On determinisation of Good-For-Games automata

Denis Kuperberg Michał Skrzypczak

LIAFA & University of Warsaw

ICALP 2015

Kyoto (京 都)

Synthesis

Synthesis

φ — specification (LTL, MSO, ...)

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

— machine

Synthesis

φ — specification (LTL, MSO, ...)

 — machine

Environment:

Machine:

Synthesis

φ — specification (LTL, MSO, ...)

⇝

 — machine

Environment: I_0,

Machine:

Synthesis

φ — specification (LTL, MSO, …)

$\rightsquigarrow$

 — machine

Environment: I_0,

$\downarrow$

$\downarrow$

Machine: O_0,

Synthesis

φ — specification (LTL, MSO, ...)

⇝

 — machine

Environment: $I_0, \;\; I_1,$

↓

↓

Machine: $O_0,$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: I_0, I_1,

$\downarrow$

$\downarrow$

Machine: O_0, O_1,

Synthesis

φ — specification (LTL, MSO, . . .)

$\rightsquigarrow$

 — machine

Environment: $I_0, \; I_1, \; I_2,$

$\downarrow$

$\downarrow$

Machine: $O_0, \; O_1,$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0, \ I_1, \ I_2,$

$\downarrow$

$\downarrow$

Machine: $O_0, \ O_1, \ O_2,$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

— machine

Environment: $I_0, \; I_1, \; I_2, \; \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, O_1, O_2, \ldots$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0,\ I_1,\ I_2,\ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$\big(I_0, O_0,\ I_1, O_1,\ \ldots\big) \models \varphi$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0,\ I_1,\ I_2,\ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$$\big(I_0, O_0,\ I_1, O_1,\ \ldots\big) \models \varphi$$

Game semantics

Synthesis

φ — specification (LTL, MSO, …)

⇝

— machine

Environment: $I_0,\ I_1,\ I_2,\ \dots$

↓

↓

Machine: $O_0,\ O_1,\ O_2,\ \dots$

$$\big(I_0, O_0,\ I_1, O_1,\ \dots\big) \models \varphi$$

Game semantics

Two players, perfect information

Synthesis

φ — specification (LTL, MSO, …)

$\rightsquigarrow$

 — machine

Environment: $I_0,\ \ I_1,\ \ I_2,\ \ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$$\big(I_0, O_0,\ I_1, O_1,\ \ldots\big) \models \varphi$$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

— machine

Environment: $I_0, \; I_1, \; I_2, \; \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \; O_1, \; O_2, \; \ldots$

$$\big(I_0, O_0, \; I_1, O_1, \; \ldots\big) \models \varphi$$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

$\exists$ wins if $\big(I_0, O_0, \; \ldots\big) \models \varphi$

Synthesis

φ — specification (LTL, MSO, …)

$\rightsquigarrow$

— machine

Environment: $I_0, \ I_1, \ I_2, \ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \ O_1, \ O_2, \ \ldots$

$$\big(I_0, O_0, \ I_1, O_1, \ \ldots\big) \models \varphi$$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

$\exists$ wins if $\underbrace{\big(I_0, O_0, \ \ldots\big) \models \varphi}_{\omega\text{-regular winning condition}}$

Synthesis

φ — specification (LTL, MSO, …)

$\rightsquigarrow$

 — machine

Environment: $I_0, \; I_1, \; I_2, \; \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \; O_1, \; O_2, \; \ldots$

$(I_0, O_0, \; I_1, O_1, \; \ldots) \models \varphi$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{(I_0, O_0, \; \ldots) \models \varphi}_{\omega\text{-regular winning condition}}$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0,\ I_1,\ I_2,\ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$$\big(I_0, O_0,\ I_1, O_1,\ \ldots\big) \models \varphi$$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{\big(I_0, O_0,\ \ldots\big) \models \varphi}_{}$

ω-regular winning condition

Solution

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

— machine

Environment: $I_0,\ I_1,\ I_2,\ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$(I_0, O_0,\ I_1, O_1,\ \ldots) \models \varphi$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{(I_0, O_0,\ \ldots) \models \varphi}_{\omega\text{-regular winning condition}}$

Solution

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0, \ I_1, \ I_2, \ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \ O_1, \ O_2, \ \ldots$

$\big(I_0, O_0, \ I_1, O_1, \ \ldots\big) \models \varphi$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{\big(I_0, O_0, \ \ldots\big) \models \varphi}_{\omega\text{-regular winning condition}}$

Solution

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

$\forall$: I_i

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0,\ I_1,\ I_2,\ \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0,\ O_1,\ O_2,\ \ldots$

$\big(I_0, O_0,\ I_1, O_1,\ \ldots\big) \models \varphi$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{\big(I_0, O_0,\ \ldots\big) \models \varphi}_{\omega\text{-regular winning condition}}$

Solution

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

$\forall$: I_i

$\exists$: O_i

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0, \; I_1, \; I_2, \; \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \; O_1, \; O_2, \; \ldots$

$\big(I_0, O_0, \; I_1, O_1, \; \ldots\big) \models \varphi$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{\big(I_0, O_0, \; \ldots\big) \models \varphi}_{\omega\text{-regular winning condition}}$

Solution

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

$\forall$: I_i

$\exists$: O_i

$\mathbf{det}$: transition of $\mathcal{A}_{\mathbf{det}}$

Synthesis

φ — specification (LTL, MSO, ...)

