history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé and on numerous external references...
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé
and on numerous external references...
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé
and on numerous external references...
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé and on numerous external references...

On Succinctness and Recognisability of Alternating Good-for-Games Automata

Udi Boker
Interdisciplinary Center (IDC) Herzliya, Israel
uboker@gmail.com

Denis Kuperberg
CNRS, UJF, Centre National Supérieur, Lyon, France
denis.kuperberg@ens-lyon.fr

Karolina Lehtimä
University of Liverpool, United Kingdom
k.lehtima@liv.ac.uk

Michał Skrzypczak
Institute of Informatics, University of Warsaw, Poland
mskrzyp@mini.uw.edu.pl

On Determinisation of Generalised Automata

Denis Kuperberg1,2,3 and Michał Skrzypczak
1 IST, IST Austria, Austria
2 INRIA, CNRS, UJF, Grenoble, France
3 Aix-Marseille Université, CNRS, Centrale Marseille, France

Abstract

We study alternating good-for-games (GFG) automata, i.e., alternating automata whose conjunctive and disjunctive choices can be resolved in an online manner, without knowledge of the input word still to be read. We show that they can be

determined in non-deterministic polynomial time.
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, Hervé and on numerous external references...
history-deterministic automata

Nicolas Markey
CNRS – IRISA (Univ. Rennes, France)

based on joint discussions with Uli, Sophie, Dylan, and on numerous external references...
Automata on infinite words and trees

Example (Büchi word automata)

![Diagram of a Büchi word automaton with states q₀, q₁, and q₂, transitions labeled a, b, and c, and accepting states marked with ⋆.]
Automata on infinite words and trees

Example (Büchi word automata)

How to extend automata to work on (infinite) trees?
Automata on infinite words and trees

Example (Büchi word automata)

How to extend automata to work on (infinite) trees?
Automata on infinite words and trees

Example (Büchi word automata)
Automata on infinite words and trees

Example (Büchi word automata)

How to extend automata to work on (infinite) trees?
Automata on infinite words and trees

Example (Büchi word automata)

How to extend automata to work on (infinite) trees?
Automata on infinite words and trees

Example (Büchi word automata)

How to extend automata to work on (infinite) trees?
Automata on infinite words and trees

Example (Büchi word automata)
Automata on infinite words and trees

Example (Büchi tree automata)

\[
\begin{align*}
\delta(q_0, a) &= (q_0, q_1) \\
\delta(q_0, b) &= (q_1, q_1) \\
\delta(q_0, c) &= (q_2, q_2) \\
\delta(q_1, ∗) &= (q_1, q_1) \\
\delta(q_2, ∗) &= (q_2, q_2)
\end{align*}
\]
Automata on infinite words and trees

Example (Büchi tree automata)

\[ \delta(q_0, a) = (q_0, q_1) \]
\[ \delta(q_0, b) = (q_1, q_1) \]
\[ \delta(q_0, c) = (q_2, q_2) \]
\[ \delta(q_1, \star) = (q_1, q_1) \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
Automata on infinite words and trees

Example (Büchi tree automata)

\[ \delta(q_0, a) = (q_0, q_1) \]
\[ \delta(q_0, b) = (q_1, q_1) \]
\[ \delta(q_0, c) = (q_2, q_2) \]
\[ \delta(q_1, \star) = (q_1, q_1) \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
Automata on infinite words and trees

Example (Büchi tree automata)

\[ \delta(q_0, a) = (q_0, q_1) \]
\[ \delta(q_0, b) = (q_1, q_1) \]
\[ \delta(q_0, c) = (q_2, q_2) \]
\[ \delta(q_1, \star) = (q_1, q_1) \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
Automata on infinite words and trees

Example (Büchi tree automata)

\[
\begin{align*}
\delta(q_0, a) &= (q_0, q_1) \\
\delta(q_0, b) &= (q_1, q_1) \\
\delta(q_0, c) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]
Automata on infinite words and trees

