Is your automaton good for playing games ?

Marc Bagnol, **Denis Kuperberg**

CNRS, LIP, ENS Lyon

University of Warwick
30/10/2018

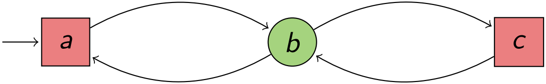
Games

Two Players: Eve Adam

Games

Two Players:

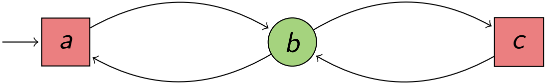
Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Games

Two Players:

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

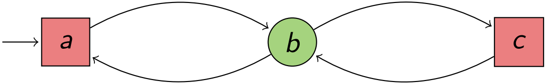


Initial vertex: $v_0 \in V$.

Games

Two Players:

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



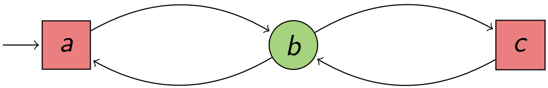
Initial vertex: $v_0 \in V$.

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: $v_0 \in V$.

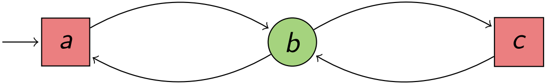
Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

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

Games

Two Players:

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



Initial vertex: $v_0 \in V$.

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

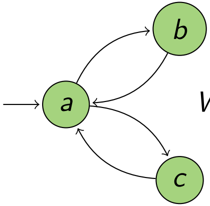
Winning Condition: $W \subseteq V^\omega$.

Eve wins a play π if $\pi \in W$

ω-regular games

Winning condition W: an ω-regular language.

Example:

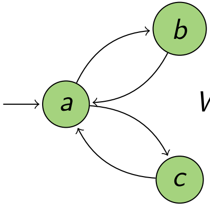


$$W = (a^* b a^* c)^\omega$$

ω-regular games

Winning condition W: an ω-regular language.

Example:



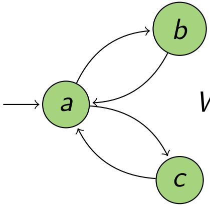
$$W = (a^* b a^* c)^\omega$$

Why study these games ?

ω-regular games

Winning condition W: an ω-regular language.

Example:



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

Why study these games ?

Applications in Logic, Verification, Proof theory, Complexity, ...

Church Synthesis Problem.

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

$\mathcal{A}_{\mathrm{Det}} \circ G$ has same winner as G.

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
$\mathcal{A}_{\mathrm{Det}} \circ G$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

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

$\mathcal{A}_{\mathrm{Det}} \circ G$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Definition (Henzinger, Piterman 2006)

$\mathcal{A}$ is Good-for-Games (GFG) if $\mathcal{A} \circ G$ has same winner as G, for any game G with winning condition $L(\mathcal{A})$.

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

$\mathcal{A}_{\mathrm{Det}} \circ G$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Definition (Henzinger, Piterman 2006)

$\mathcal{A}$ is Good-for-Games (GFG) if $\mathcal{A} \circ G$ has same winner as G, for any game G with winning condition $L(\mathcal{A})$.

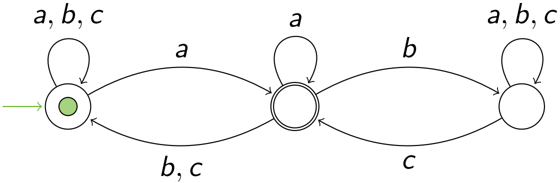
Also works for composition with infinite trees.

Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters:

Eve: resolves non-deterministic choices for transitions

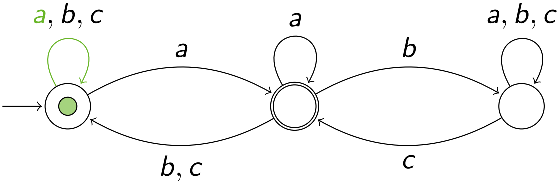


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: a

Eve: resolves non-deterministic choices for transitions

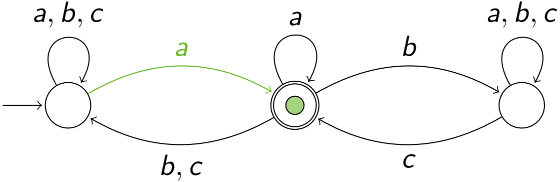


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: a a

Eve: resolves non-deterministic choices for transitions

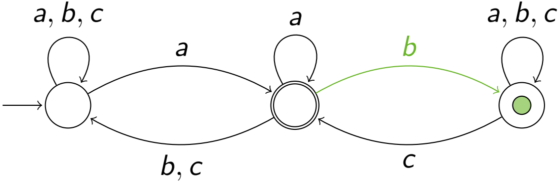


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: a a b

Eve: resolves non-deterministic choices for transitions

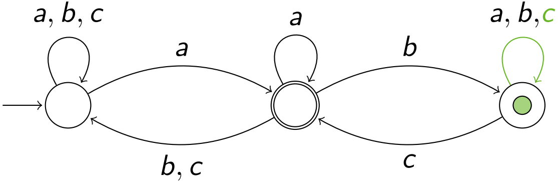


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

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

Eve: resolves non-deterministic choices for transitions

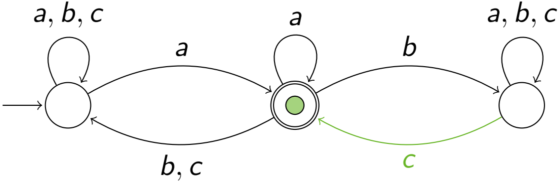


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

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

Eve: resolves non-deterministic choices for transitions

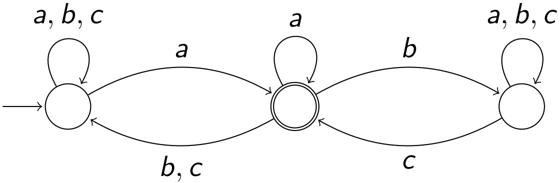


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

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

Eve: resolves non-deterministic choices for transitions



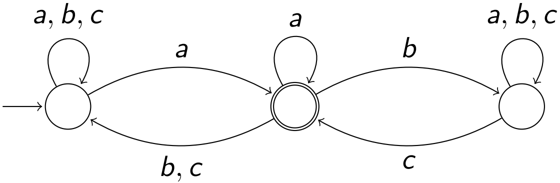
Eve wins if: $w \in L \Rightarrow$ Run accepting.

Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

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

Eve: resolves non-deterministic choices for transitions



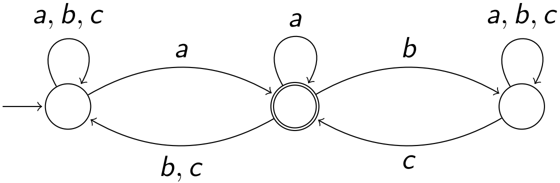
Eve wins if: $w \in L \Rightarrow$ Run accepting.

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

Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:
Adam plays letters: $a \;\; a \;\; b \;\; c \;\; c \;\; \dots \quad = \; w$

Eve: resolves non-deterministic choices for transitions



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

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

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

A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

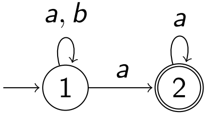
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG***.*

Fact

*There are some non-***GFG** *automata:*



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

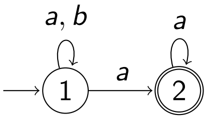
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:*



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

Adam plays $aaa\dots$ until Eve goes to 2, then ba^ω.

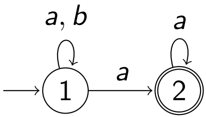
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:*



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

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

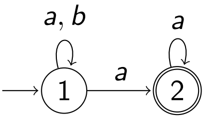
$\sigma_{\mathrm{GFG}} \leftrightarrows$ *Det automaton.*

A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:* 

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

Adam plays *aaa . . .* until Eve goes to 2, then ba^ω.