$\rightsquigarrow$

 — machine

Environment: $I_0, \; I_1, \; I_2, \; \ldots$

$\downarrow$

$\downarrow$

Machine: $O_0, \; O_1, \; O_2, \; \ldots$

$$\big(I_0, O_0, \; I_1, O_1, \; \ldots\big) \models \varphi$$

Game semantics

Two players, perfect information

$\forall$: I_i

$\exists$: O_i

(Büchi, Landweber ['69])

$\exists$ wins if $\underbrace{\big(I_0, O_0, \; \ldots\big) \models \varphi}_{\omega\text{-regular winning condition}}$

Solution

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

$\forall$: I_i

$\exists$: O_i

$\mathbf{det}$: transition of $\mathcal{A}_{\mathbf{det}}$

$\exists$ wins if

the run of $\mathcal{A}_{\mathbf{det}}$ is accepting

Complexity

Complexity

$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}}$ — det. automaton

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

Complexity

$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}$$

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Game

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Game

∀: I_i

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially}\text{ more succinct!}}$$

Game

∀: I_i

∃: O_i

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Game

∀: I_i

∃: O_i

?: transition of $\mathcal{A}_{\mathbf{non-det}}$

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Game

∀: I_i

∃: O_i

∃: transition of $\mathcal{A}_{\mathbf{non-det}}$

Complexity

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$$

Game

∀: I_i

∃: O_i

∃: transition of $\mathcal{A}_{\mathbf{non-det}}$

∀ may cheat!

Complexity

$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}$

exponentially more succinct!

Game

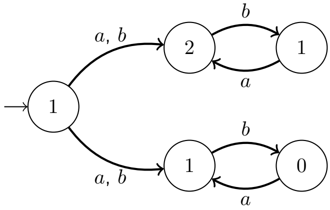
∀: I_i

∃: O_i

∃: transition of $\mathcal{A}_{\mathbf{non-det}}$

∀ may cheat!

$\mathcal{A}_{\mathbf{non-det}}$



Complexity

$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{det}} \text{ — det. automaton}}_{\textbf{expensive}!}$

[e.g. 2-EXP for LTL, non-ELEMENTARY for MSO]

Idea

$\underbrace{\varphi \rightsquigarrow \mathcal{A}_{\mathbf{non-det}}}_{\textbf{exponentially} \text{ more succinct!}}$

Game

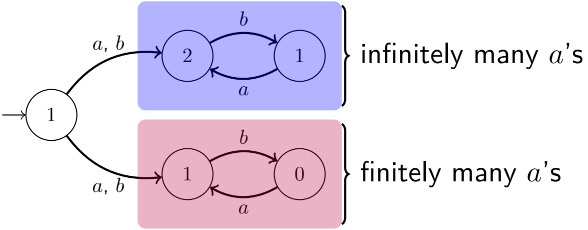
$\forall$: I_i

$\exists$: O_i

$\exists$: transition of $\mathcal{A}_{\mathbf{non-det}}$

∀ may cheat!

$\mathcal{A}_{\mathbf{non-det}}$



Good-For-Games automata

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\exists_{\sigma\colon \Sigma^* \to Q}$$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\exists_{\sigma:\,\Sigma^*\to Q} \qquad \forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})}$$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\exists_{\sigma\colon \Sigma^* \to Q} \qquad \forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \qquad \sigma(w) \text{ is accepting}$$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

$$\exists_{\sigma\colon \Sigma^* \to Q} \qquad \forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \qquad \sigma(w) \text{ is accepting}$$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\exists_{\underbrace{\sigma\colon \Sigma^* \to Q}_{\text{advice}}} \qquad \forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \qquad \sigma(w) \text{ is accepting}$$

$\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w) = \big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^* \to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^* \to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma:\,\Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$ ↙ $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

$\forall$: I_i

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

$$\underbrace{\exists_{\sigma\colon \Sigma^* \to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

Synthesis (Henzinger, Piterman ['06])

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

$\forall$: I_i

$\exists$: O_i

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

$$\underbrace{\exists_{\sigma:\,\Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

Synthesis (Henzinger, Piterman ['06])

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

∀: I_i

∃: O_i

∃: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$ → $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

- $\forall$: I_i
- $\exists$: O_i
- $\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

> σ is implicit — not included in $\mathcal{A}_{\mathbf{GFG}}$!

