On Determinisation of Good-for-Games Automata

Denis Kuperberg Michał Skrzypczak

University of Warsaw

Hightlights 2014
Paris

Success story: **synthesis** (Büchi, Landweber [1968]), ...

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$ — environment

$\exists$ — system

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$ $\pi =$

$\exists$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_0$ $\pi = I_0$

$\exists$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$ $\qquad\qquad$ $\pi = I_0, O_0$

$\exists$: O_0

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$: I_1 $\pi = I_0, O_0, I_1$

$\exists$

Success story: **synthesis** (Büchi, Landweber [1968]), . . .

φ: "specification": LTL, FO, MSO, . . .

$\forall$ $\pi = I_0, O_0, I_1, O_1$

$\exists$: O_1

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\qquad \pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\quad\quad$ $\pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$ $\quad\quad$ $\exists$ wins if $\pi \models \varphi$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\qquad \pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$ $\qquad \exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\quad \equiv \quad$ implementation

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\qquad \pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$ $\qquad \exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\equiv$ implementation

$\varphi \rightsquigarrow \mathcal{A}_{\textbf{det.}}$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$: I_n $\pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists$: O_n $\exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\equiv$ implementation

$\varphi \rightsquigarrow \mathcal{A}_{\textbf{det.}}$

$\forall$: I_n

$\exists$: O_n

det. : transition

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$: I_n $\qquad$ $\pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists$: O_n $\qquad$ $\exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\quad\equiv\quad$ implementation

$\boldsymbol{\varphi} \rightsquigarrow \mathcal{A}_{\textbf{det.}}$

$$\left.\begin{array}{rl} \forall : & I_n \\ \exists : & O_n \\ \textbf{det.} : & \text{transition} \end{array}\right\} \text{parity game}$$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\quad\quad$ $\pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$ $\quad\quad$ $\exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\equiv$ implementation

$\varphi \rightsquigarrow \mathcal{A}_{\textbf{det.}}$ } blow-up

$\forall$: I_n

$\exists$: O_n

det. : transition

} parity game

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall$: I_n $\pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists$: O_n $\exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\equiv$ implementation

$\boldsymbol{\varphi} \rightsquigarrow \mathcal{A}_{\textbf{det.}}$

$\forall$: I_n

$\exists$: O_n

det. : transition

$\boldsymbol{\varphi} \rightsquigarrow \mathcal{A}_{\textbf{non-det.}}$

Success story: **synthesis** (Büchi, Landweber [1968]), ...

φ: "specification": LTL, FO, MSO, ...

$\forall : I_n$ $\qquad \pi = I_0, O_0, I_1, O_1, I_2, O_2, \ldots$

$\exists : O_n$ $\qquad \exists$ wins if $\pi \models \varphi$

winning strategy for $\exists$ $\equiv$ implementation

$\varphi \rightsquigarrow \mathcal{A}_{\textbf{det.}}$	$\varphi \rightsquigarrow \mathcal{A}_{\textbf{non-det.}}$
$\forall$: I_n	$\forall$: I_n
$\exists$: O_n	$\exists$: O_n
det. : transition	**???** : transition

Good-for-Games automata

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

$$\exists_{\underbrace{\sigma\colon \Sigma^*\to\mathcal{A}}_{\textit{advice}}} \quad \forall_{w\in \mathrm{L}(\mathcal{A})} \quad \overbrace{\vec{\sigma}(w)}^{\left(\sigma(\epsilon),\ \sigma(w_0),\ \sigma(w_0w_1),\ \ldots\right)} \text{ is accepting}$$

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

$$\exists_{\sigma\colon \Sigma^* \to \mathcal{A}} \quad \forall_{w \in \mathrm{L}(\mathcal{A})} \quad \overbrace{\vec{\sigma}(w)}^{\left(\sigma(\epsilon),\ \sigma(w_0),\ \sigma(w_0 w_1),\ \ldots\right)} \text{ is accepting}$$

advice — σ accepts whenever possible

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{GFG}}$

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathsf{GFG}}$

$\forall$: I_n

$\exists$: O_n

$\exists$: transition $\Big\}$ using σ

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathsf{GFG}}$

$\forall$: I_n

$\exists$: O_n

$\exists$: transition

— symbolic representation

(Henzinger, Piterman [2006])