Example (Büchi tree automata)

\[ \delta(q_0, a) = (q_0, q_1) \]
\[ \delta(q_0, b) = (q_1, q_1) \]
\[ \delta(q_0, c) = (q_2, q_2) \]
\[ \delta(q_1, \ast) = (q_1, q_1) \]
\[ \delta(q_2, \ast) = (q_2, q_2) \]
Automata on infinite words and trees

Example (Büchi tree automata)

\[ \delta(q_0, a) = (q_0, q_1) \]
\[ \delta(q_0, b) = (q_1, q_1) \]
\[ \delta(q_0, c) = (q_2, q_2) \]
\[ \delta(q_1, \star) = (q_1, q_1) \]
\[ \delta(q_2, \star) = (q_2, q_2) \]
Automata on infinite words and trees

Example (Büchi tree automata)

\[
\begin{align*}
\delta(q_0, a) &= (q_0, q_1) \\
\delta(q_0, b) &= (q_1, q_1) \\
\delta(q_0, c) &= (q_2, q_2) \\
\delta(q_1, \star) &= (q_1, q_1) \\
\delta(q_2, \star) &= (q_2, q_2)
\end{align*}
\]
Deterministic vs. non-deterministic Büchi automata

Example

- States: $q_0$, $q_1$
- Transitions:
  - From $q_0$ to $q_0$ on $0$ and $1$
  - From $q_1$ to $q_1$ on $0$
- Acceptance condition:
  - $q_0$ accepts all infinite words having finitely many 1s

This automaton demonstrates the difference in acceptance for deterministic versus non-deterministic Büchi automata.
Deterministic vs. non-deterministic Büchi automata

Example

The automaton accepts all infinite words having finitely many 1s.

No deterministic Büchi automata accept this language:

0 0 0 0 0 0 0 0 0 ...
Deterministic vs. non-deterministic Büchi automata

Example

\[ \begin{array}{c}
q_0 \quad 0 \rightarrow q_1 \\
0,1 \quad 0 \\
\end{array} \]

accepts all infinite words having finitely many 1s

No deterministic Büchi automata accept this language

\[ \begin{array}{c}
0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad \ldots \\
\end{array} \]

is accepted
Deterministic vs. non-deterministic Büchi automata

Example

The automaton $\mathcal{A} = (\{q_0, q_1\}, \{0,1\}, \delta, q_0, \{q_1\})$ accepts all infinite words having finitely many 1s.

No deterministic Büchi automata accept this language

The infinite words $0,0,0,0,0,0,0,0,\ldots$ and $0,0,0,0,0,1,0,0,0,0,0,0,0,\ldots$ are not accepted by a deterministic Büchi automaton.
Deterministic vs. non-deterministic Büchi automata

Example

The automaton accepts all infinite words having finitely many 1s.

No deterministic Büchi automata accept this language:

- The first sequence of 0s is accepted.
- The second sequence, which alternates 0s and 1s, is also accepted.
Deterministic vs. non-deterministic Büchi automata

Example

\[ q_0 \xrightarrow{0} q_1 \]

accepts all infinite words having finitely many 1s

No deterministic Büchi automata accept this language

- \[ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \ldots \]
  - is accepted
- \[ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\ldots \]
  - is accepted
- \[ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \ldots \]
  - is accepted
Deterministic vs. non-deterministic Büchi automata

Example

\[ \begin{array}{c}
q_0 & \xrightarrow{0,1} & q_1 \\
\downarrow & & \uparrow \\
0 & & 0
\end{array} \]

accepts all infinite words having finitely many 1s

No deterministic Büchi automata accept this language

\[
\begin{align*}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots
\end{align*}
\]

is accepted

is accepted

is accepted
**Derived language [KSV06]**

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

---


Derived language [KSV06]

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

If $L$ is accepted by $B$, is $\Delta L$ accepted by some tree automaton $C$?