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^* \to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

- $\forall$: I_i
- $\exists$: O_i
- $\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^* \to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w \in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$ ← $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

$\forall$: I_i

$\exists$: O_i

$\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

History determinism (Colcombet, Löding ['08])

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

- $\forall$: I_i
- $\exists$: O_i
- $\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

History determinism (Colcombet, Löding ['08])

[replaces determinism for counter automata]

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

- $\forall$: I_i
- $\exists$: O_i
- $\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

History determinism (Colcombet, Löding ['08])

[replaces determinism for counter automata]

Theorem (Boker, Kuperberg, Kupferman, S. ['13])

GFG $\equiv$ Good-For-Trees $\big($derived languages $\forall\textsc{path}(L)\big)$

Good-For-Games automata

$\mathcal{A}_{\mathbf{non-det}}$ is Good-For-Games (GFG) if

$$\underbrace{\exists_{\sigma\colon \Sigma^*\to Q}}_{\text{advice}} \qquad \underbrace{\forall_{w\in \mathrm{L}(\mathcal{A}_{\mathbf{non-det}})} \quad \sigma(w) \text{ is accepting}}_{\sigma \text{ accepts whenever possible}}$$

$\sigma(w)$: $\big(\sigma(), \sigma(w_0), \sigma(w_0 w_1), \ldots\big)$

Synthesis (Henzinger, Piterman ['06])

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

- $\forall$: I_i
- $\exists$: O_i
- $\exists$: transition of $\mathcal{A}_{\mathbf{GFG}}$ (may use σ)

What is the blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$?

History determinism (Colcombet, Löding ['08])

[replaces determinism for counter automata]

Theorem (Boker, Kuperberg, Kupferman, S. ['13])

GFG $\equiv$ Good-For-Trees $\big($derived languages $\forall\mathrm{PATH}(L)\big)$

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Conjecture (Colcombet ['12])

The same holds for ω-words (parity automata).

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Conjecture (Colcombet ['12])

The same holds for ω-words (parity automata).

Theorem (Boker ['13])

There exists a Büchi GFG automaton with **no** equivalent deterministic subautomaton.

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Conjecture (Colcombet ['12])

The same holds for ω-words (parity automata).

Theorem (Boker ['13])

There exists a Büchi GFG automaton with **no** equivalent deterministic subautomaton.

[Büchi $\leftrightsquigarrow$ infinitely many accepting states]

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Conjecture (Colcombet ['12])

The same holds for ω-words (parity automata).

Theorem (Boker ['13])

There exists a Büchi GFG automaton with **no** equivalent deterministic subautomaton.

[Büchi $\leftrightsquigarrow$ infinitely many accepting states]

Question (Kupferman et al. ['13])

Does every GFG automaton admit a polynomial determinisation?

Blow-up of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

Theorem (Löding ['11])

Every GFG automaton over finite words contains an equivalent deterministic subautomaton.

$\rightsquigarrow$ no blow-up over finite words

Conjecture (Colcombet ['12])

The same holds for ω-words (parity automata).

Theorem (Boker ['13])

There exists a Büchi GFG automaton with **no** equivalent deterministic subautomaton.

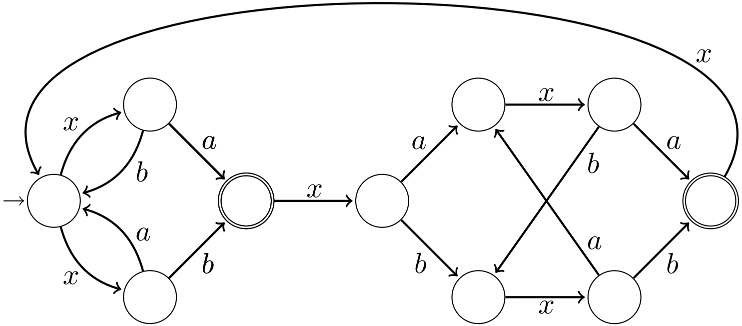
[Büchi $\updownarrow$ infinitely many accepting states]

Question (Kupferman et al. ['13])

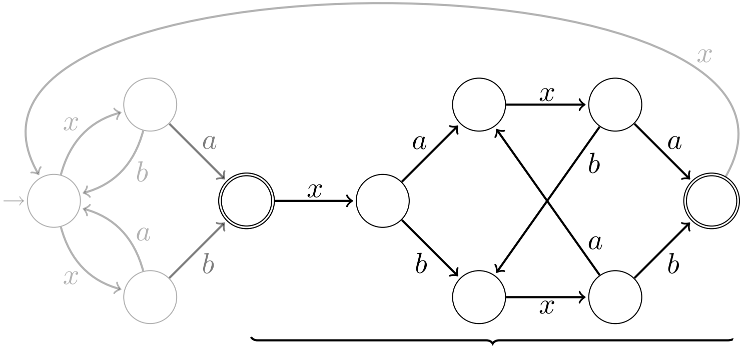
Does every GFG automaton admit a polynomial determinisation?

