Nondeterminism in the Presence of Diverse or Unknown Future.

U. Boker[1], **D. Kuperberg**[2,3], O. Kupferman[2], M. Skrzypczak[3]

[1]IST Austria

[2]Hebrew University of Jerusalem

[3]University of Warsaw.

Séminaire Graphes et Logique, LaBRI, Talence
07-01-2014

ω-words

$$w = \quad (a)-(a)-(b)-(a)-(c)-(b)-(b)-(a)-\cdots \quad \in \Sigma^\omega$$

ω-words

$$w = a - a - b - a - c - b - b - a - \cdots \in \Sigma^\omega$$

MSO logic

$$\exists_X \ \forall_{x \in X} \ a(x) \ \wedge \ \ldots$$

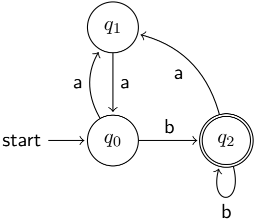
ω-words

$$w = \; a - a - b - a - c - b - b - a - \cdots \quad \in \Sigma^\omega$$

MSO logic

$$\exists_X \; \forall_{x \in X} \; a(x) \; \wedge \; \ldots$$

Finite automata



Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

deterministic : the run of $\mathcal{A}$ on w is accepting

Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

deterministic : the run of $\mathcal{A}$ on w is accepting

non-det. : exists an accepting run of $\mathcal{A}$ on w

Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

deterministic : the run of $\mathcal{A}$ on w is accepting

non-det. : exists an accepting run of $\mathcal{A}$ on w

satisfiability of formula $\rightsquigarrow$ emptiness for automaton

(polynomial in $|\mathcal{A}|$)

Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

deterministic : the run of $\mathcal{A}$ on w is accepting

non-det. : exists an accepting run of $\mathcal{A}$ on w

satisfiability of formula $\rightsquigarrow$ emptiness for automaton
(polynomial in $|\mathcal{A}|$)

$$|\mathcal{A}_{\text{non-det.}}| \ll |\mathcal{A}_{\text{det}}|$$

also simpler (cf. Safra)

Model checking

formula $\rightsquigarrow$ automaton

$$w \models \varphi \quad \text{iff} \quad \mathcal{A} \text{ accepts } w$$

deterministic : the run of $\mathcal{A}$ on w is accepting

non-det. : exists an accepting run of $\mathcal{A}$ on w

satisfiability of formula $\rightsquigarrow$ emptiness for automaton
(polynomial in $|\mathcal{A}|$)

$$|\mathcal{A}_{\text{non-det.}}| \ll |\mathcal{A}_{\text{det}}|$$

also simpler (cf. Safra)

(we focus on parity automata)

This work

Three classes in-between deterministic and non-det.:

This work

Three classes in-between deterministic and non-det.:

- Good For Games (history deterministic)

This work

Three classes in-between deterministic and non-det.:

- Good For Games (history deterministic)
- Good For Trees

This work

Three classes in-between deterministic and non-det.:

- Good For Games (history deterministic)
- Good For Trees
- Determinisable By Pruning

This work

Three classes in-between deterministic and non-det.:

- Good For Games (history deterministic)
- Good For Trees
- Determinisable By Pruning

Same expressive power for each parity index $[i, j]$
[Kupferman+Vardi for Büchi case]
[Niwinski+Walukiewicz for general $[i, j]$]
reproved here via game arguments.
but...

This work

Three classes in-between deterministic and non-det.:

- Good For Games (history deterministic)
- Good For Trees
- Determinisable By Pruning

Same expressive power for each parity index $[i, j]$
[Kupferman+Vardi for Büchi case]
[Niwinski+Walukiewicz for general $[i, j]$]
reproved here via game arguments.
but...

Containment???

Size???

Determinisation???

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

I_0

$\forall$ gives input I_0

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0

$I_0 \quad O_0$

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1

$$I_0 \quad O_0 \quad I_1$$

Synthesis problem

$\forall$ — environment

$\exists$ — system

φ — specification

$\forall$ gives input I_0

$\exists$ gives output O_0

$\forall$ gives input I_1

...

$$I_0 \quad O_0 \quad I_1 \quad \cdots$$

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
...