From word to tree automata

**Derived language [KSV06]**

Given \( L \subseteq \Sigma^\omega \), we let \( \Delta L \) be the set of trees all of whose infinite branches are labelled with words in \( L \).

If \( L \) is accepted by \( B \), is \( \Delta L \) accepted by some tree automaton \( C \)?

**Natural construction for deterministic automata**

If \( \delta_B(q, \sigma) = q' \), we let \( \delta_C(q, \sigma) = (q', q') \).

✓


Derived language [KSV06]

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

If $L$ is accepted by $B$, is $\Delta L$ accepted by some tree automaton $C$?

Natural construction for non-deterministic automata

We let $\delta_C(q, \sigma) = \{(q', q'') \mid q', q'' \in \delta_B(q, \sigma)\}$.

From word to tree automata

**Derived language [KSV06]**

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

If $L$ is accepted by $B$, is $\Delta L$ accepted by some tree automaton $C$?

**Natural construction for non-deterministic automata**

We let $\delta_C(q, \sigma) = \{(q', q'') \mid q', q'' \in \delta_B(q, \sigma)\}$.


From word to tree automata

**Derived language [KSV06]**

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

If $L$ is accepted by $B$, is $\Delta L$ accepted by some tree automaton $C$?

**Natural construction for non-deterministic automata**

We let $\delta_C(q, \sigma) = \{(q', q'') \mid q', q'' \in \delta_B(q, \sigma)\}$.

---


### Derived language [KSV06]

Given $L \subseteq \Sigma^\omega$, we let $\Delta L$ be the set of trees all of whose infinite branches are labelled with words in $L$.

If $L$ is accepted by $B$, is $\Delta L$ accepted by some tree automaton $C$?

### Natural construction for non-deterministic automata

We let $\delta_C(q, \sigma) = \{(q', q'') \mid q', q'' \in \delta_B(q, \sigma)\}$.

### Good-for-trees automata [BKKS13]

A word automaton $B$ accepting some language $L$ is **good-for-trees** if the associated tree automaton $C$ accepts $\Delta L$.

---


Reactive synthesis

Environment

Controller

Controller wins if, and only if, 

\[ i_1 \circ_1 i_2 \circ_2 i_3 \circ_3 \ldots \in L \]

Reactive synthesis

\[ i_1 = a \]

Controller

\[ \alpha, \beta \]

Environment

\[ \in \mathbb{L} \]

Reactive synthesis

Environment
Controller

\[ i_1 = a \]
\[ o_1 = \alpha \]

Controller wins if, and only if, \[ i_1 o_1 i_2 o_2 i_3 o_3 \ldots \in L \]

Reactive synthesis

Environment
Controller

\[ i_1 = a \quad i_2 = b \]
\[ o_1 = \alpha \]

Controller wins if, and only if, \( i_1 o_1 i_2 o_2 \ldots \in L \) [HP06] Henzinger, Piterman. Solving Games Without Determinization. In CSL 2006, p. 395-410. Springer, 2006.
Reactive synthesis

Environment

Controller

\[ i_1 = a \quad i_2 = b \]
\[ o_1 = \alpha \quad o_2 = \alpha \]

\[ i_3 = b \quad o_3 = \beta \]

... 

Controller wins if, and only if, 

Reactive synthesis

<table>
<thead>
<tr>
<th>Environment</th>
<th>$i_1 = a$</th>
<th>$i_2 = b$</th>
<th>$i_3 = b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>$o_1 = \alpha$</td>
<td>$o_2 = \alpha$</td>
<td></td>
</tr>
</tbody>
</table>

Controller wins if, and only if, $i_1 o_1 i_2 o_2 i_3 o_3 \ldots \in L$

Reactive synthesis

Environment

Controller

\[ i_1 = a \quad i_2 = b \quad i_3 = b \]
\[ o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \]

Controller wins if, and only if, \( i_1 \circ o_1 \circ i_2 \circ o_2 \circ i_3 \circ o_3 \ldots \in L \).