Boker's example

Boker's example

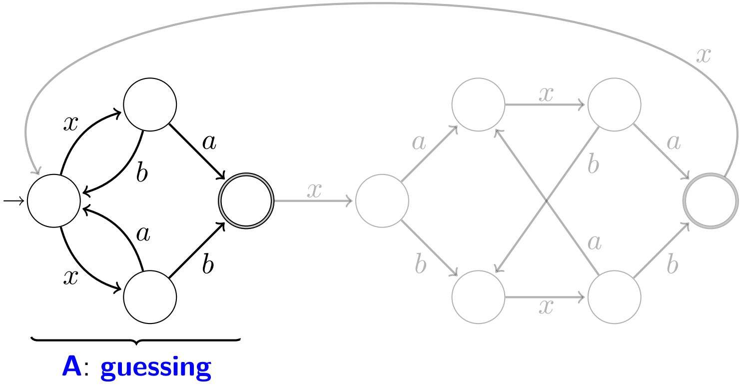


Boker's example

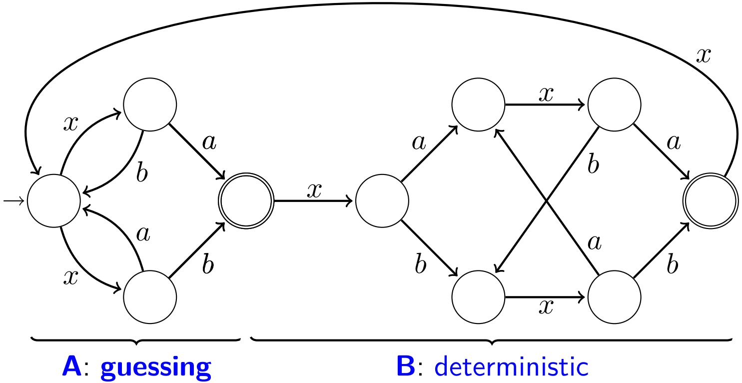


B: deterministic

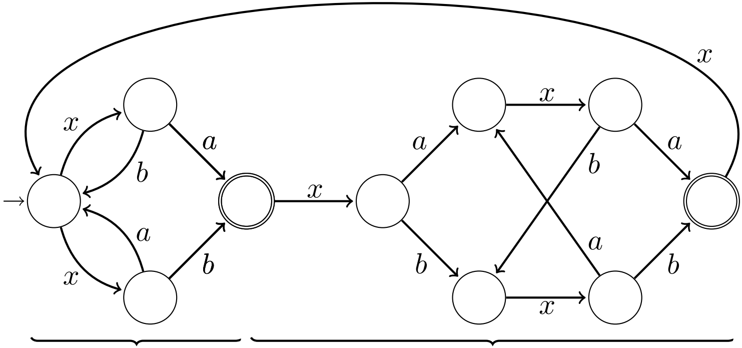
Boker's example



Boker's example



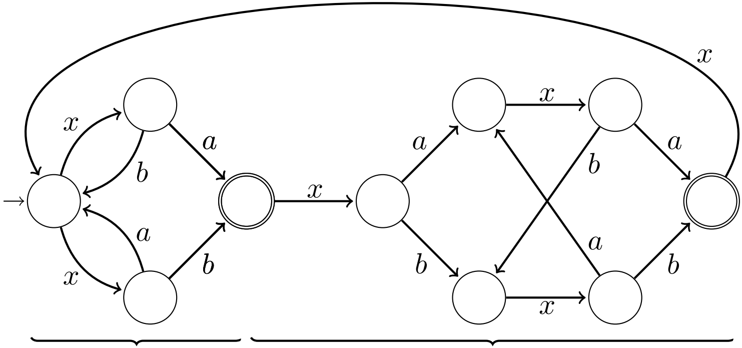
Boker's example



A: **guessing** **B**: deterministic

no equiv. det. subautomaton

Boker's example

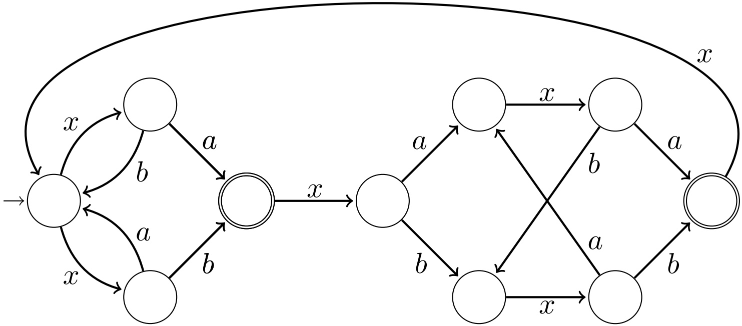


A: **guessing**   **B**: deterministic

no equiv. det. subautomaton

σ ="guess in **A** basing on **B**"

Boker's example



A: **guessing** **B**: deterministic

no equiv. det. subautomaton

σ =“guess in **A** basing on **B**”

[there exists $|\mathcal{A}_{\mathbf{det}}| \sim |\mathcal{A}_{\mathbf{GFG}}|$]