Fact

$\sigma_{\mathrm{GFG}} \leftrightarrows$ *Det automaton.*

Theorem (Boker, K., Kupferman, Skrzypczak 2013)

GFG for L + GFG for $\overline{L} \rightarrow$ *Det for L*

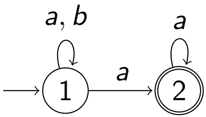
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG***.*

Fact

*There are some non-***GFG** *automata:*



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

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

$\sigma_{\mathrm{GFG}} \leftrightarrows$ *Det automaton.*

Theorem (Boker, K., Kupferman, Skrzypczak 2013)

GFG for L + *GFG for* $\overline{L} \rightarrow$ *Det for* L

Fact

If $\mathcal{A}$ *is* **GFG***, no need to know its strategy* σ_{GFG} *to solve* $\mathcal{A} \circ G$ *!*

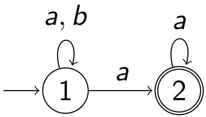
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:*



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

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

$\sigma_{\mathrm{GFG}} \leftrightarrows$ *Det automaton.*

Theorem (Boker, K., Kupferman, Skrzypczak 2013)

GFG for L + *GFG for* $\overline{L} \rightarrow$ *Det for* L

Fact

If $\mathcal{A}$ *is* **GFG**, *no need to know its strategy* σ_{GFG} *to solve* $\mathcal{A} \circ G$ *!*

$\rightarrow$ we can ignore the strategy and still use $\mathcal{A}$ as GFG in algorithms.

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

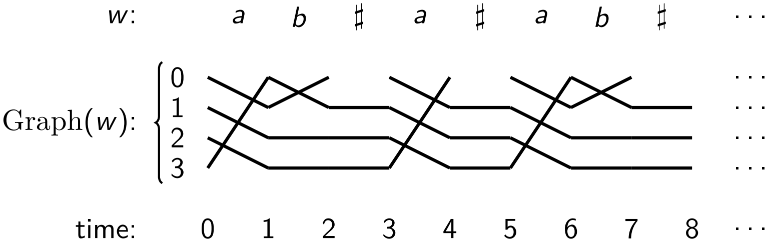
GFG *parity automata can be exponentially more succinct than deterministic ones.*

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

GFG *parity automata can be exponentially more succinct than deterministic ones.*



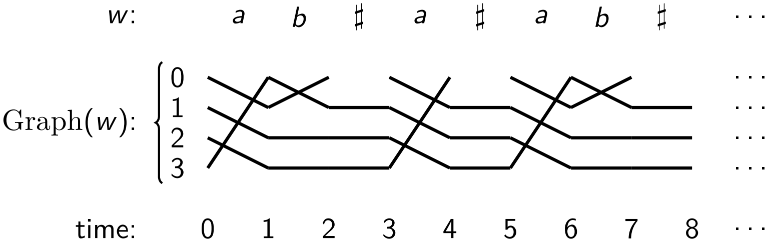
$$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}.$$

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

GFG *parity automata can be exponentially more succinct than deterministic ones.*

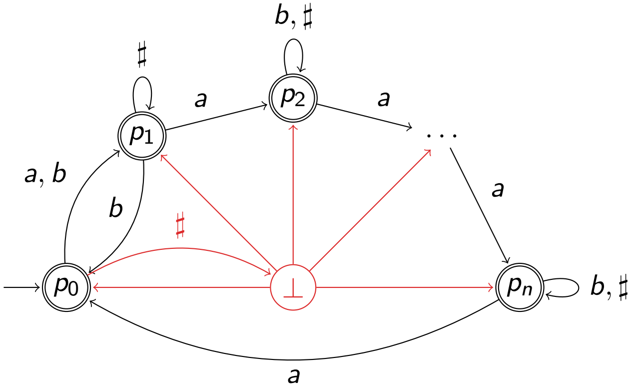


$$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}.$$

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

A small GFG automaton for L_n

GFG coBüchi automaton:



Idea: Try paths one after the other.

Recognizing GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Recognizing GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Theorem (Löding)

The GFGness problem is in P for reachability/safety automata.

