Good-for-Games Automata: State of the art and perspectives

Denis Kuperberg

CNRS, LIP, ENS Lyon

Dagstuhl Seminar
Unambiguity in Automata Theory

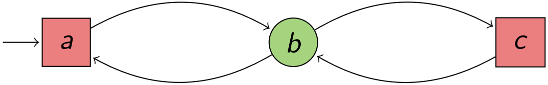
Games

Two Players: Eve Adam

Games

Two Players:

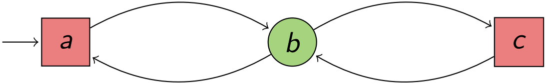
Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Games

Two Players:

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.

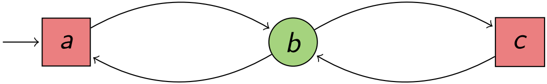


Initial vertex: $v_0 \in V$.

Games

Two Players: Eve Adam

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



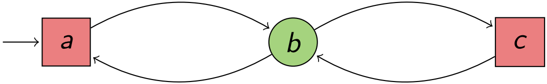
Initial vertex: $v_0 \in V$.

Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Games

Two Players: Eve Adam

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Initial vertex: $v_0 \in V$.

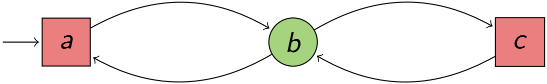
Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Winning Condition: $W \subseteq V^\omega$.

Games

Two Players: Eve Adam

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Initial vertex: $v_0 \in V$.

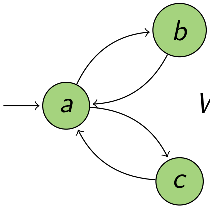
Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Winning Condition: $W \subseteq V^\omega$.

Eve wins a play π if $\pi \in W$

ω-regular games

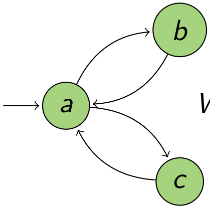
Winning condition W: an ω-regular language.



$$W = (a^* b a^* c)^\omega$$

ω-regular games

Winning condition W: an ω-regular language.



$W = (a^* b a^* c)^\omega$

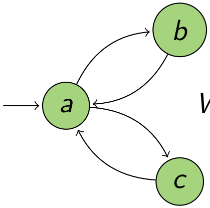
Particular case: Parity games

W is a parity condition: each vertex has a color in $\mathbb{N}$, the maximal color appearing infinitely often must be even.

Büchi=Parity $[1, 2]$ CoBüchi=Parity $[0, 1]$.

ω-regular games

Winning condition W: an ω-regular language.



$W = (a^*ba^*c)^\omega$

Particular case: Parity games

W is a parity condition: each vertex has a color in $\mathbb{N}$, the maximal color appearing infinitely often must be even.

Büchi=Parity $[1, 2]$ CoBüchi=Parity $[0, 1]$.

Theorem (Positional Determinacy [Emerson, Jutla '91])

In a parity game, Eve or Adam has a ***positional*** *winning strategy.*

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?

Solution:

1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = \mathcal{A}_{\mathrm{Det}} \circ G$.

Theorem

G' has same winner as G.
σ_{pos} in G' gives σ in G with memory $\mathcal{A}_{\mathrm{Det}}$.
$\rightarrow$ ω-regular games are finite-memory determined.

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?

Solution:

1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = \mathcal{A}_{\mathrm{Det}} \circ G$.

Theorem

G' has same winner as G.
σ_{pos} in G' gives σ in G with memory $\mathcal{A}_{\mathrm{Det}}$.
$\rightarrow$ ω-regular games are finite-memory determined.

Application: Church Synthesis

Automatically build a program from a specification L
$\Leftrightarrow$ Solving a game with winning condition L.

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?

Solution:

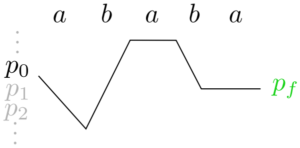
1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = \mathcal{A}_{\mathrm{Det}} \circ G$.

Theorem
G' has same winner as G.
σ_{pos} in G' gives σ in G with memory $\mathcal{A}_{\mathrm{Det}}$.
$\rightarrow$ ω-regular games are finite-memory determined.

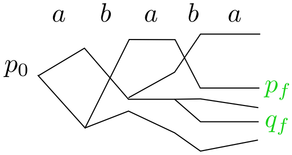
Application: Church Synthesis
Automatically build a program from a specification L
$\Leftrightarrow$ Solving a game with winning condition L.

Problem: Determinization is expensive. Maybe too strong ?

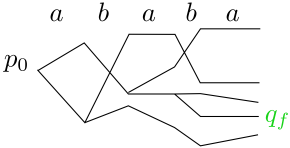
Good-for-Games Automata



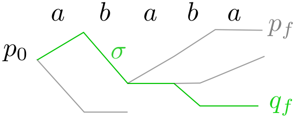
Deterministic



Non-deterministic



Unambiguous



Good-for-Games

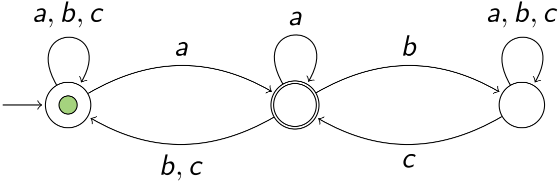
$\sigma : A^* \to Q$

Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters:

Eve: resolves non-deterministic choices for transitions



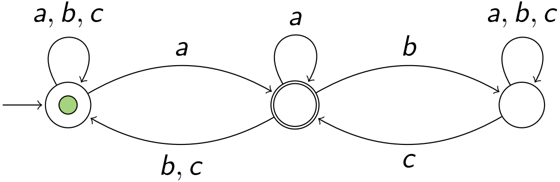
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: a

Eve: resolves non-deterministic choices for transitions

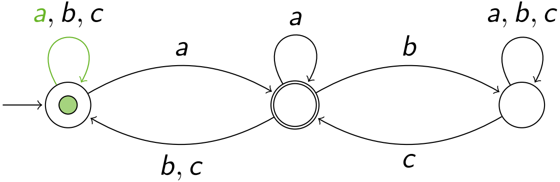


Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters: a

Eve: resolves non-deterministic choices for transitions



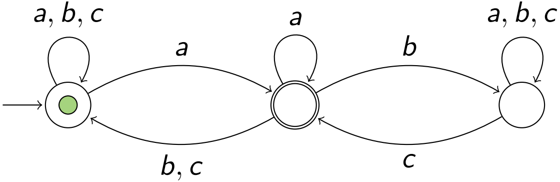
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \; a$

Eve: resolves non-deterministic choices for transitions

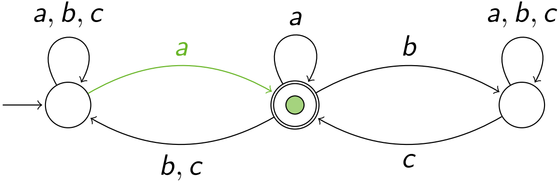


Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters: a a

Eve: resolves non-deterministic choices for transitions



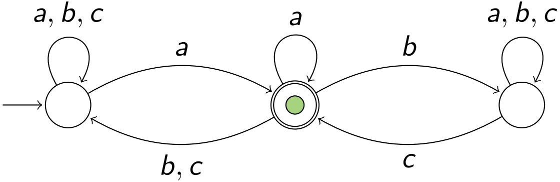
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \ \ a \ \ b$