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Theorem

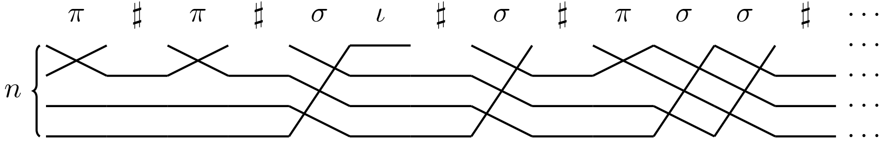
For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

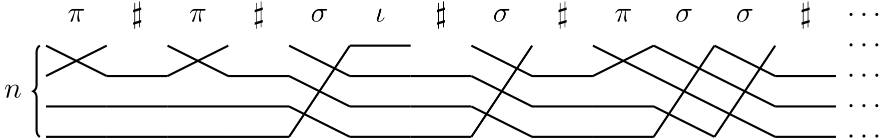
Plaits



Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits

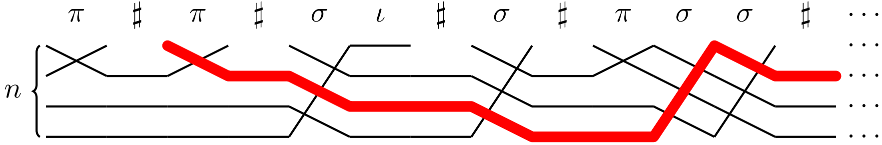


L ="exists some infinite path"

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits

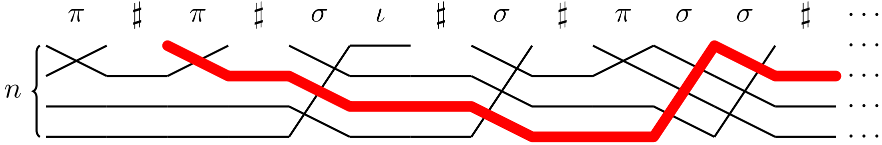


L ="exists some infinite path"

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



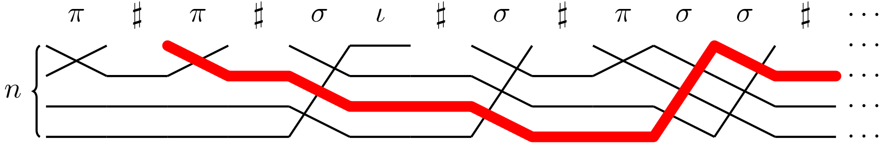
L ="exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L =“exists some infinite path”

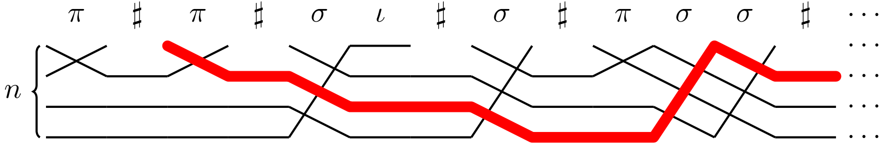
$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L = "exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

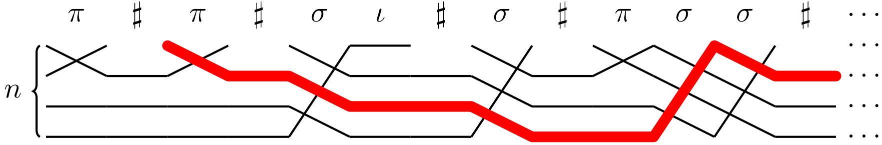
when reached a dead end, guess a new path (via a rejecting transition)

$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L ="exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

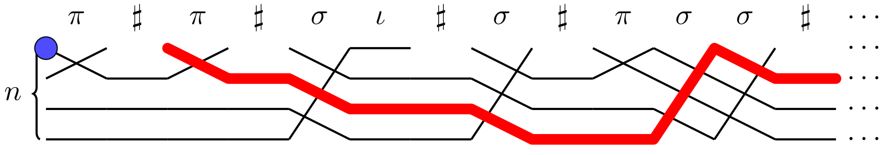
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L =“exists some infinite path”

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

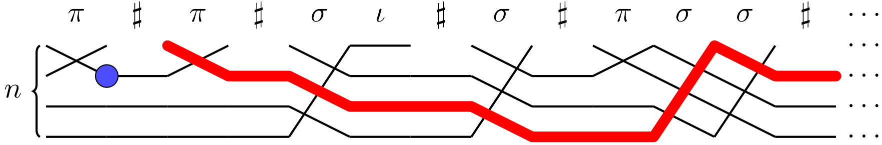
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L = "exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

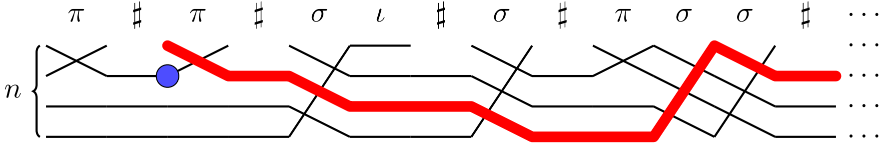
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