Theorem (K., Skrzypczak 2015)

The GFGness problem is in P for coBüchi automata.
Parity Games $\equiv$ GFGness for universal Parity automata.

Recognizing GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Theorem (Löding)

The GFGness problem is in P for reachability/safety automata.

Theorem (K., Skrzypczak 2015)

The GFGness problem is in P for coBüchi automata.
Parity Games $\equiv$ GFGness for universal Parity automata.

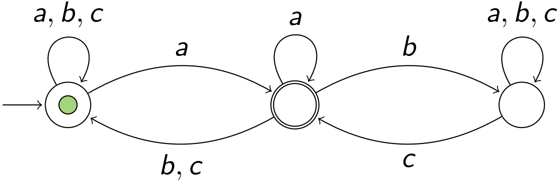
This talk:

Theorem (Bagnol, K. 2018)

The GFGness problem is in P for Büchi automata.

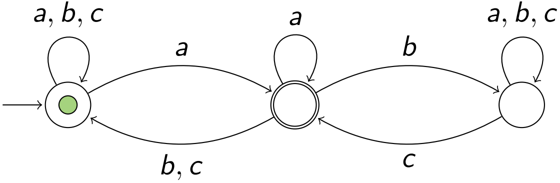
Via the GFG game

Recall: $\mathcal{A}$ is GFG $\Leftrightarrow$ Eve wins the GFG game on $\mathcal{A}$.



Via the GFG game

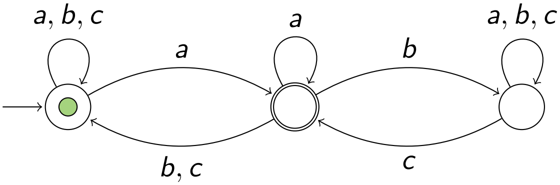
Recall: $\mathcal{A}$ is GFG $\Leftrightarrow$ Eve wins the GFG game on $\mathcal{A}$.



Solve the GFG game ?

Via the GFG game

Recall: $\mathcal{A}$ is GFG $\Leftrightarrow$ Eve wins the GFG game on $\mathcal{A}$.

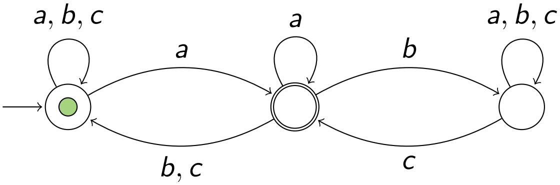


Solve the GFG game ?

Acceptance condition of the form "$u \in L \Rightarrow$run accepting"

Via the GFG game

Recall: $\mathcal{A}$ is GFG $\Leftrightarrow$ Eve wins the GFG game on $\mathcal{A}$.



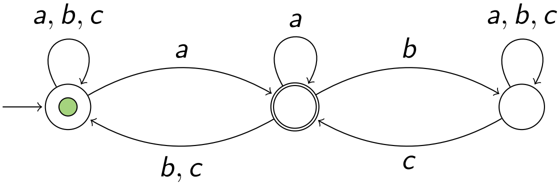
Solve the GFG game ?

Acceptance condition of the form "$u \in L \Rightarrow$run accepting"

$\rightarrow$ We need a **GFG** automaton for L!

Via the GFG game

Recall: $\mathcal{A}$ is GFG $\Leftrightarrow$ Eve wins the GFG game on $\mathcal{A}$.



Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"

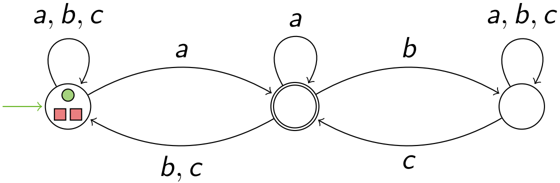
$\rightarrow$ We need a **GFG** automaton for L!