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad\right) \models \varphi$$

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
...

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad \right) \models \varphi$$

- Is it possible to win?

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
...

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad \right) \models \varphi$$

- Is it possible to win?
- Synthesize a machine that wins?

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
...

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad \right) \models \varphi$$

- Is it possible to win?
- Synthesize a machine that wins?

idea: reduce to a finite parity game

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
$\dots$

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad \right) \models \varphi$$

- Is it possible to win?
- Synthesize a machine that wins?

idea: reduce to a finite parity game

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

Synthesis problem

$\forall$ — environment
$\exists$ — system
φ — specification

$\forall$ gives input I_0
$\exists$ gives output O_0
$\forall$ gives input I_1
$\dots$

$$\exists \text{ wins if } \left(I_0 \quad O_0 \quad I_1 \quad \cdots \quad \right) \models \varphi$$

- Is it possible to win?
- Synthesize a machine that wins?

idea: reduce to a finite parity game

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
???	:	transition

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$		φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i		$\forall$	:	I_i
$\exists$	:	O_i		$\exists$	:	O_i
det.	:	transition		**???**	:	transition

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$		φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i		$\forall$	:	I_i
$\exists$	:	O_i		$\exists$	:	O_i
det.	:	transition		**???**	:	transition

$\mathcal{A}$ is Good For Games if

$$\exists_{\sigma:\,\Sigma^*\to\mathcal{A}}\ \forall_{w\in\Sigma^\omega}\quad w\in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
???	:	transition

$\mathcal{A}$ is Good For Games if

$$\underbrace{\exists_{\sigma:\,\Sigma^*\to\mathcal{A}}}_{\text{advice}} \forall_{w\in\Sigma^\omega} \quad w \in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$		φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i		$\forall$	:	I_i
$\exists$	:	O_i		$\exists$	:	O_i
det.	:	transition		**???**	:	transition

$\mathcal{A}$ is Good For Games if

$$\underbrace{\exists_{\sigma:\,\Sigma^*\to\mathcal{A}}}_{\text{advice}} \forall_{w\in\Sigma^\omega} \quad \underbrace{w \in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$		φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i		$\forall$	:	I_i
$\exists$	:	O_i		$\exists$	:	O_i
det.	:	transition		**???**	:	transition

$\mathcal{A}$ is Good For Games if

$$\exists_{\sigma:\,\Sigma^*\to\mathcal{A}}\ \forall_{w\in\Sigma^\omega}\quad w \in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{GFG}}$
$\forall$	:	I_i
$\exists$	:	O_i
$\exists$	:	transition

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
???	:	transition

$\mathcal{A}$ is Good For Games if

$$\exists_{\sigma : \Sigma^* \to \mathcal{A}} \forall_{w \in \Sigma^\omega} \quad w \in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{GFG}}$
$\forall$	:	I_i
$\exists$	:	O_i
$\exists$	:	transition

- [Henzinger, Piterman]

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
???	:	transition

$\mathcal{A}$ is Good For Games if

$$\exists_{\sigma\,:\,\Sigma^*\to\mathcal{A}}\ \forall_{w\in\Sigma^\omega}\quad w\in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{GFG}}$
$\forall$	:	I_i
$\exists$	:	O_i
$\exists$	:	transition

- [Henzinger, Piterman]
- also known as **History Determinism**

Good For Games

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
det.	:	transition

φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{non-det.}}$
$\forall$	:	I_i
$\exists$	:	O_i
???	:	transition

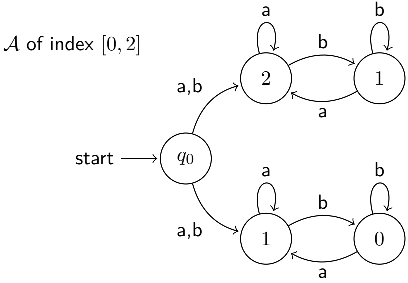
$\mathcal{A}$ is Good For Games if

$$\exists_{\sigma : \Sigma^* \to \mathcal{A}} \forall_{w \in \Sigma^\omega} \quad w \in \mathrm{L}(\mathcal{A}) \Rightarrow \sigma(w) \text{ is accepting}$$