Eve: resolves non-deterministic choices for transitions



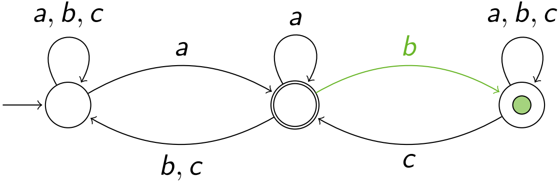
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: a a b

Eve: resolves non-deterministic choices for transitions



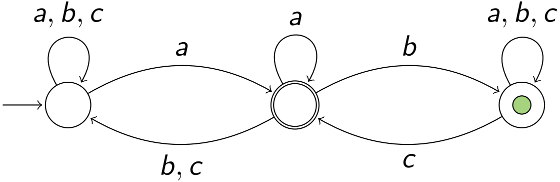
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \;\; a \;\; b \;\; c$

Eve: resolves non-deterministic choices for transitions



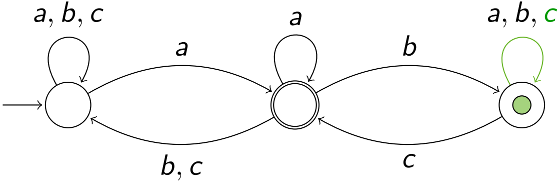
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \quad a \quad b \quad c$

Eve: resolves non-deterministic choices for transitions



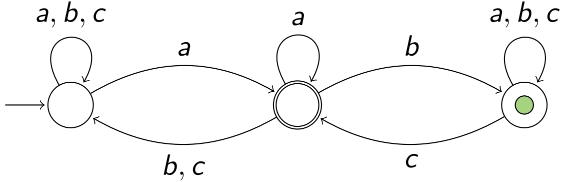
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \;\; a \;\; b \;\; c \;\; c$

Eve: resolves non-deterministic choices for transitions

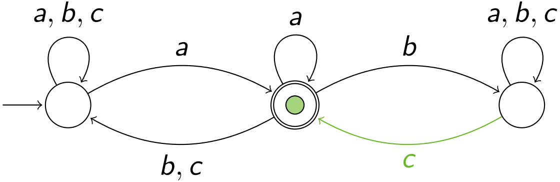


Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters: $a \quad a \quad b \quad c \quad c$

Eve: resolves non-deterministic choices for transitions

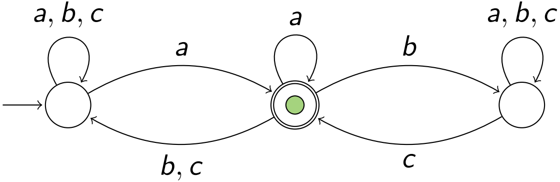


Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters: $a \;\; a \;\; b \;\; c \;\; c \;\; \ldots \quad = \;\; w$

Eve: resolves non-deterministic choices for transitions



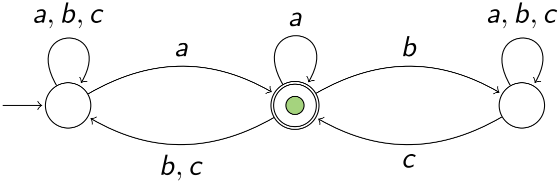
Eve wins if: $w \in L \Rightarrow$ Run accepting.

Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:
Adam plays letters: $a \;\; a \;\; b \;\; c \;\; c \;\; \ldots \;\; = \; w$

Eve: resolves non-deterministic choices for transitions



Eve wins if: $w \in L \Rightarrow$ Run accepting.

$\mathcal{A}$ GFG $\Leftrightarrow$ Eve wins the Letter game on $\mathcal{A}$
$\Leftrightarrow$ there is a strategy $\sigma_{\mathrm{GFG}} : A^* \to Q$ accepting all words of $L(\mathcal{A})$.

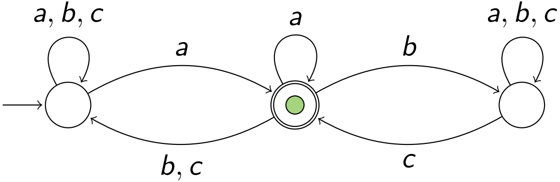
Definition of GFG via a game

$\mathcal{A}$ ND automaton on finite or infinite words.

Letter game of $\mathcal{A}$:

Adam plays letters: $a \;\; a \;\; b \;\; c \;\; c \;\dots \;\; = \; w$

Eve: resolves non-deterministic choices for transitions



Eve wins if: $w \in L \Rightarrow$ Run accepting.

$\mathcal{A}$ GFG $\Leftrightarrow$ Eve wins the Letter game on $\mathcal{A}$

$\Leftrightarrow$ there is a strategy $\sigma_{\mathrm{GFG}} : A^* \to Q$ accepting all words of $L(\mathcal{A})$.

Not a parity game! Only ω-regular.

Why "Good-for-games" ?

Theorem (Henzinger, Piterman '06)

Let $\mathcal{A}$ a parity GFG automaton, and G game with winning condition $L(\mathcal{A})$.
Then $\mathcal{A} \circ G$ (where Eve controls $\mathcal{A}$) is a parity game with same winner as G.

Proof: Eve can drive $\mathcal{A}$ according to σ_{GFG}.

Why “Good-for-games” ?

Theorem (Henzinger, Piterman ’06)

Let $\mathcal{A}$ a parity GFG automaton, and G game with winning condition $L(\mathcal{A})$.
Then $\mathcal{A} \circ G$ (where Eve controls $\mathcal{A}$) is a parity game with same winner as G.

Proof: Eve can drive $\mathcal{A}$ according to σ_{GFG}.

$\rightarrow$ We can use GFG instead of determinism to solve games.

Why "Good-for-games" ?

Theorem (Henzinger, Piterman '06)

Let $\mathcal{A}$ a parity GFG automaton, and G game with winning condition $L(\mathcal{A})$.
Then $\mathcal{A} \circ G$ (where Eve controls $\mathcal{A}$) is a parity game with same winner as G.

Proof: Eve can drive $\mathcal{A}$ according to σ_{GFG}.

$\rightarrow$ We can use GFG instead of determinism to solve games.

GFG automata can be defined as those enjoying this property.

Why "Good-for-games" ?

Theorem (Henzinger, Piterman '06)

Let $\mathcal{A}$ a parity GFG automaton, and G game with winning condition $L(\mathcal{A})$.
Then $\mathcal{A} \circ G$ (where Eve controls $\mathcal{A}$) is a parity game with same winner as G.

Proof: Eve can drive $\mathcal{A}$ according to σ_{GFG}.

$\rightarrow$ We can use GFG instead of determinism to solve games.

GFG automata can be defined as those enjoying this property.

Corollary

Church synthesis is in $\mathrm{PT{\small IME}}$ if the input is a GFG automaton.

Why “Good-for-games” ?