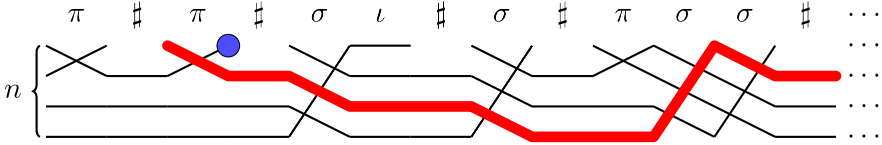
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L ="exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

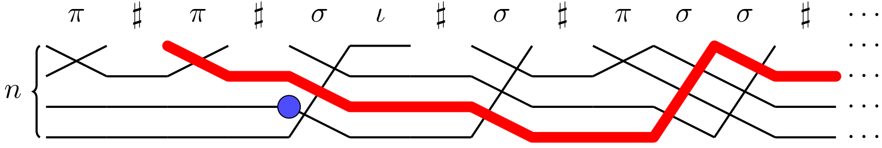
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L = "exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

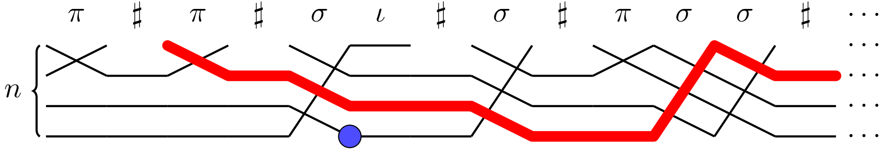
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

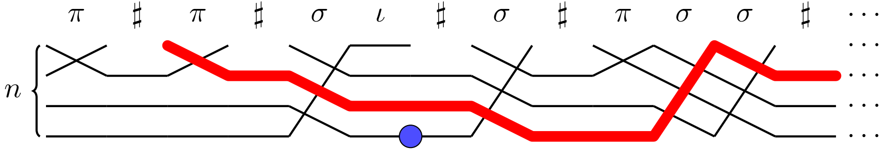
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

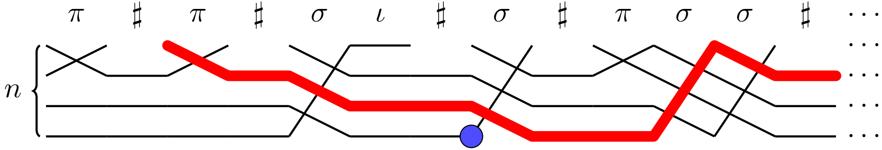
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L ="exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

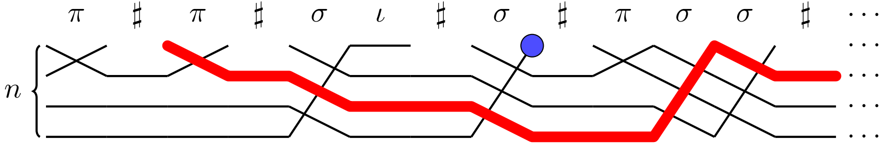
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L =“exists some infinite path”

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

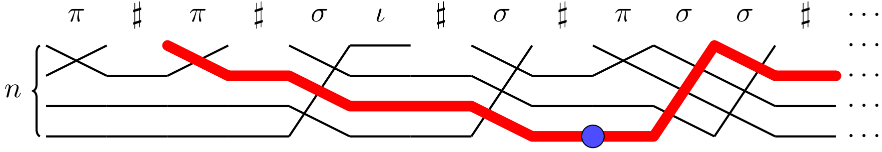
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L = "exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

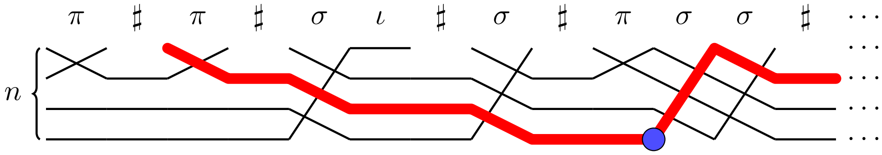
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

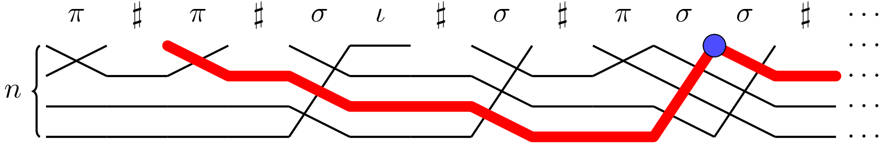
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

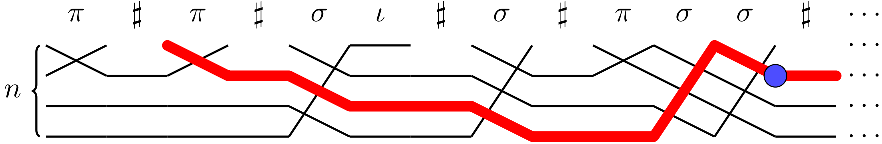
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