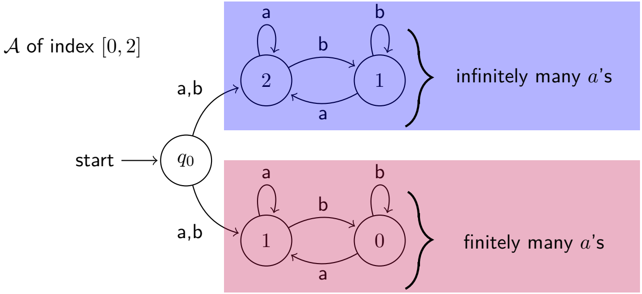
φ	$\rightsquigarrow$	$\mathcal{A}_{\textbf{GFG}}$
$\forall$	:	I_i
$\exists$	:	O_i
$\exists$	:	transition

- [Henzinger, Piterman]
- also known as **History Determinism**
- applications to counter automata [Colcombet]

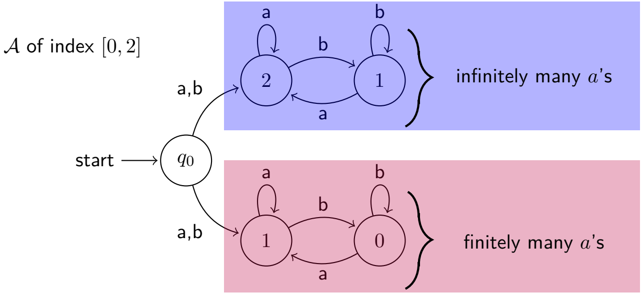
A non-example



A non-example



A non-example



$$\mathrm{L}(\mathcal{A}) = \{a,b\}^\omega$$ but $\mathcal{A}$ is **not** GFG

Good For Trees — CTL* model checking

Good For Trees — CTL* model checking

A subproblem:

Given	:	$\mathcal{A}$ for an LTL path formula φ
Compute	:	$\mathcal{B}$ for the CTL* formula Aφ

Good For Trees — CTL* model checking

A subproblem:

$$\begin{array}{rcl} \text{Given} & : & \mathcal{A} \text{ for an LTL path formula } \varphi \\ \text{Compute} & : & \mathcal{B} \text{ for the CTL* formula } \mathsf{A}\varphi \end{array}$$

ω-word automaton for $L \rightsquigarrow$ tree automaton for $\mathrm{der}(L)$

Good For Trees — CTL* model checking

A subproblem:

$$\text{Given} \quad : \quad \mathcal{A} \text{ for an LTL path formula } \varphi$$

$$\text{Compute} \quad : \quad \mathcal{B} \text{ for the CTL* formula } \mathsf{A}\varphi$$

ω-word automaton for $L \rightsquigarrow$ tree automaton for $\mathrm{der}(L)$

$\mathrm{der}(L) =$ set of partial Σ-branching trees with all branches in L

Good For Trees — CTL* model checking

A subproblem:

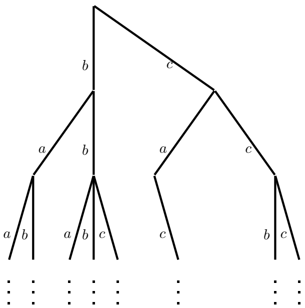
Given : $\mathcal{A}$ for an LTL path formula φ

Compute : $\mathcal{B}$ for the CTL* formula Aφ

ω-word automaton for $L \rightsquigarrow$ tree automaton for $\mathrm{der}(L)$

$\mathrm{der}(L) =$ set of partial Σ-branching trees with all branches in L

$\Sigma = \{a, b, c\}$



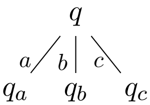
Good For Trees

$\mathrm{der}(L)$ = set of partial Σ-branching trees with all branches in L

Good For Trees

$\mathrm{der}(L)$ = set of partial Σ-branching trees with all branches in L

Brutal construction $\widehat{\mathcal{A}}$:



whenever

$$q \xrightarrow{a} q_a$$

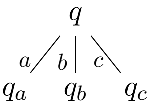
$$q \xrightarrow{b} q_b$$

$$q \xrightarrow{c} q_c$$

Good For Trees

$\mathrm{der}(L)$ = set of partial Σ-branching trees with all branches in L

Brutal construction $\widehat{\mathcal{A}}$:



