circ$ in w, for $\circ \in I$.
- A word $w \in I^\omega$ is safe if $EL(w(0) \cdots w(n)) \geq 0$ for every $n \geq 0$.
- A word $w \in I^\omega$ is eventually safe if it has a safe suffix.
- Let $\Sigma = I \times I$ and

$$L_{\mathrm{es}} = \left\{ \binom{w_0}{w_1} \in \Sigma^\omega \;\middle|\; \text{some } w_i \text{ is eventually safe} \right\}.$$

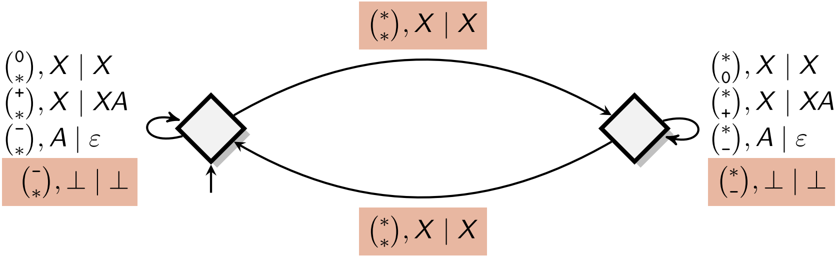
Lemma (Lehtinen, Z. 2020)

$L_{es} \in \omega\text{-}HD\text{-}CFL \setminus \omega\text{-}DCFL$.

$L_{\text{es}} \in \omega$-HD-CFL

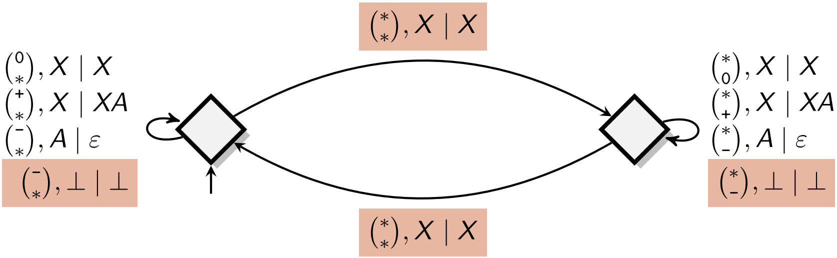


$L_{es} \in \omega$-HD-CFL



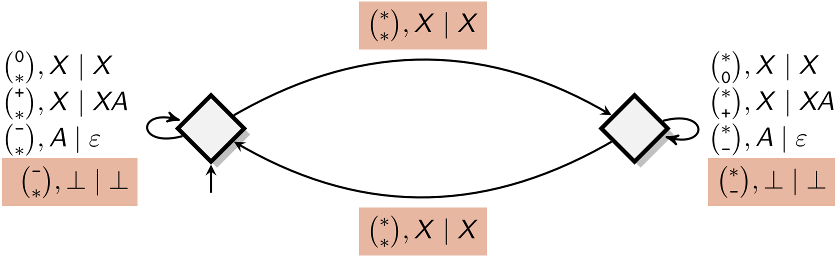
- Red transitions have color 1, all others have color 0 (can easily be turned into a state-based parity condition by adding auxiliary states).

$L_{\mathrm{es}} \in \omega$-HD-CFL



- Red transitions have color 1, all others have color 0 (can easily be turned into a state-based parity condition by adding auxiliary states).
- The ω-PDA accepts L_{es}.