Reactive synthesis

Environment  \( i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \)

Controller  \( o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \)

Controller wins if, and only if, \( i_1 \cdot o_1 \cdot i_2 \cdot o_2 \cdot i_3 \cdot o_3 \cdot \ldots \in \mathcal{L}_{HP06} \)

Reactive synthesis

Environment

Controller

\[ i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \]

\[ o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \]

Controller wins if, and only if,

\[ i_1 o_1 i_2 o_2 i_3 o_3 \ldots \in L \]

\[ \text{[HP06]} \quad \text{Henzinger, Piterman. Solving Games Without Determinization. In CSL 2006, p. 395-410. Springer, 2006.} \]
Reactive synthesis

Environment: \( i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \)

Controller: \( o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \)

Controller wins if, and only if, \( i_1 \quad o_1 \quad i_2 \quad o_2 \quad i_3 \quad o_3 \quad \ldots \in L \)

G

\[ E \quad a, b \quad C \]

\[ \alpha, \beta \]
Reactive synthesis

Environment

Controller

i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots

\quad o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ \ldots \in L \)

Example

if \( L \) is defined by a deterministic automaton \( D \):

\[
\begin{array}{l}
(a, \alpha, \beta) \\
(a, b, \alpha)
\end{array}
\]

\[
\begin{array}{l}
q_0 \\
q_1
\end{array}
\]

(\( G(b \Rightarrow F \beta) \))
Reactive synthesis

Environment: \( i_1 = a \), \( i_2 = b \), \( i_3 = b \), ...  
Controller: \( o_1 = \alpha \), \( o_2 = \alpha \), \( o_3 = \beta \), ...  

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ ... \in L \)

Example

If \( L \) is defined by a deterministic automaton \( \mathcal{D} \):  
\[ (G(b \Rightarrow F \beta)) \]

- take product of \( \mathcal{D} \) and \( G \)
- solve the resulting game (in polynomial time).
Reactive synthesis

Environment

\[ i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \]

Controller

\[ o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \]

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ \ldots \in L \)

Example

if \( L \) is defined by a deterministic automaton \( D \):

\[ (G(b \Rightarrow F \beta)) \]

- take product of \( D \) and \( G \)
- solve the resulting game (in polynomial time).

\[ a, b \rightarrow \]

\[ \alpha, \beta \rightarrow \]
Reactive synthesis

Environment

Controller

Controller wins if, and only if, \( i_1 o_1 i_2 o_2 i_3 o_3 \ldots \in L \)

Example

if \( L \) is defined by a deterministic automaton \( D \):

\[ (G(b \Rightarrow F \beta)) \]

- take product of \( D \) and \( G \)
- solve the resulting game (in polynomial time).

Reactive synthesis

Environment
\[ i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \]

Controller\n\[ o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \]

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ \ldots \in L \)

Example
if \( L \) is defined by non-deterministic automaton \( \mathcal{N} \):

\[ \mathcal{G} \mathcal{F} b \iff \mathcal{G} \mathcal{F} \beta \]
Reactive synthesis

Environment

\[ i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots \]

Controller

\[ o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots \]

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ \ldots \in L \)

Example

if \( L \) is defined by non-deterministic automaton \( \mathcal{N} \):

\[ \langle G \ F \ b \ \iff \ G \ F \ \beta \rangle \]
Reactive synthesis

Environment

Controller

Controller wins if, and only if, $i_1 o_1 i_2 o_2 i_3 o_3 \ldots \in L$

Example

if $L$ is defined by non-deterministic automaton $\mathcal{N}$:

$\mathcal{G} \xrightarrow{a, b} \mathcal{C}$

$\mathcal{G} F b \iff \mathcal{G} F \beta$
Reactive synthesis

Environment

Controller

\[
i_1 = a \quad i_2 = b \quad i_3 = b \quad \ldots
\]
\[
o_1 = \alpha \quad o_2 = \alpha \quad o_3 = \beta \quad \ldots
\]