$\mathrm{L} =$"exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

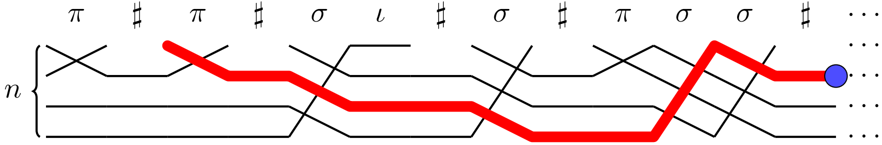
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L ="exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

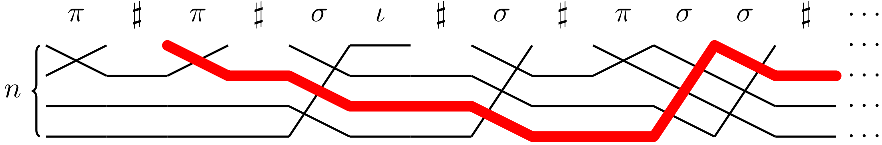
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L =“exists some infinite path”

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

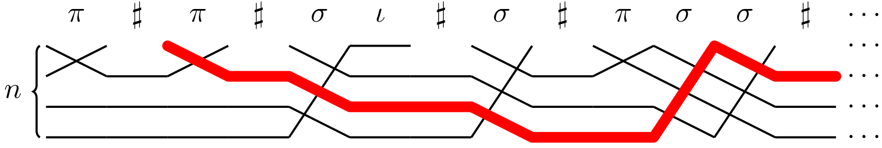
$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Theorem

For co-Büchi automata $|\mathcal{A}_{\mathbf{det}}| \sim 2^{|\mathcal{A}_{\mathbf{GFG}}|}$

Plaits



L = "exists some infinite path"

$\mathcal{A}_{\mathbf{GFG}}$: guess a path and follow

when reached a dead end, guess a new path (via a rejecting transition)

$|\mathcal{A}_{\mathbf{GFG}}| \sim n$

σ: try the oldest path available

Lemma

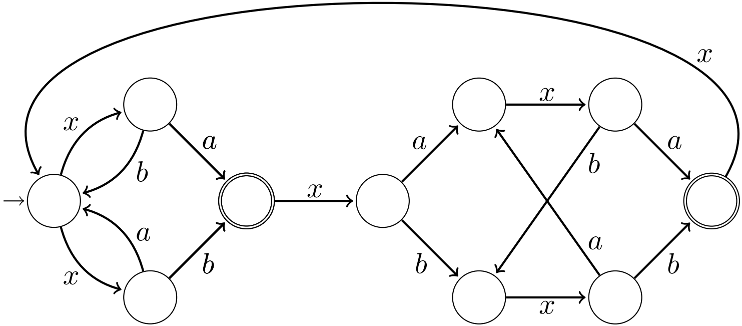
$|\mathcal{A}_{\mathbf{det}}| \geq 2^n$

Theorem

For Büchi automata $|\mathcal{A}_{\mathbf{det}}| \leqslant |\mathcal{A}_{\mathbf{GFG}}|^2$

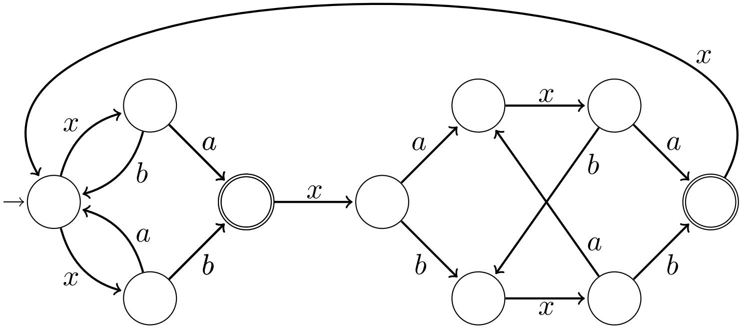
Theorem

For Büchi automata $|\mathcal{A}_{\mathbf{det}}| \leqslant |\mathcal{A}_{\mathbf{GFG}}|^2$



Theorem

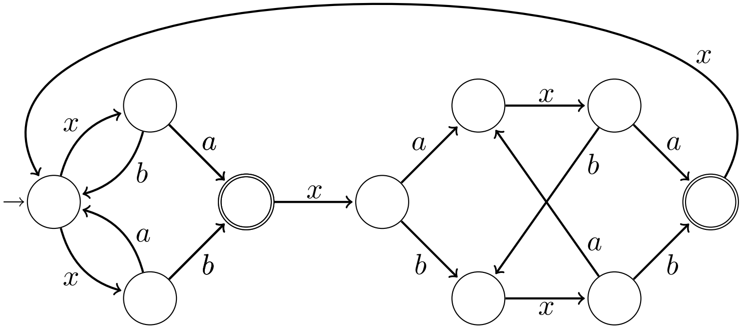
For Büchi automata $|\mathcal{A}_{\mathbf{det}}| \leqslant |\mathcal{A}_{\mathbf{GFG}}|^2$



Where the advice σ comes from?