$L_{es} \in \omega$-HD-CFL



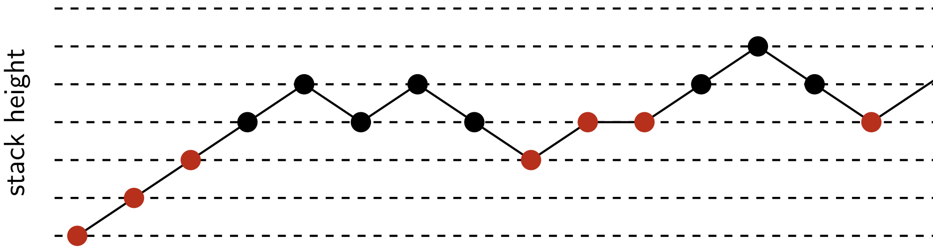
- Red transitions have color 1, all others have color 0 (can easily be turned into a state-based parity condition by adding auxiliary states).
- The ω-PDA accepts L_{es}.
- But we also need a resolver: Given $w = \binom{w_0^0}{w_0^1} \cdots \binom{w_n^0}{w_n^1}$ let m^i be the minimal m such that $w_m^i \cdots w_n^i$ is safe. Then, we define the resolver to guide the run
  - to the left state, if $m^0 \leq m^1$, and
  - to the right state otherwise.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

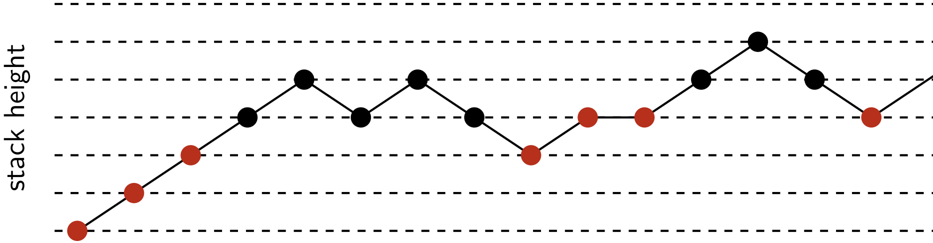
Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.



Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

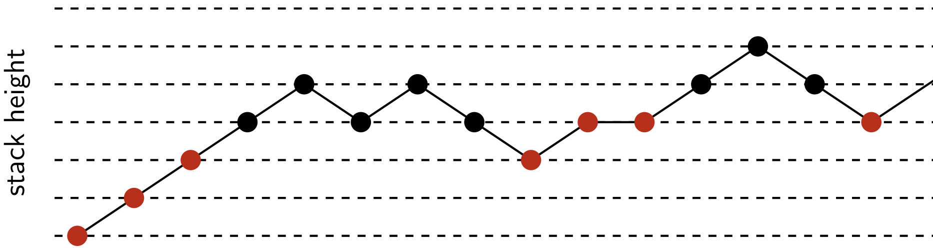


Lemma

1. Every infinite run has infinitely many steps.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

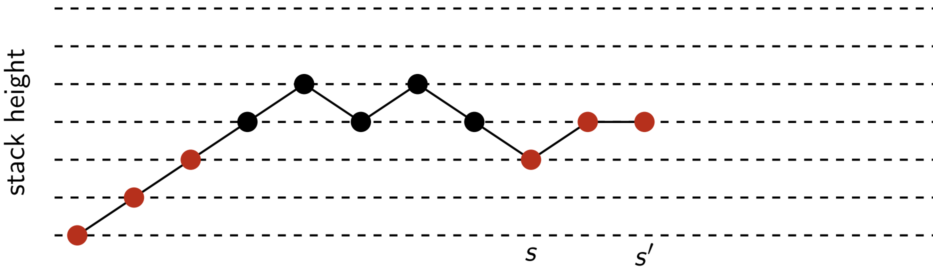


Lemma

1. Every infinite run has infinitely many steps.
2. If $s < s'$ are steps of a run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ such that c_s and $c_{s'}$ have the same state and same topmost stack symbol, then $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})$ also induces a run.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