Upper bound: EXPTIME (by computing $\mathcal{A}_{\mathrm{Det}}$)

Abstracting the GFG game

The game G_2:
Adam plays letters:

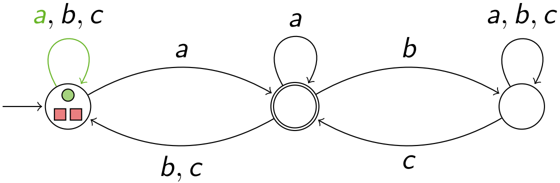
Eve: moves one token Adam: moves two tokens



Abstracting the GFG game

The game G_2:
Adam plays letters: a

Eve: moves one token Adam: moves two tokens

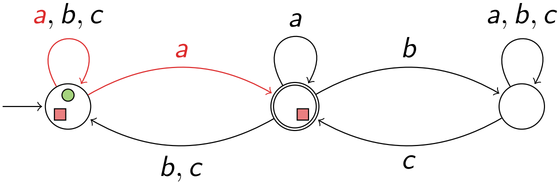


Abstracting the GFG game

The game G_2:

Adam plays letters: a

Eve: moves one token Adam: moves two tokens

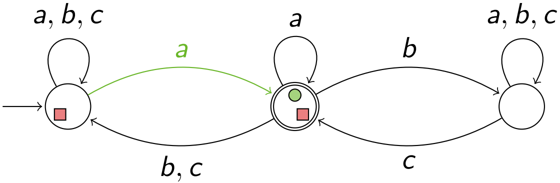


Abstracting the GFG game

The game G_2:

Adam plays letters: a a

Eve: moves one token Adam: moves two tokens

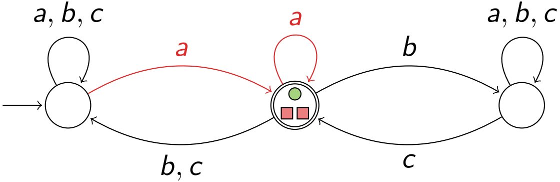


Abstracting the GFG game

The game G_2:

Adam plays letters: a a

Eve: moves one token Adam: moves two tokens

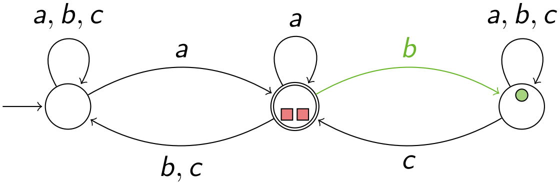


Abstracting the GFG game

The game G_2:

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

Eve: moves one token Adam: moves two tokens

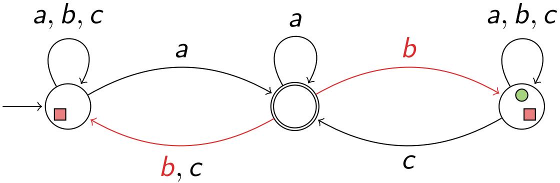


Abstracting the GFG game

The game G_2:

Adam plays letters: a a b

Eve: moves one token Adam: moves two tokens

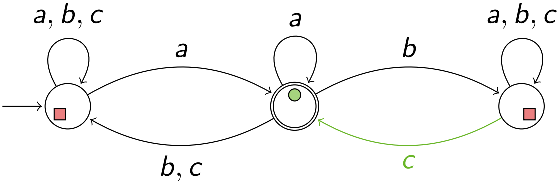


Abstracting the GFG game

The game G_2:

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

Eve: moves one token Adam: moves two tokens

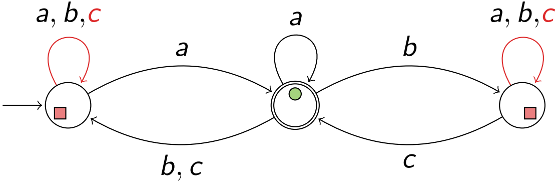


Abstracting the GFG game

The game G_2:

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

Eve: moves one token Adam: moves two tokens

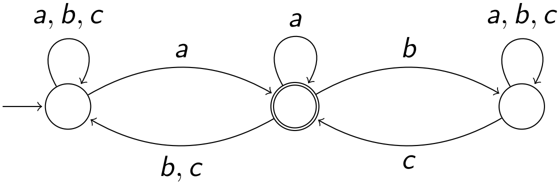


Abstracting the GFG game

The game G_2:

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

Eve: moves one token Adam: moves two tokens



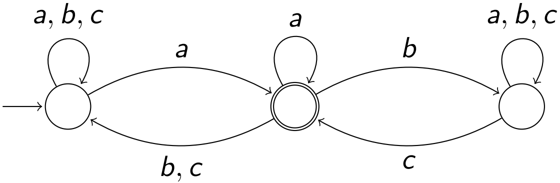
Eve wins if: $\blacksquare_1$ or $\blacksquare_2$ accepts $\Rightarrow$ $\bullet$ accepts.