whenever

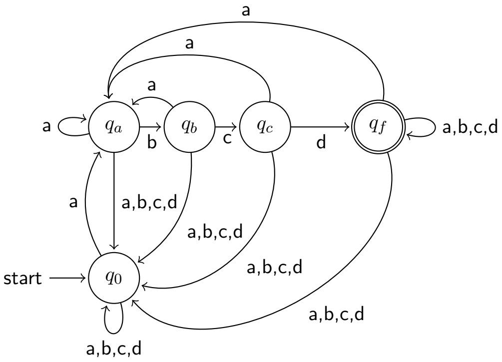
$$q \xrightarrow{a} q_a$$

$$q \xrightarrow{b} q_b$$

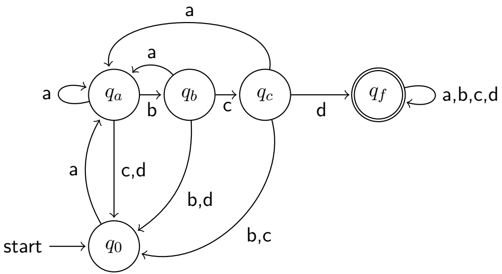
$$q \xrightarrow{c} q_c$$

$\mathcal{A}$ is Good For Trees if $\mathbf{L}(\widehat{\mathcal{A}}) = \mathbf{der}(L)$

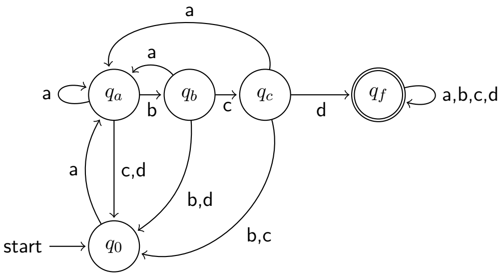
Determinisable By Pruning



Determinisable By Pruning

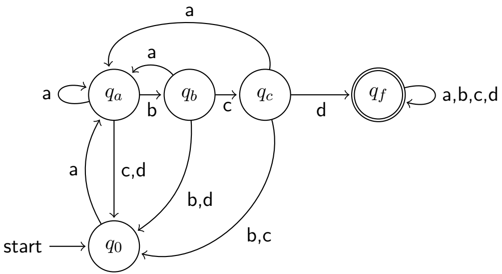


Determinisable By Pruning



$\mathcal{A}$ is DBP if $\mathcal{A}$ contains a deterministic subautomaton for $L(\mathcal{A})$

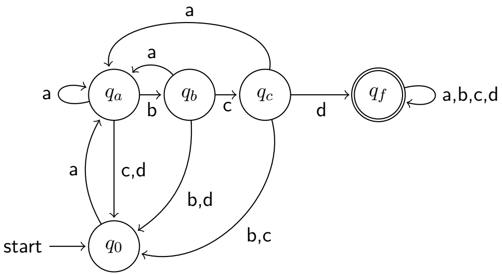
Determinisable By Pruning



$\mathcal{A}$ is DBP if $\mathcal{A}$ contains a deterministic subautomaton for $L(\mathcal{A})$

- simpler transition relations [Henzinger, Piterman]

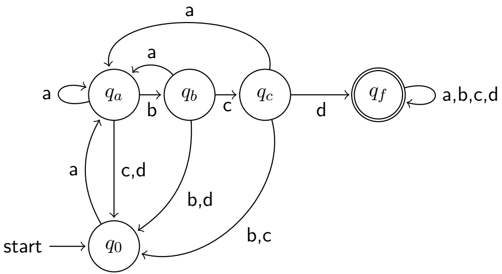
Determinisable By Pruning



$\mathcal{A}$ is DBP if $\mathcal{A}$ contains a deterministic subautomaton for $L(\mathcal{A})$

- simpler transition relations [Henzinger, Piterman]
- convenient for symbolic treatment