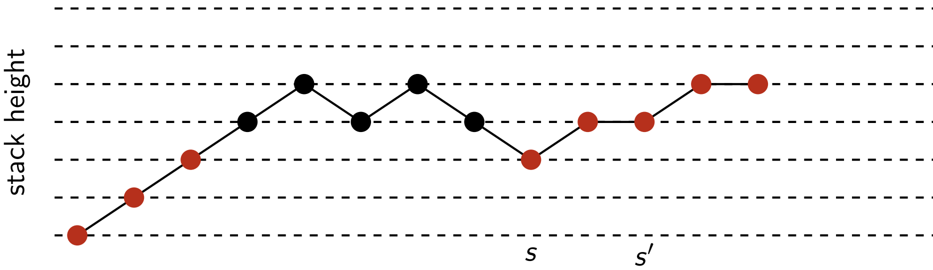


Lemma

1. Every infinite run has infinitely many steps.
2. If $s < s'$ are steps of a run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ such that c_s and $c_{s'}$ have the same state and same topmost stack symbol, then $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})$ also induces a run.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

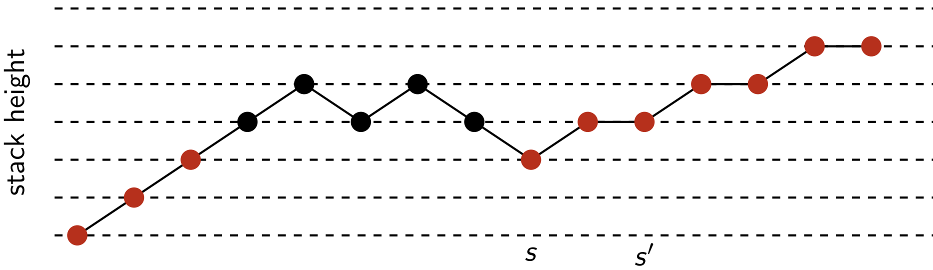


Lemma

1. Every infinite run has infinitely many steps.
2. If $s < s'$ are steps of a run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ such that c_s and $c_{s'}$ have the same state and same topmost stack symbol, then $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})$ also induces a run.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.

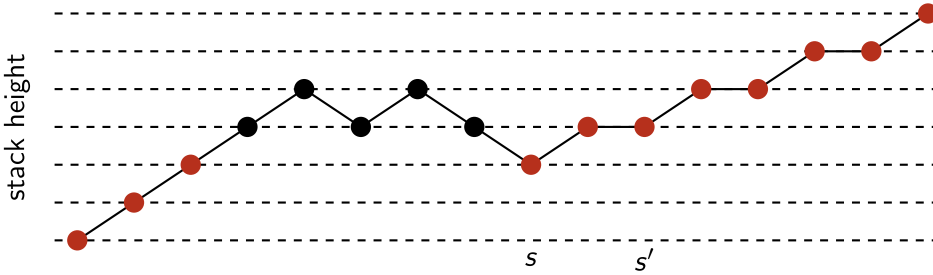


Lemma

1. Every infinite run has infinitely many steps.
2. If $s < s'$ are steps of a run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ such that c_s and $c_{s'}$ have the same state and same topmost stack symbol, then $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})$ also induces a run.

Runs Have Steps

Let $\rho = c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ be a run of an ω-PDA (i.e., a path through the configuration graph of the underlying PDS). A **step** of ρ is a position $s \in \mathbb{N}$ such that $\mathrm{sh}(c_s) \leq \mathrm{sh}(c_{s'})$ for all $s' > s$.



Lemma

1. Every infinite run has infinitely many steps.
2. If $s < s'$ are steps of a run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ such that c_s and $c_{s'}$ have the same state and same topmost stack symbol, then $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})$ also induces a run.

$L_{\mathrm{es}} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\,\cdots$.

$L_{\mathrm{es}} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\,\cdots$.
- Then (π_i denotes the projection to the i-th component),

  $$EL(\pi_0(x_0(x_1)^3\cdots(x_1)^{2^{2j}-1})) = -j$$

  for every $j > 1$ and

  $$EL(\pi_1(x_0(x_1)^3\cdots(x_0)^{2^{2j-1}-1})) = -j$$

  for every $j > 0$. This implies $w_{\overline{es}} \notin L_{\mathrm{es}}$.