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathsf{GFG}}$

$\forall$: I_n

$\exists$: O_n

$\exists$: transition

— symbolic representation

(Henzinger, Piterman [2006])

— also known as **History Determinism**

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathsf{GFG}}$

$\forall$: I_n

$\exists$: O_n

$\exists$: transition

— symbolic representation

(Henzinger, Piterman [2006])

— also known as **History Determinism**

— counter automata

(Colcombet, Löding [2010])

Good-for-Games automata

$\mathcal{A}$ is Good-for-Games if

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

$\varphi \rightsquigarrow \mathcal{A}_{\mathsf{GFG}}$

$\forall$: I_n

$\exists$: O_n

$\exists$: transition

— symbolic representation

(Henzinger, Piterman [2006])

— also known as **History Determinism**

— counter automata

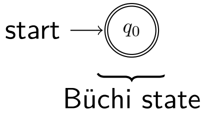
(Colcombet, Löding [2010])

— Good-for-Trees automata

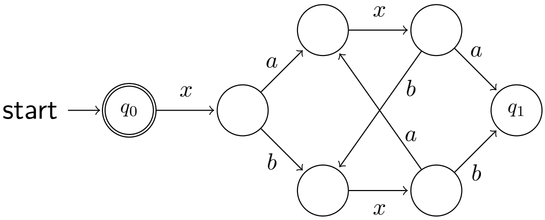
(Boker, K., Kupferman, S. [2013])

The GFG example (Boker [2013])

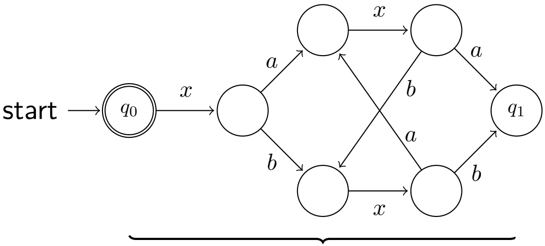
The GFG example (Boker [2013])



The GFG example (Boker [2013])

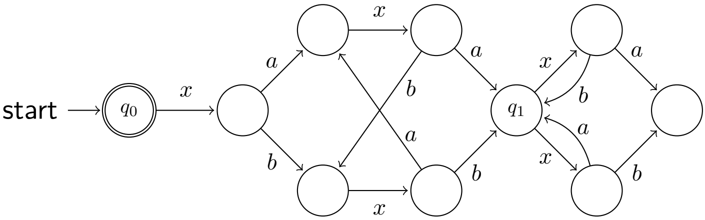


The GFG example (Boker [2013])

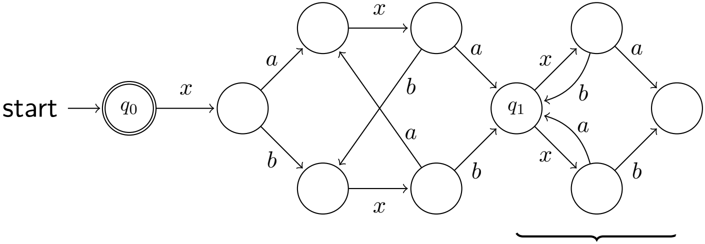


words $\left(x\ \{a, b\}\right)^*$
containing $xaxa$ or $xbxb$

The GFG example (Boker [2013])



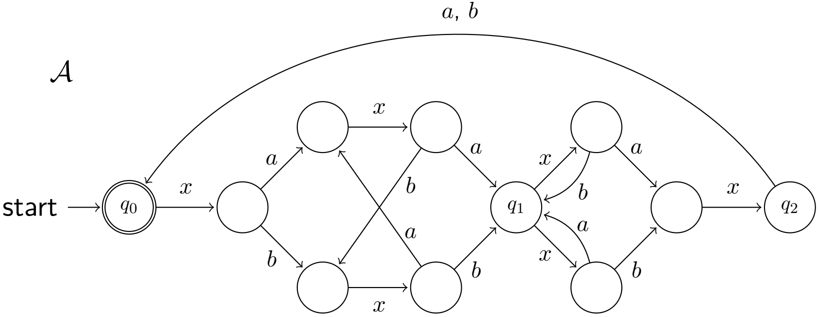
The GFG example (Boker [2013])