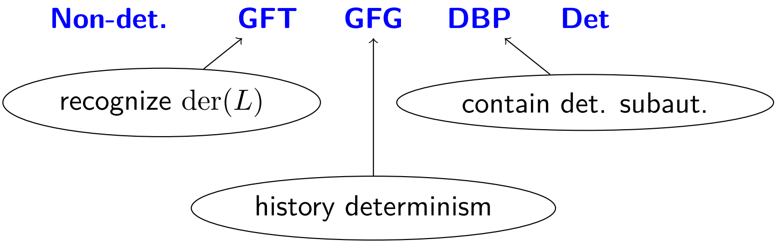
Determinisable By Pruning



$\mathcal{A}$ is DBP if $\mathcal{A}$ contains a deterministic subautomaton for $L(\mathcal{A})$

- simpler transition relations [Henzinger, Piterman]
- convenient for symbolic treatment
- easier to construct

Containment



Containment

$$\textbf{Non-det.} \supset \textbf{GFT} \overset{?}{\supseteq} \textbf{GFG} \overset{?}{\supseteq} \textbf{DBP} \supset \textbf{Det}$$

Containment

$$\textbf{Non-det.} \supset \textbf{GFT} = \textbf{GFG} \supsetneq \textbf{DBP} \supset \textbf{Det}$$

Containment

$$\textbf{Non-det.} \supset \textbf{GFT} = \textbf{GFG} \supsetneq \textbf{DBP} \supset \textbf{Det}$$

determinacy

Containment

$$\textbf{Non-det.} \supset \textbf{GFT} = \textbf{GFG} \supsetneq \textbf{DBP} \supset \textbf{Det}$$

two examples

Game for GFT = GFG

Game for **GFT = GFG**

Given automaton $\mathcal{A}$:

$$\begin{array}{lcl} \forall & : & a_i \\ \exists & : & q_i \xrightarrow{a_i} q_{i+1} \end{array}$$

Game for **GFT = GFG**

Given automaton $\mathcal{A}$:

$$\begin{array}{ccc} \forall & : & a_i \\ \exists & : & q_i \xrightarrow{a_i} q_{i+1} \end{array}$$

$\exists$ wins if:

run $(q_0, q_1, \ldots)$ is accepting $\quad \vee \quad$ word $(a_0, a_1, \ldots)$ is **not** in $\mathrm{L}(\mathcal{A})$

Game for **GFT = GFG**

Given automaton $\mathcal{A}$:

$$\begin{array}{ccl} \forall & : & a_i \\ \exists & : & q_i \xrightarrow{a_i} q_{i+1} \end{array}$$

$\exists$ wins if:

run $(q_0, q_1, \ldots)$ is accepting $\quad\vee\quad$ word $(a_0, a_1, \ldots)$ is **not** in $\mathrm{L}(\mathcal{A})$

- a winning strategy for $\exists$ is an advice $\sigma\colon \Sigma^* \to \mathcal{A}$
  $\rightsquigarrow$ GFG

Game for **GFT** = **GFG**

Given automaton $\mathcal{A}$:

$$\begin{array}{ccl} \forall & : & a_i \\ \exists & : & q_i \xrightarrow{a_i} q_{i+1} \end{array}$$

$\exists$ wins if:

run $(q_0, q_1, \ldots)$ is accepting $\quad\vee\quad$ word $(a_0, a_1, \ldots)$ is **not** in $\mathrm{L}(\mathcal{A})$

- a winning strategy for $\exists$ is an advice $\sigma\colon \Sigma^* \to \mathcal{A}$
  $\rightsquigarrow$ GFG
- a winning strategy for $\forall$ induces a tree $t \in \mathrm{der}(L) - \mathrm{L}(\widehat{\mathcal{A}})$
  $\rightsquigarrow$ **not** GFT

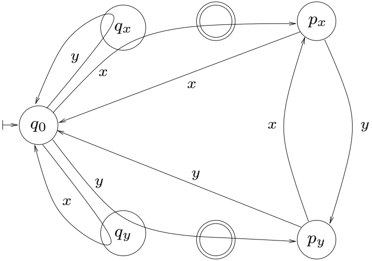
Examples for GFG $\supsetneq$ DBP

Examples for **GFG** $\supsetneq$ **DBP**

Büchi: infinitely many xx or yy

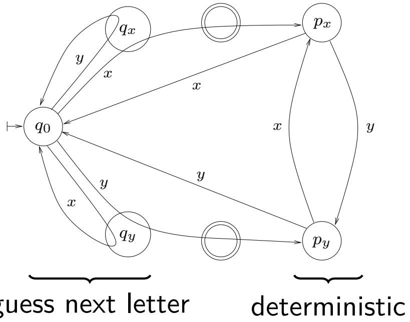
Examples for **GFG** $\supsetneq$ **DBP**