Theorem

For Büchi automata $|\mathcal{A}_{\mathbf{det}}| \leqslant |\mathcal{A}_{\mathbf{GFG}}|^2$



Where the advice σ comes from?

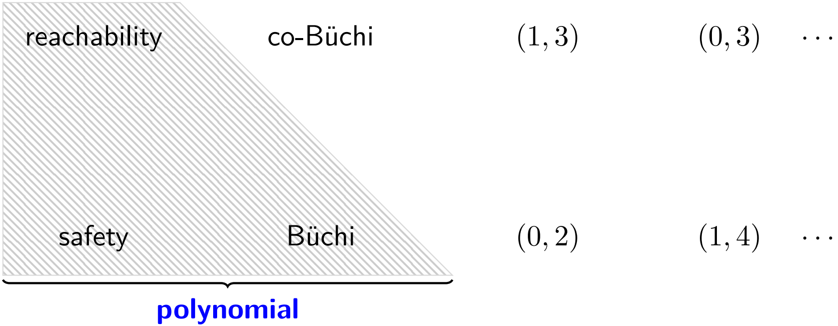
⇝ signatures of Walukiewicz + iterative normalisation

Picture of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

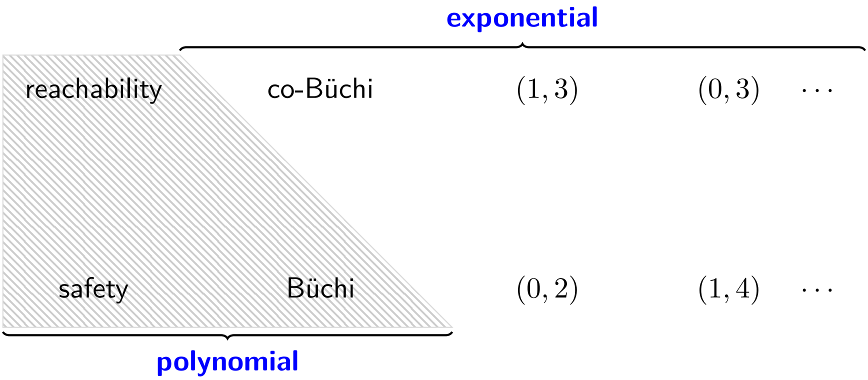
Picture of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$

reachability	co-Büchi	$(1,3)$	$(0,3)$	$\cdots$
safety	Büchi	$(0,2)$	$(1,4)$	$\cdots$

Picture of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$



Picture of $\mathcal{A}_{\mathbf{GFG}} \rightsquigarrow \mathcal{A}_{\mathbf{det}}$



Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

[EXPTIME algorithm from (Henzinger, Piterman ['06])]

Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

[EXPTIME algorithm from (Henzinger, Piterman ['06])]

Proposition

For parity automata at least as hard as solving parity games

Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

[EXPTIME algorithm from (Henzinger, Piterman ['06])]

Proposition

For parity automata at least as hard as solving parity games

Proposition

NP algorithm for Büchi automata

Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

[EXPTIME algorithm from (Henzinger, Piterman ['06])]

Proposition

For parity automata at least as hard as solving parity games

Proposition

NP algorithm for Büchi automata

Theorem

PTIME algorithm for co-Büchi automata

Problem

Input: an automaton $\mathcal{A}_{\mathbf{non-det}}$
Output: is $\mathcal{A}_{\mathbf{non-det}}$ GFG?

[EXPTIME algorithm from (Henzinger, Piterman ['06])]

Proposition

For parity automata at least as hard as solving parity games

Proposition

NP algorithm for Büchi automata

Theorem

PTIME algorithm for co-Büchi automata

Question

What about $(1, 3)$-parity automata?

Summary

Summary

Two positive results:

Summary

Two positive results:

co-Büchi GFG automata
are exponentially succinct

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

Büchi GFG automata can be efficiently determinised

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

$\rightsquigarrow$

potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

⇝

potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised

⇝

fast complementation algorithm

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

⇝

potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised

⇝

fast complementation algorithm

More **efficient** characterisations:

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

⇝

potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised

⇝

fast complementation algorithm

More **efficient** characterisations:

NP for Büchi

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct
⇝
potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised
⇝
fast complementation algorithm

More **efficient** characterisations:

NP for Büchi

PTime for co-Büchi

Summary

Two positive results:

co-Büchi GFG automata are exponentially succinct

⇝

potential speed-up in synthesis

Büchi GFG automata can be efficiently determinised

⇝

fast complementation algorithm

More **efficient** characterisations:

NP for Büchi

PTime for co-Büchi

On the way:

game theoretic arguments,

pumping techniques,

...