Theorem (Henzinger, Piterman ’06)

Let $\mathcal{A}$ a parity GFG automaton, and G game with winning condition $L(\mathcal{A})$.
Then $\mathcal{A} \circ G$ (where Eve controls $\mathcal{A}$) is a parity game with same winner as G.

Proof: Eve can drive $\mathcal{A}$ according to σ_{GFG}.

$\rightarrow$ We can use GFG instead of determinism to solve games.

GFG automata can be defined as those enjoying this property.

Corollary

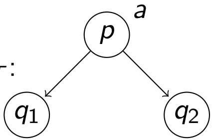
Church synthesis is in PTime if the input is a GFG automaton.

Remark: Synthesis is ExpTime-complete for nondet specification, and 2ExpTime-complete for LTL specification.

Good-for-Trees

$\mathcal{A}$ automaton on infinite words $\mapsto$ $\mathcal{A}_T$ automaton on infinite trees

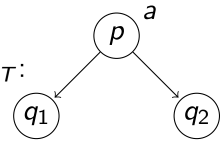
If $p \xrightarrow{a} q_1$ and $p \xrightarrow{a} q_2$ in $\mathcal{A}$, then put in $\mathcal{A}_T$:



Good-for-Trees

$\mathcal{A}$ automaton on infinite words $\mapsto$ $\mathcal{A}_T$ automaton on infinite trees

If $p \xrightarrow{a} q_1$ and $p \xrightarrow{a} q_2$ in $\mathcal{A}$, then put in $\mathcal{A}_T$:



Definition

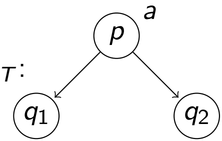
$\mathcal{A}$ is **Good-for-Trees** (GFT) if

$$L(\mathcal{A}_T) = \{t \mid \text{all branches of } t \text{ are in } L(\mathcal{A})\}$$

Good-for-Trees

$\mathcal{A}$ automaton on infinite words $\mapsto$ $\mathcal{A}_T$ automaton on infinite trees

If $p \xrightarrow{a} q_1$ and $p \xrightarrow{a} q_2$ in $\mathcal{A}$, then put in $\mathcal{A}_T$:



Definition

$\mathcal{A}$ is **Good-for-Trees** (GFT) if

$$L(\mathcal{A}_T) = \{t \mid \text{all branches of } t \text{ are in } L(\mathcal{A})\}$$

Theorem (Boker, K., Kupferman, Skrzypczak '13)

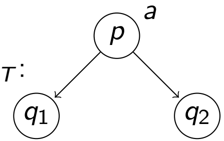
If rank of the trees $\geq$ size of the alphabet, then

$$\mathrm{GFG} = \mathrm{GFT}$$

Good-for-Trees

$\mathcal{A}$ automaton on infinite words $\mapsto$ $\mathcal{A}_T$ automaton on infinite trees

If $p \xrightarrow{a} q_1$ and $p \xrightarrow{a} q_2$ in $\mathcal{A}$, then put in $\mathcal{A}_T$:



Definition

$\mathcal{A}$ is **Good-for-Trees** (GFT) if

$$L(\mathcal{A}_T) = \{t \mid \text{all branches of } t \text{ are in } L(\mathcal{A})\}$$

Theorem (Boker, K., Kupferman, Skrzypczak '13)

If rank of the trees $\geq$ size of the alphabet, then

$$\mathrm{GFG} = \mathrm{GFT}$$

Hypothesis is necessary !

Link with determinism

Fact

Every deterministic automaton is GFG.

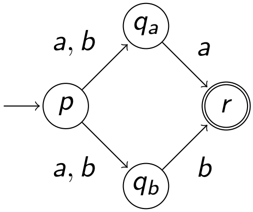
Link with determinism

Fact

Every deterministic automaton is GFG.

Some (unamb.) non-GFG automaton:

$$L = (a + b)(a + b)$$



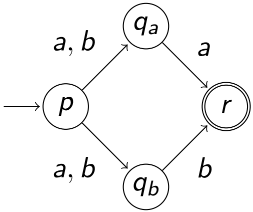
Link with determinism

Fact

Every deterministic automaton is GFG.

Some (unamb.) non-GFG automaton:

$$L = (a+b)(a+b)$$



Definition

Determinizable by Pruning (DBP):

Determinizable by removing some transitions.

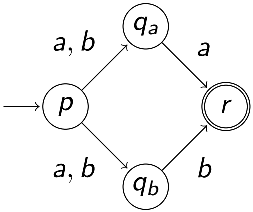
Link with determinism

Fact

Every deterministic automaton is GFG.

Some (unamb.) non-GFG automaton:

$$L = (a + b)(a + b)$$



Definition

Determinizable by Pruning (DBP):

Determinizable by removing some transitions.

Fact

DBP = *"GFG with a positional strategy".*

$\rightarrow$ *Every DBP automaton is GFG.*

Some GFG automata

Theorem

On finite words, $\mathrm{DBP} = \mathrm{GFG}$*.*

Proof: σ_{GFG}: always go to state accepting the maximal language.

Some GFG automata

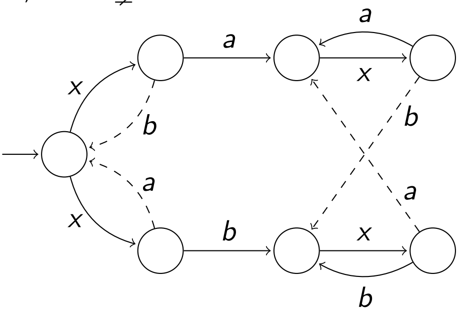
Theorem

On finite words, DBP $=$ GFG.

Proof: σ_{GFG}: always go to state accepting the maximal language.

Theorem ([Boker, K., Kupferman, Skrzypczak '13])

On infinite words, DBP $\subsetneq$ GFG.



A GFG coBüchi automaton for $(xa + xb)^*[(xa)^\omega + (xb)^\omega]$.

Algorithmic properties of GFG automata

Theorem (Easy inclusion checking)

If $\mathcal{A}$ is nondet and $\mathcal{B}$ is GFG, we can decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ in $\textsc{PTime}$.

Proof: Simulation game.

Algorithmic properties of GFG automata

Theorem (Easy inclusion checking)

If $\mathcal{A}$ is nondet and $\mathcal{B}$ is GFG, we can decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ in $\mathrm{PT{\scriptsize IME}}$.

Proof: Simulation game.

Theorem (Easy Union, Intersection)

If $\mathcal{A}$ and $\mathcal{B}$ are GFG, then their union and intersection using cartesian product are GFG.

Algorithmic properties of GFG automata

Theorem (Easy inclusion checking)

If $\mathcal{A}$ is nondet and $\mathcal{B}$ is GFG, we can decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ in $\mathrm{PT{\scriptsize IME}}$.

Proof: Simulation game.

Theorem (Easy Union, Intersection)

If $\mathcal{A}$ and $\mathcal{B}$ are GFG, then their union and intersection using cartesian product are GFG.

Theorem (Hard complementation)

If $\mathcal{A}$ is GFG for L and $\mathcal{B}$ is GFG for $\overline{L}$, then we can build in $\mathrm{PT{\scriptsize IME}}$ a deterministic automaton for L based on $\mathcal{A} \times \mathcal{B}$.