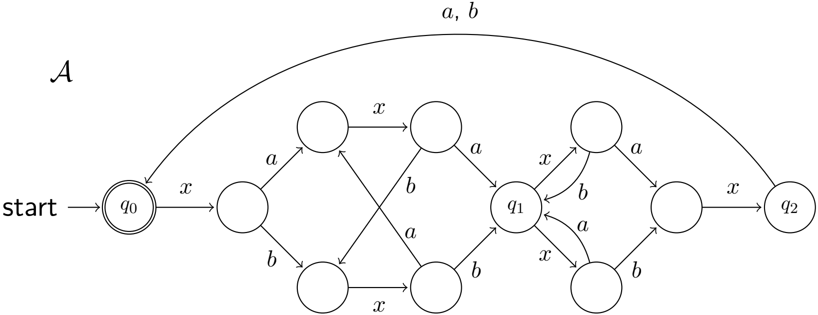
nondeterministically

word xa or xb

The GFG example (Boker [2013])

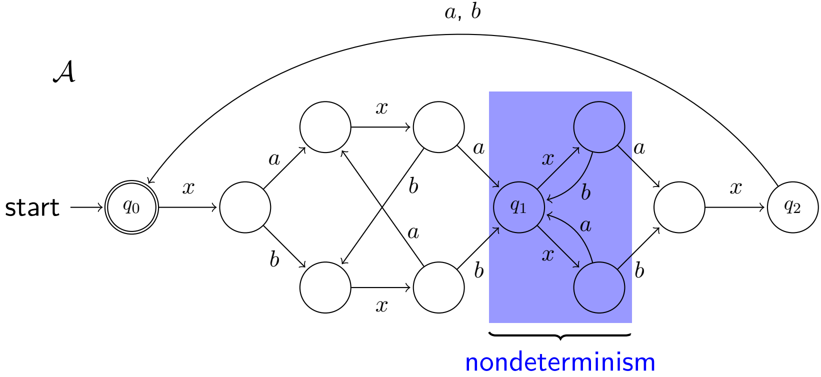


The GFG example (Boker [2013])



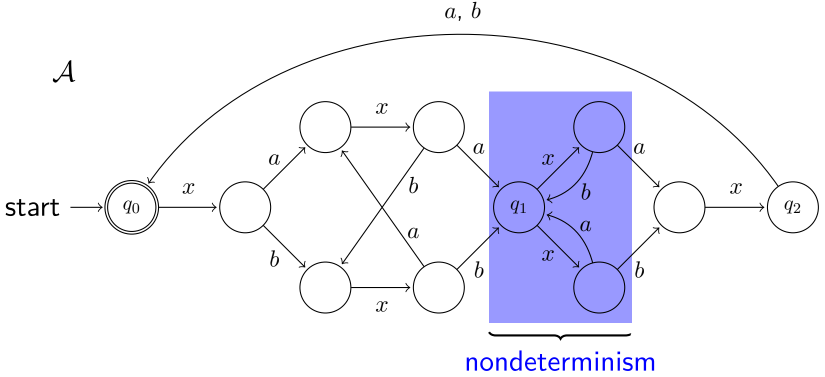
$\mathrm{L}(\mathcal{A}) =$ words $\left(x\ \{a, b\}\right)^\omega$ containing infinitely many $xaxa$ or $xbxb$

The GFG example (Boker [2013])



$$\mathrm{L}(\mathcal{A}) = \begin{array}{l}\text{words } \left(x\ \{a,b\}\right)^{\omega} \text{ containing} \\ \text{infinitely many } xaxa \text{ or } xbxb\end{array}$$

The GFG example (Boker [2013])



$\mathrm{L}(\mathcal{A}) =$ words $\left(x\ \{a,b\}\right)^\omega$ containing infinitely many $xaxa$ or $xbxb$

σ: guess that last a/b will reappear

Determinisation

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Büchi automata	**co-Büchi automata**

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Büchi automata	**co-Büchi automata**
There exists $\mathcal{B}$ polynomial in $\mathcal{A}$.	