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\cdots$.
- Then (π_i denotes the projection to the i-th component),

  $$EL(\pi_0(x_0(x_1)^3\cdots(x_1)^{2^{2j}-1})) = -j$$

  for every $j > 1$ and

  $$EL(\pi_1(x_0(x_1)^3\cdots(x_0)^{2^{2j-1}-1})) = -j$$

  for every $j > 0$. This implies $w_{\overline{es}} \notin L_{\mathrm{es}}$.
- As every prefix of $w_{\overline{es}}$ can be extended to a word in L_{es}, $\mathcal{P}$ has a (rejecting!) run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ processing $w_{\overline{es}}$.

$L_{\mathrm{es}} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\cdots$.
- Then (π_i denotes the projection to the i-th component),

  $$EL(\pi_0(x_0(x_1)^3\cdots(x_1)^{2^{2j}-1})) = -j$$

  for every $j > 1$ and

  $$EL(\pi_1(x_0(x_1)^3\cdots(x_0)^{2^{2j-1}-1})) = -j$$

  for every $j > 0$. This implies $w_{\overline{es}} \notin L_{\mathrm{es}}$.
- As every prefix of $w_{\overline{es}}$ can be extended to a word in L_{es}, $\mathcal{P}$ has a (rejecting!) run $c_0 \xrightarrow{\tau_0} c_1 \xrightarrow{\tau_1} c_2 \xrightarrow{\tau_2} \cdots$ processing $w_{\overline{es}}$.

$L_{\mathsf{es}} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\,\cdots$.
- This run contains two steps s and s' such that
  - **0.** both configurations have the same state and topmost stack symbol,
  - **1.** The maximal color in $c_s \cdots c_{s'}$ is odd, and
  - **2.** the sequence $\ell(\tau_s)\cdots\ell(\tau_{s'-1})$ processes an infix w of $w_{\overline{es}}$ with $EL(\pi_i(w)) > 0$, for some $i \in \{0, 1\}$.

$L_{es} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\,\cdots$.
- This run contains two steps s and s' such that
  0. both configurations have the same state and topmost stack symbol,
  1. The maximal color in $c_s \cdots c_{s'}$ is odd, and
  2. the sequence $\ell(\tau_s)\cdots\ell(\tau_{s'-1})$ processes an infix w of $w_{\overline{es}}$ with $EL(\pi_i(w)) > 0$, for some $i \in \{0,1\}$.
- The run induced by $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})^\omega$ is
  1. rejecting, but
  2. processes a word with a safe suffix in component i.

$L_{\mathsf{es}} \notin \omega$-DCFL

Towards a contradiction, assume there is a deterministic ω-PDA $\mathcal{P}$ accepting L_{es}.

- Define $x_0 = \binom{+}{0}\binom{+}{-}$ and $x_1 = \binom{0}{+}\binom{-}{+}$.
- Define $w_{\overline{es}} = x_0\,(x_1)^3\,(x_0)^7\,(x_1)^{15}\,(x_0)^{31}\,(x_1)^{63}\,\cdots$.
- This run contains two steps s and s' such that
  0. both configurations have the same state and topmost stack symbol,
  1. The maximal color in $c_s \cdots c_{s'}$ is odd, and
  2. the sequence $\ell(\tau_s)\cdots\ell(\tau_{s'-1})$ processes an infix w of $w_{\overline{es}}$ with $EL(\pi_i(w)) > 0$, for some $i \in \{0,1\}$.
- The run induced by $\tau_0 \cdots \tau_{s-1}(\tau_s \cdots \tau_{s'-1})^\omega$ is
  1. rejecting, but
  2. processes a word with a safe suffix in component i.
- Thus, contrary to our assumption, $\mathcal{P}$ does not accept L_{es}.

The Full Picture

Theorem (Lehtinen, Z. 2020)

ω-HD-CFL $\subsetneq$ ω-CFL.

Theorem (Lehtinen, Z. 2020)

ω-HD-CFL $\subsetneq$ ω-CFL.

Proof.