Algorithmic properties of GFG automata

Theorem (Easy inclusion checking)

If $\mathcal{A}$ is nondet and $\mathcal{B}$ is GFG, we can decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ in $\mathrm{PT{\small IME}}$.

Proof: Simulation game.

Theorem (Easy Union, Intersection)

If $\mathcal{A}$ and $\mathcal{B}$ are GFG, then their union and intersection using cartesian product are GFG.

Theorem (Hard complementation)

If $\mathcal{A}$ is GFG for L and $\mathcal{B}$ is GFG for $\overline{L}$, then we can build in $\mathrm{PT{\small IME}}$ a deterministic automaton for L based on $\mathcal{A} \times \mathcal{B}$.

Proof: Pos. Strategy in Letter game of $U = \mathcal{A} \times \mathcal{B}$ accepting all words.

How much memory is needed in the GFG strategy ?

Lemma

If $\mathcal{A}$ is GFG and $\mathcal{D}$ is a det. automaton for $L(\mathcal{A})$, then there is a strategy σ_{GFG} with memory $\mathcal{D}$.

Proof: Solve the letter game the classical way.

How much memory is needed in the GFG strategy ?

Lemma

If $\mathcal{A}$ is GFG and $\mathcal{D}$ is a det. automaton for $L(\mathcal{A})$, then there is a strategy σ_{GFG} with memory $\mathcal{D}$.

Proof: Solve the letter game the classical way.

Fact

If $\mathcal{A}$ is GFG with σ_{GFG} of memory M, then $\mathcal{D} = \mathcal{A} \times M$ is a det. automaton for $L(\mathcal{A})$.

How much memory is needed in the GFG strategy ?

Lemma

If $\mathcal{A}$ is GFG *and $\mathcal{D}$ is a det. automaton for $L(\mathcal{A})$, then there is a strategy σ_{GFG} with memory $\mathcal{D}$.*

Proof: Solve the letter game the classical way.

Fact

If $\mathcal{A}$ is GFG *with σ_{GFG} of memory M, then $\mathcal{D} = \mathcal{A} \times M$ is a det. automaton for $L(\mathcal{A})$.*

Conclusion:

Size of memory in $\sigma_{\mathrm{GFG}} \approx$ size of equivalent det. automaton

How much memory is needed in the GFG strategy ?

Lemma

If $\mathcal{A}$ is GFG and $\mathcal{D}$ is a det. automaton for $L(\mathcal{A})$, then there is a strategy σ_{GFG} with memory $\mathcal{D}$.

Proof: Solve the letter game the classical way.

Fact

If $\mathcal{A}$ is GFG with σ_{GFG} of memory M, then $\mathcal{D} = \mathcal{A} \times M$ is a det. automaton for $L(\mathcal{A})$.

Conclusion:

Size of memory in $\sigma_{\mathrm{GFG}} \approx$ size of equivalent det. automaton

Corollary

Det and GFG are equi-expressive for any acceptance condition.

How much memory is needed in the GFG strategy ?

Lemma

If $\mathcal{A}$ is GFG *and $\mathcal{D}$ is a det. automaton for $L(\mathcal{A})$, then there is a strategy σ_{GFG} with memory $\mathcal{D}$.*

Proof: Solve the letter game the classical way.

Fact

If $\mathcal{A}$ is GFG *with σ_{GFG} of memory M, then $\mathcal{D} = \mathcal{A} \times M$ is a det. automaton for $L(\mathcal{A})$.*

Conclusion:

Size of memory in $\sigma_{\text{GFG}} \approx$ size of equivalent det. automaton

Corollary

Det and GFG *are equi-expressive for any acceptance condition.*

Important Remark:

GFG automata can be used in algorithms without knowing σ_{GFG}.
The strategy σ_{GFG} “hides” the determinism.

State-blow-up for determinization

Finite words:
GFG=DBP, Determinization in PTIME [Löding].

State-blow-up for determinization

Finite words:
GFG=DBP, Determinization in PTime [Löding].

Büchi (Parity $[1,2]$):

- Simple exponential determinization: powerset not Safra.
- Determinization: $O(n^2)$ states and NP [K., Skrzypczak '15].
- Conjecture $O(n)$ states and in PTime.

State-blow-up for determinization

Finite words:
GFG=DBP, Determinization in PTIME [Löding].

Büchi (Parity $[1, 2]$):

- Simple exponential determinization: powerset not Safra.
- Determinization: $O(n^2)$ states and NP [K., Skrzypczak '15].
- Conjecture $O(n)$ states and in PTIME.

CoBüchi (Parity $[0, 1]$):
GFG automata can be exponentially more succinct than deterministic ones [K., Skrzypczak '15]

Exponential succinctness

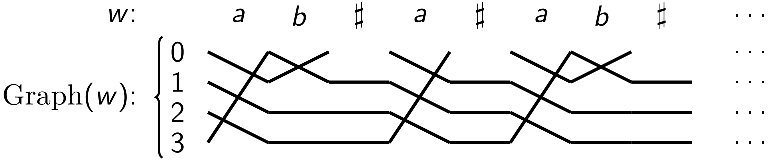
To define L_n, letters $\{a, b, \sharp\}$ act on $\{0, 1, \ldots, n-1\}$:

$$a : \; +1 \mod n \qquad b : \; 0 \leftrightarrow 1 \qquad \sharp : \text{“cuts” } 0$$

Exponential succinctness

To define L_n, letters $\{a, b, \sharp\}$ act on $\{0, 1, \ldots, n-1\}$:

$$a : \ +1 \mod n \qquad b : \ 0 \leftrightarrow 1 \qquad \sharp : \text{“cuts” } 0$$



$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}.$

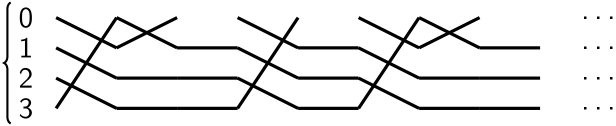
Exponential succinctness

To define L_n, letters $\{a, b, \sharp\}$ act on $\{0, 1, \ldots, n-1\}$:

$$a: \ +1 \mod n \qquad b: \ 0 \leftrightarrow 1 \qquad \sharp: \text{“cuts” } 0$$

w: a b $\sharp$ a $\sharp$ a b $\sharp$ $\cdots$

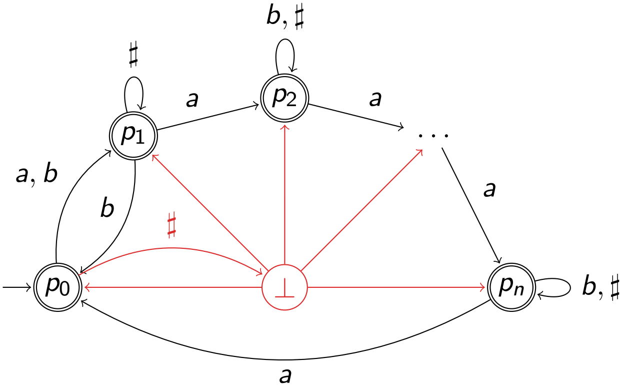
$\mathrm{Graph}(w)$:

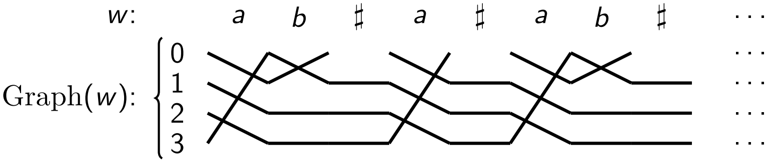


$$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}.$$

Lemma: Any Det Parity automaton for L_n needs $\dfrac{2^{\frac{n}{2}}}{n+1}$ states.