Controller wins if, and only if, \( i_1 \ o_1 \ i_2 \ o_2 \ i_3 \ o_3 \ \ldots \in L \)

**Good-for-games automata [HP06]**

An automaton \( \mathcal{A} \) on alphabet \( \Sigma \) is good-for-games if for any two-player zero-sum game \( \mathcal{G} \) with \( \Sigma \)-labelled transitions and winning condition \( \mathcal{L}(\mathcal{A}) \), the games \( \mathcal{G} \) and \( \mathcal{G} \times \mathcal{A} \) have the same winner.

History-deterministic automata

Letter game (aka. monitor game)

The letter game on a word automaton $\mathcal{A}$ is a 2-player game played as follows:

- initially, a token is placed on the (unique, wlog) initial state of $\mathcal{A}$;
- iteratively, and ad infinitum:
  - Adam proposes a letter $\sigma$ in $\Sigma$;
  - Eve moves the token along a $\sigma$-transition in $\mathcal{A}$.

In this process, Adam defines a word $w$, and Eve builds an run $\pi$ of $\mathcal{A}$ on $w$.

Adam wins if $w$ is accepted by $\mathcal{A}$ and $\pi$ is not an accepting run; Eve wins othw.
History-deterministic automata

Letter game (aka. monitor game)

The letter game on a word automaton $\mathcal{A}$ is a 2-player game played as follows:
- initially, a token is placed on the (unique, wlog) initial state of $\mathcal{A}$;
- iteratively, and ad infinitum:
  - Adam proposes a letter $\sigma$ in $\Sigma$;
  - Eve moves the token along a $\sigma$-transition in $\mathcal{A}$.

In this process, Adam defines a word $w$, and Eve builds an run $\pi$ of $\mathcal{A}$ on $w$.
- Adam wins if $w$ is accepted by $\mathcal{A}$ and $\pi$ is not an accepting run; Eve wins othw.

History-deterministic automata [HP06,Col09]

Automaton $\mathcal{A}$ is history-deterministic if Eve has a winning strategy (called resolver) in the letter game.

History-deterministic automata

Example

This co-Büchi automaton accepting \( \{ w \mid |w|_b < \infty \} \) is history-deterministic:

\[
\begin{array}{c}
q_0 \xrightarrow{a,b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1 \xrightarrow{a} q_1
\end{array}
\]

History-deterministic automata

Example

This co-Büchi automaton accepting \{w \mid |w|_b < \infty\} is history-deterministic:

Resolver: always go to $q_1$ when reading $a$ in $q_0$. 
**History-deterministic automata**

**Example**

This co-Büchi automaton accepting \( \{ w \mid |w|_b < \infty \} \) is history-deterministic:

![Automaton Diagram]

Resolver: always go to \( q_1 \) when reading \( a \) in \( q_0 \).

**Determinizable-by-pruning automata [AKL10]**

An automaton \( A \) is determinizable-by-pruning if a language-equivalent deterministic automaton can be obtained from \( A \) by removing some transitions.

**History-deterministic automata**

**Example**

This co-Büchi automaton accepting \( \{ w \mid |w|_b < \infty \} \) is history-deterministic:

```
q_0 \xrightarrow{a,b} q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_1
```

Resolver: always go to \( q_1 \) when reading \( a \) in \( q_0 \).

**Determinizable-by-pruning automata [AKL10]**

An automaton \( A \) is determinizable-by-pruning if a language-equivalent deterministic automaton can be obtained from \( A \) by removing some transitions.

Obviously, determinizability-by-pruning implies history-determinism.