Show that

$$L_\# = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\} \cup \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$$

is not history-deterministic. $\square$

Theorem (Lehtinen, Z. 2020)

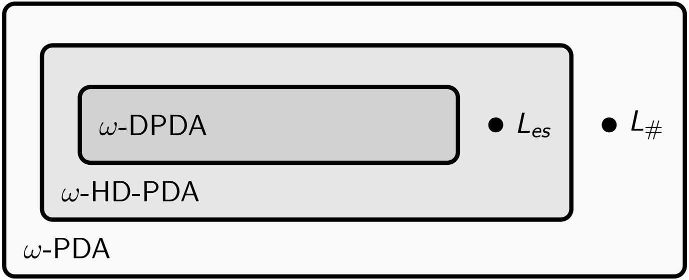
ω-HD-CFL $\subsetneq$ ω-CFL.

Proof.

Show that

$$L_\# = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\} \cup \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$$

is not history-deterministic. □



The Full Picture

Theorem (Lehtinen, Z. 2020)

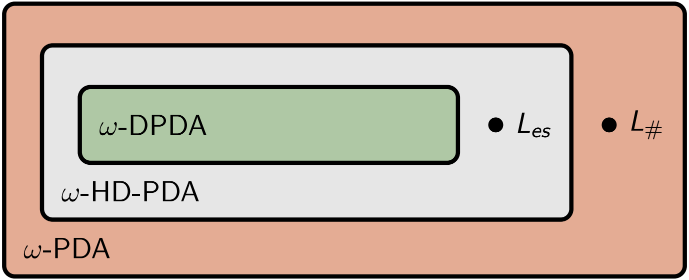
ω-HD-CFL $\subsetneq$ ω-CFL.

Proof.

Show that

$$L_\# = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\} \cup \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$$

is not history-deterministic. □



Theorem (Lehtinen, Z. 2020)

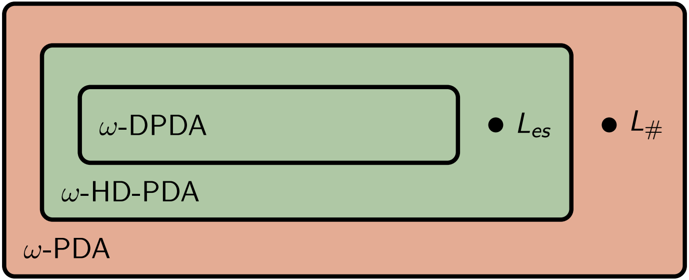
ω-HD-CFL $\subsetneq$ ω-CFL.

Proof.

Show that

$$L_{\#} = \{(a\#)^n(b\#)^n\#^\omega \mid n \geq 1\} \cup \{(a\#)^n(b\#)^{2n}\#^\omega \mid n \geq 1\}$$

is not history-deterministic. $\square$



Good News

Theorem (Lehtinen, Z. 2020)

Solving Gale-Stewart games with history-deterministic contextfree winning conditions is EXPTIME*-complete.*

Good News

Theorem (Lehtinen, Z. 2020)

Solving Gale-Stewart games with history-deterministic contextfree winning conditions is EXPTIME*-complete.*

Proof Sketch

Disclaimer: We only consider ε-free automata here (allowing ε-transition is not technically hard, but requires slightly more cumbersome notation).

Good News

Theorem (Lehtinen, Z. 2020)

Solving Gale-Stewart games with history-deterministic contextfree winning conditions is ExpTime*-complete.*

Proof Sketch

- Lower bound inherited from deterministic ω-PDA.