Small CoBüchi GFG for L_n





More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(ω-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]

- Strictly between DPDA and PDA
- Exponential ($<$DPDA) and doubly exponential ($>$PDA) gaps

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(ω-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]

- Strictly between DPDA and PDA
- Exponential ($<$DPDA) and doubly exponential ($>$PDA) gaps

Infinite trees: Guidable aut. [Colcombet+Löding '08, Skrzypczak '21]

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(ω-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]

- Strictly between DPDA and PDA
- Exponential ($<$DPDA) and doubly exponential ($>$PDA) gaps

Infinite trees: Guidable aut. [Colcombet+Löding '08, Skrzypczak '21]

I/O-Aware GFG [Faran, Kupferman '20]

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(ω-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]

- Strictly between DPDA and PDA
- Exponential ($<$DPDA) and doubly exponential ($>$PDA) gaps

Infinite trees: Guidable aut. [Colcombet+Löding '08, Skrzypczak '21]

I/O-Aware GFG [Faran, Kupferman '20]

$(\max, +)$ **automata** [Filiot, Jecker, Lhote, Pérez, Raskin '17]

More general frameworks

Co-invention: History-deterministic for **cost functions** [Colcombet '09]

HD vs GFG (in quantitative automata) [Boker, Lehtinen]

Alternating automata [Boker, Colcombet, K., Lehtinen, Skrzypczak]

- σ_{Eve} and σ_{Adam}
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(ω-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]

- Strictly between DPDA and PDA
- Exponential ($<$DPDA) and doubly exponential ($>$PDA) gaps

Infinite trees: Guidable aut. [Colcombet+Löding '08, Skrzypczak '21]

I/O-Aware GFG [Faran, Kupferman '20]

$(\max, +)$ **automata** [Filiot, Jecker, Lhote, Pérez, Raskin '17]

Discounted Sum, LimInf, LimSup [Boker, Lehtinen]

Building GFG automata

Some efforts:

- Incremental powerset construction [K', Majumdar '18]
- Fragment of CoBüchi languages [Iosti, K. '19]
- Minimization of GFG CoBüchi automata in PTIME [Abu Radi, Kupferman '20]

Det CoBüchi $\longmapsto$ Minimal GFG coBüchi

Building GFG automata

Some efforts:

- Incremental powerset construction [K', Majumdar '18]
- Fragment of CoBüchi languages [Iosti, K. '19]
- Minimization of GFG CoBüchi automata in PTime [Abu Radi, Kupferman '20]

Det CoBüchi $\longmapsto$ Minimal GFG coBüchi

Less ambitious goal: recognizing GFG automata.

Recognizing GFG automata

GFGness problem: input $\mathcal{A}$ a ND automaton, is it GFG?

Recognizing GFG automata

GFGness problem: input $\mathcal{A}$ a ND automaton, is it GFG?

Theorem ([Henzinger, Piterman '06])

We can solve the Letter game in $\textsc{ExpTime}$.

Recognizing GFG automata

GFGness problem: input $\mathcal{A}$ a ND automaton, is it GFG?

Theorem ([Henzinger, Piterman '06])

We can solve the Letter game in EXPTIME.

Proof:

- Compute a deterministic parity automaton for $L(\mathcal{A})$,
- Use it to transform the Letter game into a parity game G' of exponential size,
- Solve G'.

Recognizing GFG automata

GFGness problem: input $\mathcal{A}$ a ND automaton, is it GFG?

Theorem ([Henzinger, Piterman '06])

We can solve the Letter game in $\textsc{ExpTime}$.

Proof:

- Compute a deterministic parity automaton for $L(\mathcal{A})$,
- Use it to transform the Letter game into a parity game G' of exponential size,
- Solve G'.

So GFGness is decidable, but can we do better than $\textsc{ExpTime}$?

Abstracting the Letter game: finite words

Theorem (Löding)

The GFGness problem is in $\mathrm{PT{\scriptsize IME}}$ *for finite words automata.*

Abstracting the Letter game: finite words

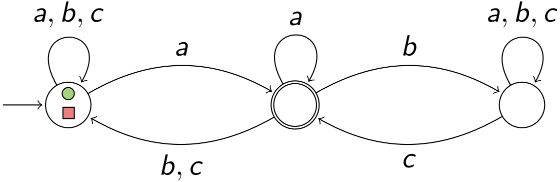
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters:

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

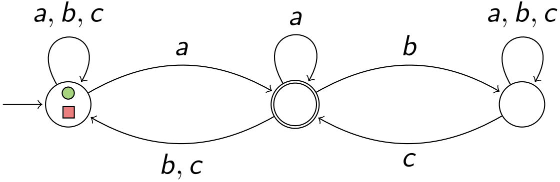
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

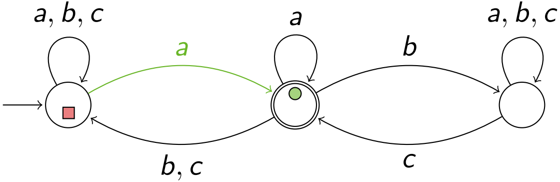
Theorem (Löding)

The GFGness problem is in PTIME for finite words automata.

The game G_1:

Adam plays letters: a

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

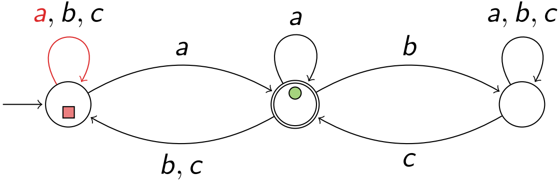
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

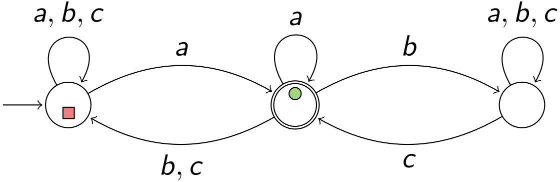
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a a

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

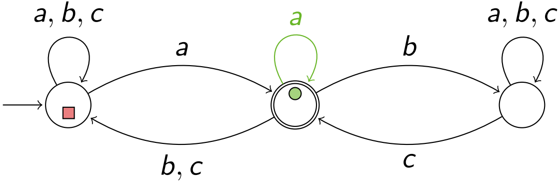
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a a

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

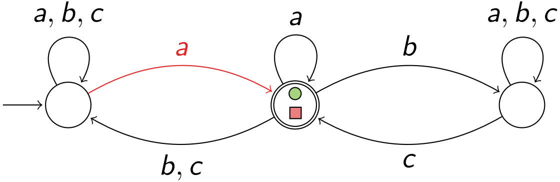
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a a