Example

This Büchi automaton accepting $\{w \mid |w|_b < \infty\}$ is not history-deterministic:

Adam plays letter $a$ as long as the token is in $q_0$; if Eve never moves to $q_1$, she loses; otherwise, from $q_1$, Adam proposes the suffix $ba^\omega$. 
History-deterministic automata

Example

This Büchi automaton accepting \( \{ w \mid |w|_b < \infty \} \) is not history-deterministic:

Eve does not win the letter game:
- Adam plays letter \( a \) as long as the token is in \( q_0 \);
- If Eve never move to \( q_1 \), she looses; otherwise, from \( q_1 \), Adam propose suffix \( ba^\omega \).
Results for $\omega$-automata

**Theorem ([HP06,BKKS13,BL19])**

*Good-for-treeness, good-for-gameness and history-determinism are equivalent.*

*Determinizability-by-pruning is stronger.*


Results for $\omega$-automata

**Theorem ([HP06,BKKS13,BL19])**

*Good-for-treeness, good-for-gameness and history-determinism are equivalent. Determinizability-by-pruning is stronger.*

**Proof**

- History-determinism implies good-for-gameness and game-for-treeness: the resolver can be used to resolve non-determinism.
- The other direction is a bit more involved.
- That determinizability-by-pruning is stronger is shown with the next example.

History-deterministic automata

Example

This Büchi automaton accepting \[ ((aaa + aba)^* \cdot (aaa aaa + aba aba))^{\omega} \] is history-deterministic, but not determinizable-by-pruning:

History-deterministic automata

Example

This Büchi automaton accepting \([(aaa + aba)^* \cdot (aaa aaa + aba aba)]^\omega\) is history-deterministic, but not determinizable-by-pruning:

Resolver:
In \(q_0\), go to \(q_1\) if we just read \(aaa\), otherwise, go to \(q_2\).

Deciding history-determinism

Example (Hardness)

If $A$ is deterministic, the automaton below is history-deterministic (and determinizable-by-pruning) if, and only if, $L(B) \subseteq L(A)$.

Deciding history-determinism: $k$-token games

$k$-token games

Given an automaton $\mathcal{A}$, the $k$-token game $G_k(\mathcal{A})$ on $\mathcal{A}$ runs as follows:

- initially, $k$ tokens of Adam and one token of Eve are on the initial state of $\mathcal{A}$;
- repeatedly, *ad infinitum*:
  - Adam proposes a letter $\sigma \in \Sigma$;
  - Eve moves her token along a $\sigma$-transition;
  - Adam moves his $k$ tokens along $\sigma$-transitions.
- Eve wins if either Adam fails to produce an accepting run, or if she produces an accepting run.

Deciding history-determinism: $k$-token games

**Lemma**

If $A$ is history-deterministic, then Eve wins $G_k(A)$.

Deciding history-determinism: $k$-token games

Lemma

\textit{If } $A$ \textit{is history-deterministic, then Eve wins } $G_k(A)$.

Lemma

\textit{There exists a non-history-deterministic automaton } $B$ \textit{for which Eve wins } $G_1(B)$.

Deciding history-determinism: $k$-token games

**Lemma**

If $A$ is history-deterministic, then Eve wins $G_k(A)$.

**Lemma**

There exists a non-history-deterministic automaton $B$ for which Eve wins $G_1(B)$.

**Proof**

Adam wins the letter game: play $a$ until Eve moves to $q$; then play $ba^\omega$;

Eve wins the 1-token game: follow Adam’s token.

Deciding history-determinism: $k$-token games

**Lemma**

$Eve$ wins $G_2(A)$ if, and only if, she wins $G_k(A)$ for all $k \geq 2$.


Deciding history-determinism: $k$-token games

**Lemma**

*Eve wins $G_2(\mathcal{A})$ if, and only if, she wins $G_k(\mathcal{A})$ for all $k \geq 2$.***

**Proof**

By induction on $k$:

$G_k(\mathcal{A})$  $G_{k+1}(\mathcal{A})$

$G_2(\mathcal{A})$  $G_k(\mathcal{A})$


Deciding history-determinism: $k$-token games

**Lemma**

Eve wins $G_2(\mathcal{A})$ if, and only if, she wins $G_k(\mathcal{A})$ for all $k \geq 2$.