Good News

Theorem (Lehtinen, Z. 2020)

Solving Gale-Stewart games with history-deterministic contextfree winning conditions is EXPTIME*-complete.*

Proof Sketch

- Lower bound inherited from deterministic ω-PDA.
- For the upper bound, let $\mathcal{P} = ((Q, \Gamma, q_I, \Delta), \Sigma_I \times \Sigma_O, \ell, \Omega)$ be a HD-PDA and let L be the following language over $\Sigma_I \times \Sigma'_O$ with $\Sigma'_O = \Sigma_O \times \Delta$:
$$\left\{ \binom{\alpha_0}{(\beta_0, \tau_0)} \binom{\alpha_1}{(\beta_1, \tau_1)} \binom{\alpha_2}{(\beta_2, \tau_2)} \cdots \;\middle|\; \tau_0\tau_1\tau_2\cdots \text{ is an accepting run on } \binom{\alpha_0}{\beta_0}\binom{\alpha_1}{\beta_1}\binom{\alpha_2}{\beta_2}\cdots \right\}.$$

Good News

Theorem (Lehtinen, Z. 2020)

Solving Gale-Stewart games with history-deterministic contextfree winning conditions is $\textsc{ExpTime}$*-complete.*

Proof Sketch

- Lower bound inherited from deterministic ω-PDA.
- For the upper bound, let $\mathcal{P} = ((Q, \Gamma, q_I, \Delta), \Sigma_I \times \Sigma_O, \ell, \Omega)$ be a HD-PDA and let L be the following language over $\Sigma_I \times \Sigma'_O$ with $\Sigma'_O = \Sigma_O \times \Delta$:
$$\left\{ \binom{\alpha_0}{(\beta_0, \tau_0)} \binom{\alpha_1}{(\beta_1, \tau_1)} \binom{\alpha_2}{(\beta_2, \tau_2)} \cdots \;\middle|\; \begin{array}{l} \tau_0\tau_1\tau_2\cdots \text{ is an} \\ \text{accepting run on} \\ \binom{\alpha_0}{\beta_0}\binom{\alpha_1}{\beta_1}\binom{\alpha_2}{\beta_2}\cdots \end{array} \right\}.$$
- L is accepted by a deterministic ω-PDA of polynomial size: simulate $\mathcal{P}$ while processing input.
- Thus, we can solve $\mathcal{G}(L)$ in exponential time.

Correctness

We claim that both games have the same winner.

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{cccccc} \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{ccccc} \boxed{\alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n} \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P}))$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{tikzcd}[row sep=small, column sep=small]
\alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \arrow[d, bend left, "\sigma"] \\
\beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n \\
\tau_0 & \tau_1 & \cdots & \tau_{n-1} &
\end{tikzcd}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{llclc} \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P}))$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$
\begin{array}{ccccc}
\alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \\
\beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n \\
\tau_0 & \tau_1 & \cdots & \tau_{n-1} &
\end{array}
$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P}))$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{cccccc}
\alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n & \\
\beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n & r \\
\tau_0 & \tau_1 & \cdots & \tau_{n-1} & \tau_n & \\
 & & & r & &
\end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{ccccc} \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \tau_n \end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{cccccc} \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n & \alpha_{n+1} \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n & \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \tau_n & \end{array}$$

Correctness

We claim that both games have the same winner.

- First, let σ be a winning strategy for $\mathcal{G}(L(\mathcal{P})$ and let r be a resolver for $\mathcal{P}$.
- σ and r can be combined into a strategy σ' for $\mathcal{G}(L)$: chose letters in Σ_O according to σ and transitions according to r.

$$\begin{array}{cccccc} \alpha_0 & \alpha_1 & \cdots & \alpha_{n-1} & \alpha_n & \alpha_{n+1} \\ \beta_0 & \beta_1 & \cdots & \beta_{n-1} & \beta_n & \\ \tau_0 & \tau_1 & \cdots & \tau_{n-1} & \tau_n & \end{array}$$

- σ' is winning, as σ is winning (it guarantees that the play is in $L(\mathcal{P})$) and r is a resolver (it constructs on-the-fly an accepting run on words in $L(\mathcal{P})$).

Correctness

We claim that both games have the same winner.

- Now, let σ' be a winning strategy for $\mathcal{G}(L)$, i.e., $\sigma' \colon \Sigma_I^* \to (\Sigma_I \times \Delta)$.