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

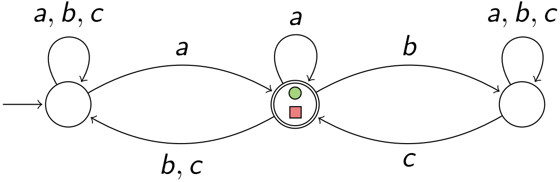
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a a b

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

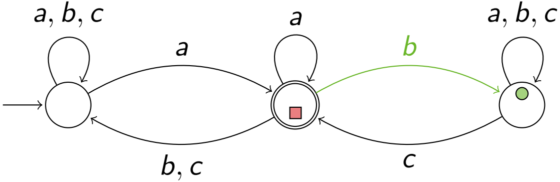
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: a a b

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

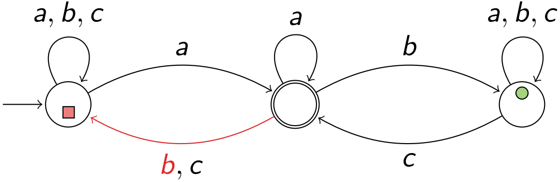
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

<u>The game G_1</u>:

Adam plays letters: a a b

Eve: moves one token ●, Adam: moves one token ■



Abstracting the Letter game: finite words

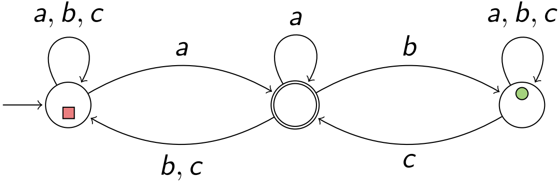
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: $a\ \ a\ \ b\ \ c$

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

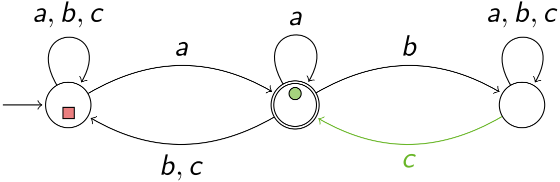
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: $a \;\; a \;\; b \;\; c$

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

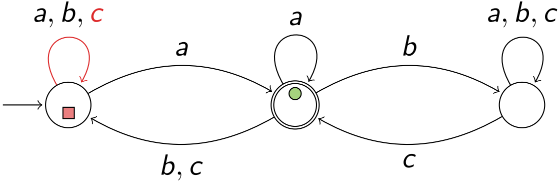
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: $a \;\; a \;\; b \;\; c$

Eve: moves one token, Adam: moves one token



Abstracting the Letter game: finite words

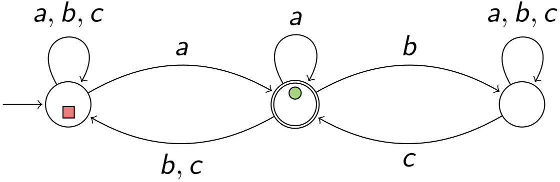
Theorem (Löding)

The GFGness problem is in PTime *for finite words automata.*

The game G_1:

Adam plays letters: $a \;\; a \;\; b \;\; c \; \ldots \quad = \; w$

Eve: moves one token ●, Adam: moves one token ■



Eve wins if at all times: ■ accepting $\Rightarrow$ ● accepting.

Abstracting the Letter game: finite words

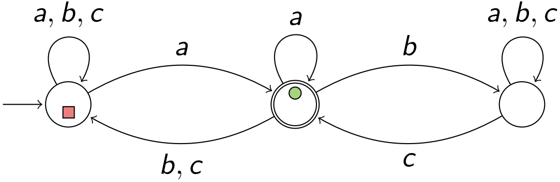
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: $a \; a \; b \; c \; \ldots \; = w$

Eve: moves one token ●, Adam: moves one token ■



Eve wins if at all times: ■ accepting ⇒ ● accepting.

Theorem: Eve wins $G_1 \Leftrightarrow \mathcal{A}$ is GFG.

Abstracting the Letter game: finite words

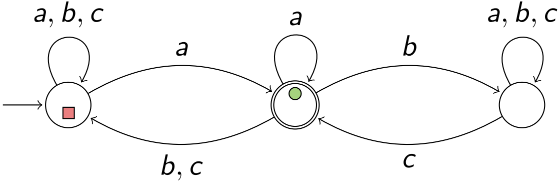
Theorem (Löding)

The GFGness problem is in $\textsc{PTime}$ *for finite words automata.*

<u>The game G_1</u>:

Adam plays letters: $a \;\; a \;\; b \;\; c \;\; \ldots \quad = \; w$

Eve: moves one token ●, Adam: moves one token ■



Eve wins if at all times: ■ accepting ⇒ ● accepting.

Theorem: Eve wins $G_1 \Leftrightarrow \mathcal{A}$ is GFG.

G_1 is a *safety* game, solvable in polynomial time.

Abstracting the Letter game: finite words

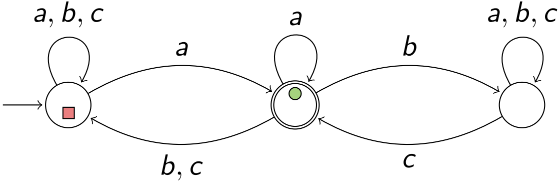
Theorem (Löding)

The GFGness problem is in PTIME *for finite words automata.*

The game G_1:

Adam plays letters: $a \;\; a \;\; b \;\; c \;\ldots \;\; = \; w$

Eve: moves one token ●, Adam: moves one token ■



Eve wins if at all times: ■ accepting ⇒ ● accepting.

Theorem: Eve wins $G_1 \Leftrightarrow \mathcal{A}$ is GFG.

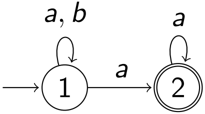
G_1 is a *safety* game, solvable in polynomial time.
Pos. Strategy $\mapsto$ Det. Automaton.

On infinite words

Fact

G_1 does not characterize GFG *Büchi (resp. CoBüchi) automata.*

not GFG:



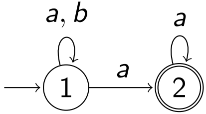
$(a+b)^* a^\omega$

On infinite words

Fact

G_1 does not characterize GFG *Büchi (resp. CoBüchi) automata.*

not GFG:



$(a+b)^* a^\omega$

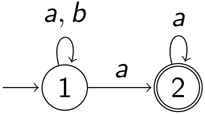
But Eve wins G_1: follow Adam's token one step behind.

On infinite words

Fact

G_1 does not characterize GFG Büchi (resp. CoBüchi) automata.

not GFG:



$(a+b)^* a^\omega$

But Eve wins G_1: follow Adam's token one step behind.

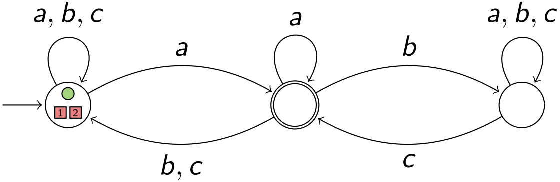
We need a better abstraction of the Letter game.

Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters:

Eve: moves one token ●, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$

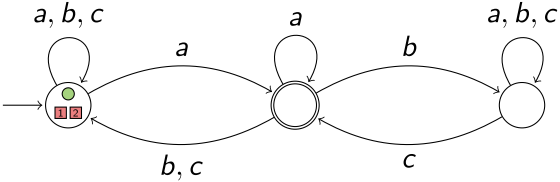


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a

Eve: moves one token, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$

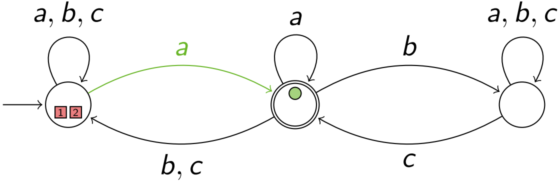


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a

Eve: moves one token, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$

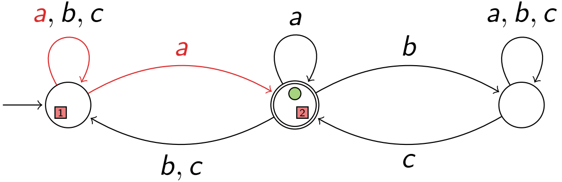


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a

Eve: moves one token, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$

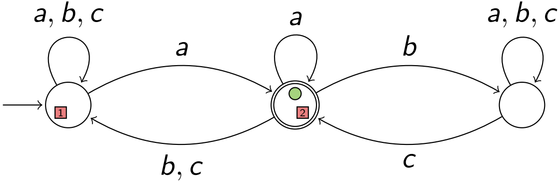


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a

Eve: moves one token, Adam: moves two tokens 1, 2

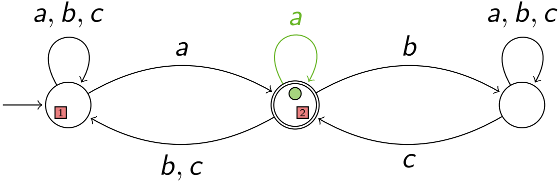


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a

Eve: moves one token, Adam: moves two tokens 1, 2

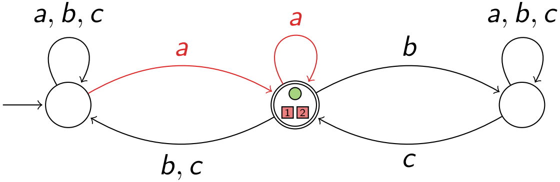


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a

Eve: moves one token, Adam: moves two tokens 1, 2

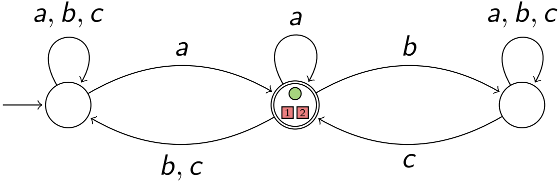


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a b

Eve: moves one token, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$

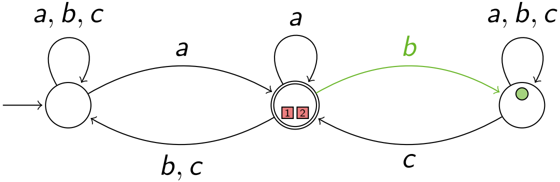


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a b

Eve: moves one token, Adam: moves two tokens 1, 2

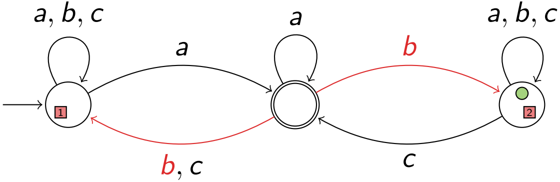


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a \quad a \quad b$

Eve: moves one token, Adam: moves two tokens 1, 2

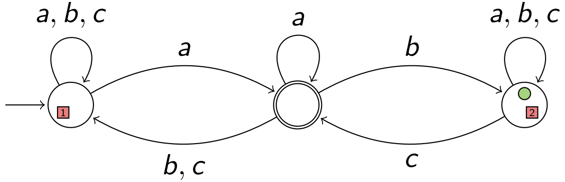


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a\ \ a\ \ b\ \ c$

Eve: moves one token, Adam: moves two tokens 1, 2

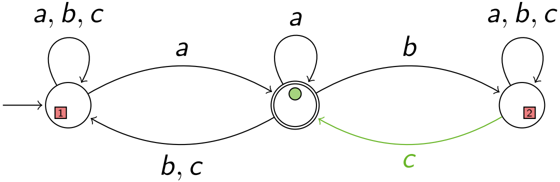


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a \quad a \quad b \quad c$

Eve: moves one token, Adam: moves two tokens 1, 2

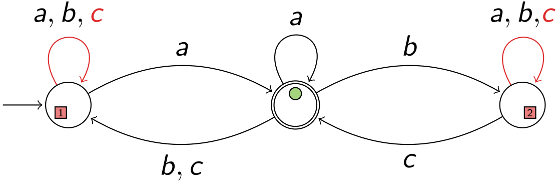


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: a a b c

Eve: moves one token, Adam: moves two tokens 1, 2

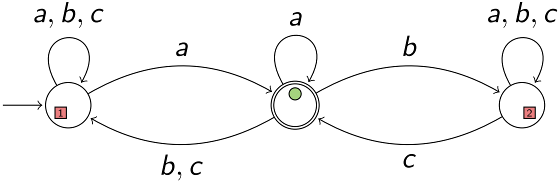


Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a \;\; a \;\; b \;\; c \; \ldots \quad = \; w$

Eve: moves one token ●, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$



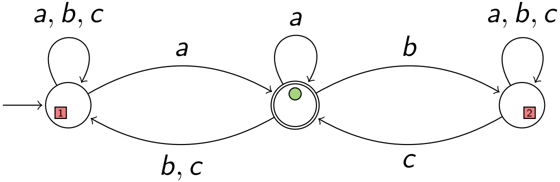
Eve wins if in the long run: $\boxed{1}$ or $\boxed{2}$ accepts $\Rightarrow$ ● accepts.

Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a \;\; a \;\; b \;\; c \;\; \ldots \quad = \;\; w$

Eve: moves one token ●, Adam: moves two tokens $\boxed{1}$, $\boxed{2}$



Eve wins if in the long run: $\boxed{1}$ or $\boxed{2}$ accepts $\Rightarrow$ ● accepts.

The G_2 Conjecture

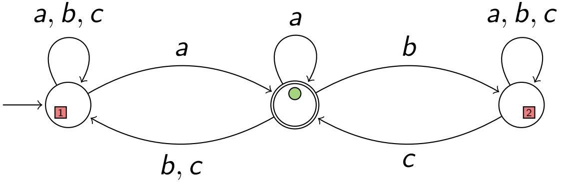
For all automata $\mathcal{A}$: Eve wins $G_2 \Leftrightarrow \mathcal{A}$ is GFG.