Abstracting the GFG game

The game G_2:

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

Eve: moves one token Adam: moves two tokens



Eve wins if: ■$_1$ or ■$_2$ accepts $\Rightarrow$ ● accepts.

Goal: Show that Eve wins $G_2 \Leftrightarrow$ Eve wins the GFG game.

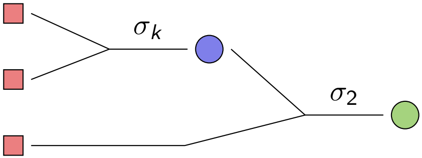
Proof sketch

Lemma

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

Proof sketch: $G_2 \Rightarrow G_3$

- play a *virtual token* ● against the first 2 ■
- play G_2 strategy against ● and last ■



Main proof sketch for $G_2 \Leftrightarrow \mathrm{GFG}$

Assume:

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

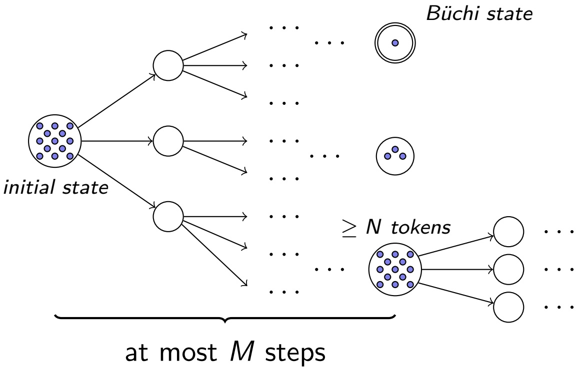
Main proof sketch for $G_2 \Leftrightarrow \mathrm{GFG}$

Assume:

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

Build strategy for Eve against τ_{GFG}:

- move k virtual tokens ● against τ_{GFG}
- play σ_k against these k tokens



Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
Partial k-Determinization of $\mathcal{A}_{\mathrm{ND}}$
Increase k until $\mathcal{A}$ is **GFG**.
Worst case: $k = |\mathcal{A}|$, reach the full determinization construction.

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
Partial k-Determinization of $\mathcal{A}_{\mathrm{ND}}$
Increase k until $\mathcal{A}$ is **GFG**.
Worst case: $k = |\mathcal{A}|$, reach the full determinization construction.

For **efficiency**: solution to the GFGness problem is needed !
So for now only efficient for coBüchi (and Büchi)

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
Partial k-Determinization of $\mathcal{A}_{\mathrm{ND}}$
Increase k until $\mathcal{A}$ is **GFG**.
Worst case: $k = |\mathcal{A}|$, reach the full determinization construction.

For **efficiency**: solution to the GFGness problem is needed !
So for now only efficient for coBüchi (and Büchi)

Directly from LTL formula φ [Iosti, K. (unpublished)]
Adapted for Chruch Synthesis
For now: Fragment of coBüchi languages

Worst-case blowup result: GFG is doubly exponential in $|\varphi|$.

Heuristic and modular, can be combined with other approaches.

Conclusion

Results

- **GFG** automata are intermediate between **Det** and **ND**.
- They are useful to solve complex games.
- They can bring an exponential advantage compared to determinism.
- For Büchi and coBüchi conditions, they are recognized efficiently.

Perspectives

- $G_2 \Leftrightarrow$ GFG for Parity automata ?
- Complexity of the Parity GFGness Problem ?
- New ways to build **GFG** automata ?
- Generalization to other models (alternating, pushdown, ...) ?