Correctness

We claim that both games have the same winner.

- Now, let σ' be a winning strategy for $\mathcal{G}(L)$, i.e., $\sigma' \colon \Sigma_I^* \to (\Sigma_I \times \Delta)$.
- We define the strategy $\sigma \colon \Sigma_I^* \to \Sigma_I$ by $\sigma(w) = \beta$ for $\sigma'(w) = (\beta, \tau)$ (i.e., we just drop the second component).

Correctness

We claim that both games have the same winner.

- Now, let σ' be a winning strategy for $\mathcal{G}(L)$, i.e., $\sigma' \colon \Sigma_I^* \to (\Sigma_I \times \Delta)$.
- We define the strategy $\sigma \colon \Sigma_I^* \to \Sigma_I$ by $\sigma(w) = \beta$ for $\sigma'(w) = (\beta, \tau)$ (i.e., we just drop the second component).
- It is winning since σ' guarantees that the word over $\Sigma_I \times \Sigma_O$ constructed by the two players has an accepting run, i.e., it is in $L(\mathcal{P})$. $\square$

Correctness

We claim that both games have the same winner.

- Now, let σ' be a winning strategy for $\mathcal{G}(L)$, i.e., $\sigma' \colon \Sigma_I^* \to (\Sigma_I \times \Delta)$.
- We define the strategy $\sigma \colon \Sigma_I^* \to \Sigma_I$ by $\sigma(w) = \beta$ for $\sigma'(w) = (\beta, \tau)$ (i.e., we just drop the second component).
- It is winning since σ' guarantees that the word over $\Sigma_I \times \Sigma_O$ constructed by the two players has an accepting run, i.e., it is in $L(\mathcal{P})$. $\square$

Corollary

If Player O wins a Gale-Stewart game with history-deterministic ω-contextfree winning condition, then she has a finitely representable winning strategy (and the representation is computable in exponential time).

A Caveat

History-determinism is not a syntactic definition (unlike determinism, which is easily checkable).

A Caveat

History-determinism is not a syntactic definition (unlike determinism, which is easily checkable).

Theorem (Lehtinen, Z. 2020)

The following problems are undecidable:

1. *Given an ω-PDA $\mathcal{P}$, is $\mathcal{P}$ history-deterministic?*
2. *Given an ω-PDA $\mathcal{P}$, is $L(\mathcal{P})$ in ω-HD-CFL?*

A Caveat

History-determinism is not a syntactic definition (unlike determinism, which is easily checkable).

Theorem (Lehtinen, Z. 2020)

The following problems are undecidable:

1. *Given an ω-PDA $\mathcal{P}$, is $\mathcal{P}$ history-deterministic?*
2. *Given an ω-PDA $\mathcal{P}$, is $L(\mathcal{P})$ in ω-HD-CFL?*

Nevertheless, the reduction just sketched can be used for arbitrary (nondeterministic) ω-PDA $\mathcal{P}$:

- If Player O wins $\mathcal{G}(L)$, then she also wins $\mathcal{G}(L(\mathcal{P}))$.
- However, if she does not win $\mathcal{G}(L)$, then $\mathcal{G}(L(\mathcal{P}))$ is
  1. either won by Player I, or
  2. it is won by Player O and $\mathcal{P}$ is not history-deterministic.

Some Open Problems

- Is there a class of grammars that captures the history-deterministic contextfree languages?
- Can the history-deterministic contextfree languages be captured by some fragment of Second-order Logic?

Some Open Problems

- Is there a class of grammars that captures the history-deterministic contextfree languages?
- Can the history-deterministic contextfree languages be captured by some fragment of Second-order Logic?
- What kind of "machines" are required to implement resolvers for history-deterministic (ω-) PDA? It is known that pushdown transducer are not sufficient!

Some Open Problems

- Is there a class of grammars that captures the history-deterministic contextfree languages?
- Can the history-deterministic contextfree languages be captured by some fragment of Second-order Logic?
- What kind of “machines” are required to implement resolvers for history-deterministic (ω-) PDA? It is known that pushdown transducer are not sufficient!
- Equivalence of deterministic PDA over finite words is decidable. What about equivalence of history-deterministic PDA over finite words?