Büchi: infinitely many xx or yy



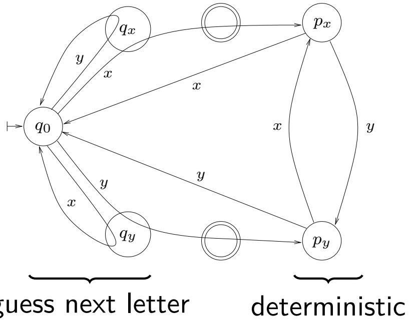
Examples for **GFG** $\supsetneq$ **DBP**

Büchi: infinitely many xx or yy



Examples for **GFG** $\supsetneq$ **DBP**

Büchi: infinitely many xx or yy



co-Büchi: follows from the *blow-up*

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

- in general $\sim 2^{n \log n}$

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

- in general $\sim 2^{n \log n}$
- for DBP $\leq n$

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

- in general $\sim 2^{n \log n}$
- for DBP $\leq n$
- what about GFG?

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

Results

- if $\mathcal{A}, \mathcal{B}$ are GFG and $\mathrm{L}(\mathcal{A}) = \overline{\mathrm{L}(\mathcal{B})}$ then $\sim |\mathcal{A}| \times |\mathcal{B}|$

Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

Results

- if $\mathcal{A}, \mathcal{B}$ are GFG and $\mathrm{L}(\mathcal{A}) = \overline{\mathrm{L}(\mathcal{B})}$ then $\sim |\mathcal{A}| \times |\mathcal{B}|$
- determinisation $\sim$ size of memory of an advice σ

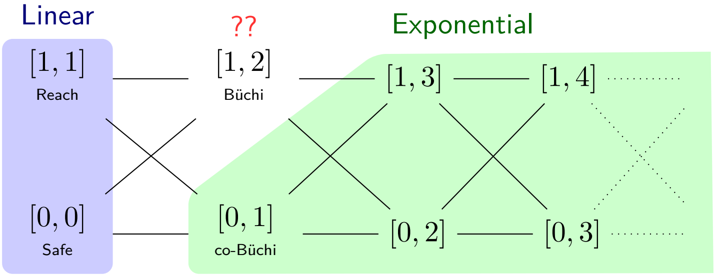
Blow-up

Problem: what is the blow-up when determinising?

(given $\mathcal{A}$ of size n find equivalent deterministic one)

Results

- if $\mathcal{A}, \mathcal{B}$ are GFG and $\mathrm{L}(\mathcal{A}) = \overline{\mathrm{L}(\mathcal{B})}$ then $\sim |\mathcal{A}| \times |\mathcal{B}|$
- determinisation $\sim$ size of memory of an advice σ



Summary

Summary

$$\mathbf{GFT} = \mathbf{GFG} \supsetneq \mathbf{DBP}$$

Summary

$$\mathbf{GFT} = \mathbf{GFG} \supsetneq \mathbf{DBP}$$

polynomial determinisation for pairs of GFG automata

Summary

$$\mathbf{GFT} = \mathbf{GFG} \supsetneq \mathbf{DBP}$$

polynomial determinisation for pairs of GFG automata

no polynomial determinisation for some GFG automaton

Summary

$$\mathbf{GFT} = \mathbf{GFG} \supsetneq \mathbf{DBP}$$

polynomial determinisation for pairs of GFG automata

no polynomial determinisation for some GFG automaton

Todo

Is there some subclass of GFG with polynomial determinisations?

Summary

$$\mathbf{GFT} = \mathbf{GFG} \supsetneq \mathbf{DBP}$$

polynomial determinisation for pairs of GFG automata

no polynomial determinisation for some GFG automaton

Todo

Is there some subclass of GFG with polynomial determinisations?

What about Büchi GFG?