**Proof**

By induction on $k$:

- $G_2(\mathcal{A})$
- $G_k(\mathcal{A})$
- $G_{k+1}(\mathcal{A})$


Deciding history-determinism: \( k \)-token games

**Lemma**

Eve wins \( G_2(\mathcal{A}) \) if, and only if, she wins \( G_k(\mathcal{A}) \) for all \( k \geq 2 \).

**Proof**

By induction on \( k \):

\[ G_k(\mathcal{A}) \rightarrow \tau_k \rightarrow G_{k+1}(\mathcal{A}) \]


Deciding history-determinism: $k$-token games

**Lemma**

$Eve$ wins $G_2(A)$ if, and only if, she wins $G_k(A)$ for all $k \geq 2$.

**Proof**

By induction on $k$:

1. Base case: $G_2(A)$
2. Inductive step: $G_{k+1}(A)$


Deciding history-determinism: \(k\)-token games

**Lemma**

Eve wins \(G_2(\mathcal{A})\) if, and only if, she wins \(G_k(\mathcal{A})\) for all \(k \geq 2\).

**Proof**

By induction on \(k\):

\[
\begin{array}{c}
G_k(\mathcal{A}) \\
\end{array}
\quad
\begin{array}{c}
G_{k+1}(\mathcal{A}) \\
\end{array}
\]

Deciding history-determinism: \( k \)-token games

**Lemma**

Eve wins \( G_2(\mathcal{A}) \) if, and only if, she wins \( G_k(\mathcal{A}) \) for all \( k \geq 2 \).

**Proof**

By induction on \( k \):

Deciding history-determinism: $k$-token games

**Lemma**

Eve wins $G_2(A)$ if, and only if, she wins $G_k(A)$ for all $k \geq 2$.

**Theorem (BK19,BKLS20)**

For Büchi and co-Büchi acceptance conditions, $A$ is history-deterministic if, and only if, Eve wins the 2-token game.


Deciding history-determinism: $k$-token games

**Lemma**

Eve wins $G_2(A)$ if, and only if, she wins $G_k(A)$ for all $k \geq 2$.

**Theorem (BK19,BKLS20)**

For Büchi and co-Büchi acceptance conditions, $A$ is history-deterministic if, and only if, Eve wins the 2-token game.

**Corollary**

History-determinism is decidable in polynomial time for Büchi and co-Büchi automata.


More results and open problems on history-determinism

<table>
<thead>
<tr>
<th>Deciding history-determinism</th>
</tr>
</thead>
<tbody>
<tr>
<td>does the 2-token game characterize history-determinism for parity automata?</td>
</tr>
</tbody>
</table>

Expressiveness

ω-automata can be determinized, so on expressiveness gap; co-Büchi history-deterministic automata can be exponentially more succinct than deterministic ones. Is there an equivalent gap for Büchi automata?

Extensions of automata

- History-determinism is undecidable for pushdown automata;
- History-deterministic timed automata are not determinizable.


More results and open problems on history-determinism

Deciding history-determinism
- *does the 2-token game characterize history-determinism for parity automata?*

Expressiveness [KS15]
- $\omega$-automata can be determinized, so on expressiveness gap;
- co-Büchi history-deterministic automata can be exponentially more succinct than deterministic ones.
- *is there an equivalent gap for Büchi automata?*

### Deciding history-determinism

- *does the 2-token game characterize history-determinism for parity automata?*

### Expressiveness [KS15]

- ω-automata can be determinized, so on expressiveness gap;
- co-Büchi history-deterministic automata can be exponentially more succinct than deterministic ones.
- *is there an equivalent gap for Büchi automata?*

### Extensions of automata [LZ20,BHL+22]

- history-determinism is undecidable for pushdown automata;
- history-deterministic timed automata are not determinizable.

---