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Büchi automata

There exists $\mathcal{B}$ polynomial in $\mathcal{A}$.

[efficient determinisation procedure]

co-Büchi automata

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Büchi automata

There exists $\mathcal{B}$ polynomial in $\mathcal{A}$.

[efficient determinisation procedure]

co-Büchi automata

Minimal $\mathcal{B}$ may be exponential in $\mathcal{A}$.

Determinisation

Given GFG automaton $\mathcal{A}$ is there small deterministic $\mathcal{B}$ with

$$\mathrm{L}(\mathcal{A}) = \mathrm{L}(\mathcal{B})?$$

Lower-bound: linear

Upper-bound: exponential

Solution:

Büchi automata	**co-Büchi automata**
There exists $\mathcal{B}$ polynomial in $\mathcal{A}$.	Minimal $\mathcal{B}$ may be exponential in $\mathcal{A}$.
[efficient determinisation procedure]	[GFG automata are succinct]

Proof sketches

Proof sketches

Büchi automata:

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

— signatures of Walukiewicz

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

— signatures of Walukiewicz

— iterative normalization of $\mathcal{A}$

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

— signatures of Walukiewicz

— iterative normalization of $\mathcal{A}$

— dependency graph over $\mathcal{A}$

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

— signatures of Walukiewicz

— iterative normalization of $\mathcal{A}$

— dependency graph over $\mathcal{A}$

co-Büchi automata:

Proof sketches

Büchi automata:

— brutal (exponential) determinisation

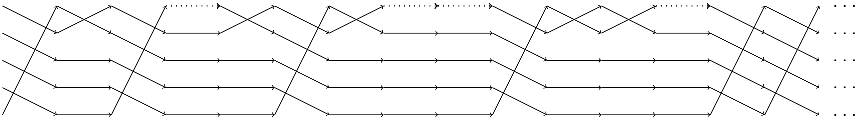
— signatures of Walukiewicz

— iterative normalization of $\mathcal{A}$

— dependency graph over $\mathcal{A}$

co-Büchi automata:

— permutation languages



Proof sketches

Büchi automata:

— brutal (exponential) determinisation

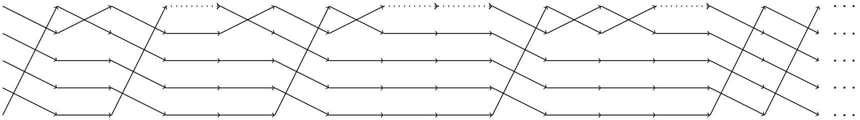
— signatures of Walukiewicz

— iterative normalization of $\mathcal{A}$

— dependency graph over $\mathcal{A}$

co-Büchi automata:

— permutation languages



— compactness argument for *limitary pumping*

Summary

Summary

Efficient determinisation construction for Büchi GFG

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime** $\cap$ **co-NEXPtime**]

[**EXPtime** for bounded index]

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime ∩ co-NEXPtime**]

— **NP** algorithm for Büchi

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime** $\cap$ **co-NEXPtime**]

— **NP** algorithm for Büchi

— **Ptime** algorithm for co-Büchi

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime** $\cap$ **co-NEXPtime**]

— **NP** algorithm for Büchi

— **Ptime** algorithm for co-Büchi
[construction of a GFG candidate]

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime ∩ co-NEXPtime**]

— **NP** algorithm for Büchi

— **Ptime** algorithm for co-Büchi

[construction of a GFG candidate]

TODO:

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime** $\cap$ **co-NEXPtime**]

— **NP** algorithm for Büchi

— **Ptime** algorithm for co-Büchi
[construction of a GFG candidate]

TODO:

— **Ptime** vs. **NP** for Büchi

Summary

Efficient determinisation construction for Büchi GFG

Succinctness of co-Büchi GFG

Also: deciding if $\mathcal{A}$ is GFG:

[known algorithm: **NEXPtime ∩ co-NEXPtime**]

— **NP** algorithm for Büchi

— **Ptime** algorithm for co-Büchi
[construction of a GFG candidate]

TODO:

— **Ptime** vs. **NP** for Büchi

— what about deciding GFG for higher indices?