Some Open Problems

- Is there a class of grammars that captures the history-deterministic contextfree languages?
- Can the history-deterministic contextfree languages be captured by some fragment of Second-order Logic?
- What kind of "machines" are required to implement resolvers for history-deterministic (ω-) PDA? It is known that pushdown transducer are not sufficient!
- Equivalence of deterministic PDA over finite words is decidable. What about equivalence of history-deterministic PDA over finite words?
- There is an uncomputable succinctness gap between deterministic and nondeterministic PDA. Where do history-deterministic PDA lie in this gap? So far, only (doubly-) exponential gaps are known.
- Many more.

A (Biased and Incomplete) List of References

- Igor Walukiewicz: “Pushdown Processes: Games and Model-Checking”. Information and Computation 164(2), 2001
- Orna Kupferman, Moshe Y. Vardi: “An Automata-Theoretic Approach to Reasoning about Infinite-State Systems”. CAV 2000

A (Biased and Incomplete) List of References

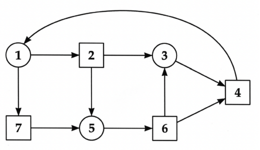
- Igor Walukiewicz: "Pushdown Processes: Games and Model-Checking". Information and Computation 164(2), 2001
- Orna Kupferman, Moshe Y. Vardi: "An Automata-Theoretic Approach to Reasoning about Infinite-State Systems". CAV 2000
- Karoliina Lehtinen, Martin Zimmermann: "Good-for-games ω-Pushdown Automata". Logical Methods in Computer Science 18(1), 2022
- Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, Martin Zimmermann: "A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct". MFCS 2021

A (Biased and Incomplete) List of References

- Igor Walukiewicz: "Pushdown Processes: Games and Model-Checking". Information and Computation 164(2), 2001
- Orna Kupferman, Moshe Y. Vardi: "An Automata-Theoretic Approach to Reasoning about Infinite-State Systems". CAV 2000
- Karoliina Lehtinen, Martin Zimmermann: "Good-for-games ω-Pushdown Automata". Logical Methods in Computer Science 18(1), 2022
- Shibashis Guha, Ismaël Jecker, Karoliina Lehtinen, Martin Zimmermann: "A Bit of Nondeterminism Makes Pushdown Automata Expressive and Succinct". MFCS 2021
- Nathanaël Fijalkow et al.: "Games on Graphs". arXiv:2305.10546
- Felix Klein, Alexander Weinert, Martin Zimmermann: "Lecture Notes Infinite Games". `homes.cs.aau.dk/ mzi/`

The Really Big Picture

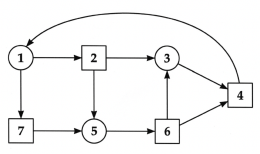
The Really Big Picture



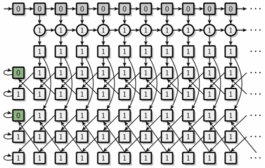
Yesterday:
finite graphs

∞

The Really Big Picture



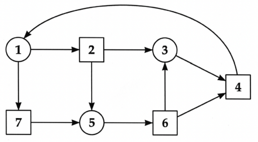
Yesterday:
finite graphs



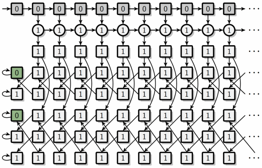
Here:
countable graphs

∞

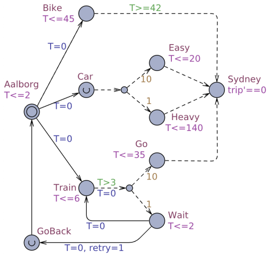
The Really Big Picture



Yesterday:
finite graphs



Here:
countable graphs



This afternoon:
uncountable graphs

∞