Abstracting the Letter game: infinite words

The game G_2:

Adam plays letters: $a \;\; a \;\; b \;\; c \; \ldots \quad = \; w$

Eve: moves one token ●,   Adam: moves two tokens $\boxed{1}$, $\boxed{2}$



Eve wins if in the long run: $\boxed{1}$ or $\boxed{2}$ accepts $\Rightarrow$ ● accepts.

The G_2 Conjecture

For all automata $\mathcal{A}$: Eve wins $G_2 \Leftrightarrow \mathcal{A}$ is GFG.

Solving G_2 is polynomial $\Rightarrow$ Efficient algorithm for GFGness.

Why G_2 is powerful: The game G_k

Game G_k: k tokens $\square$. Some $\boxed{i}$ accepts $\Rightarrow$ $\bigcirc$ must accept.

Why G_2 is powerful: The game G_k

Game G_k: k tokens. Some i accepts $\Rightarrow$ must accept.

Lemma

Eve wins $G_2 \Leftrightarrow$ *Eve wins* G_k *for all* $k \geq 2$.

Why G_2 is powerful: The game G_k

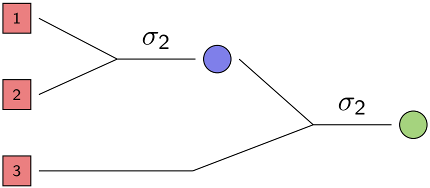
Game G_k: k tokens $\square$. Some $\boxed{i}$ accepts $\Rightarrow$ $\bigcirc$ must accept.

Lemma

Eve wins G_2 $\Leftrightarrow$ Eve wins G_k for all $k \geq 2$.

Proof sketch: $G_2 \Rightarrow G_3$

- play a *virtual token* $\bigcirc$ against $\boxed{1}$ and $\boxed{2}$.
- play G_2 strategy against $\bigcirc$ and $\boxed{3}$.



Explorable automata

Definition (k-Letter game)

k-letter game: Letter game where Eve moves k tokens instead of one. She wins if
$w \in L \Rightarrow$ at least one token follows an accepting run.

Definition

$\mathcal{A}$ is k-GFG if Eve wins the k-Letter game.
$\mathcal{A}$ is **Explorable** if it is k-GFG for some k.

Explorable automata

Definition (k-Letter game)

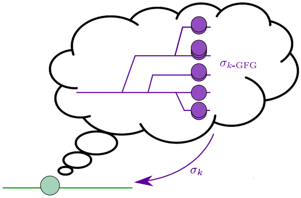
k-letter game: Letter game where Eve moves k tokens instead of one. She wins if
$w \in L \Rightarrow$ at least one token follows an accepting run.

Definition

$\mathcal{A}$ is k-GFG if Eve wins the k-Letter game.
$\mathcal{A}$ is **Explorable** if it is k-GFG for some k.

Theorem (Unpublished)

If $\mathcal{A}$ is explorable: Eve wins $G_2 \Leftrightarrow \mathcal{A}$ is GFG.



What is known about the G_2 conjecture

G_2 characterizes GFGness on

- Explorable parity automata [Unpublished]
- LimSup, LimInf automata [Boker, Lehtinen, on Arxiv]
- Büchi automata [Bagnol, K. '18]
- CoBüchi automata [Boker, K., Lehtinen, Skrzypcak, on Arxiv]

$\rightarrow$ GFGness is in PTime for these automata.

What is known about the G_2 conjecture

G_2 characterizes GFGness on

- Explorable parity automata [Unpublished]
- LimSup, LimInf automata [Boker, Lehtinen, on Arxiv]
- Büchi automata [Bagnol, K. '18]
- CoBüchi automata [Boker, K., Lehtinen, Skrzypcak, on Arxiv]

$\rightarrow$ GFGness is in PTIME for these automata.

For now, even with 3 parity ranks, the GFGNess problem is only known to be in EXPTIME.

What is known about the G_2 conjecture

G_2 characterizes GFGness on

- Explorable parity automata [Unpublished]
- LimSup, LimInf automata [Boker, Lehtinen, on Arxiv]
- Büchi automata [Bagnol, K. '18]
- CoBüchi automata [Boker, K., Lehtinen, Skrzypcak, on Arxiv]

$\rightarrow$ GFGness is in PTime for these automata.

For now, even with 3 parity ranks, the GFGNess problem is only known to be in ExpTime.

Theorem ([Boker, K., Lehtinen, Skrzypcak, on Arxiv])

If the G_2 conjecture is true on non-deterministic automata, it is true on alternating automata.

Main proof sketch for $G_2 \Rightarrow \mathrm{GFG}$ on Büchi

Assume for contradiction:

- Eve wins G_2, so Eve wins G_k with strategy σ_k, for a big k.
- Adam wins the Letter game with finite-memory strategy τ_{GFG}.

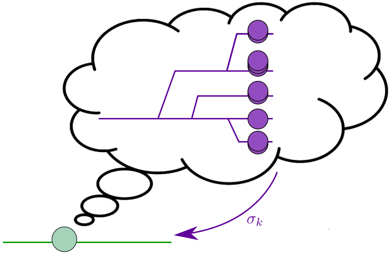
Main proof sketch for $G_2 \Rightarrow \mathrm{GFG}$ on Büchi

Assume for contradiction:

- Eve wins G_2, so Eve wins G_k with strategy σ_k, for a big k.
- Adam wins the Letter game with finite-memory strategy τ_{GFG}.

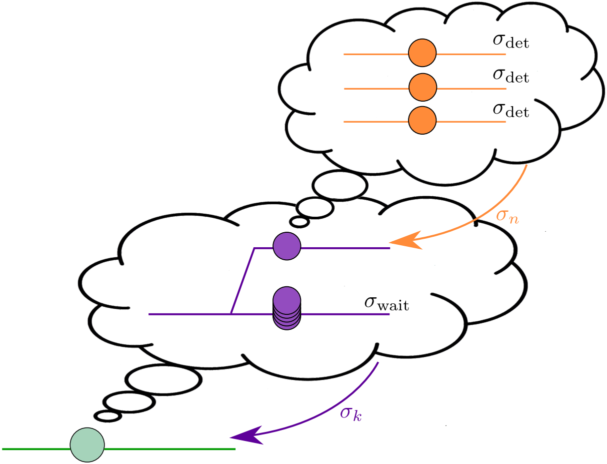
Idea for a strategy against τ_{GFG} in the Letter game:

- move k virtual tokens uniformly
- play σ_k against these k tokens



Trick: Word from τ_{GFG} $\Rightarrow$ one Büchi for some ● every M steps.
$\rightsquigarrow$ ● wins agains τ_{GFG}, contradiction.

G_2 for CoBüchi



Current and future work

- Studying GFG models in many frameworks.
  - Expressivity
  - Succinctness
  - Complexity
- Practical applications, experimental evaluations.
- Understanding k-GFG and Explorable automata.
- Prove or disprove the G_2 conjecture for parity automata.
- ...

Current and future work

- Studying GFG models in many frameworks.
  - Expressivity
  - Succinctness
  - Complexity
- Practical applications, experimental evaluations.
- Understanding k-GFG and Explorable automata.
- Prove or disprove the G_2 conjecture for parity automata.
- ...