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

Denis Kuperberg

CNRS, LIP, ENS Lyon

Dagstuhl Seminar
Unambiguity in Automata Theory
Two Players: Eve, Adam

Arena: finite graph $G = (V, E)$, with $V = V_a \sqcup V_b$.

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 $\pi$ if $\pi \in W$. 
Games

Two Players: 

Arena: finite graph $G = (V, E)$, with $V = V_{\text{green}} \cup V_{\text{red}}$.

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 $\pi$ if $\pi \in W$. 
Games

Two Players:

- Eve
- Adam

Arena: finite graph $G = (V, E)$, with $V = V_\circ \cup V_\square$.

Initial vertex: $v_0 \in V$. 

Winning Condition: $W \subseteq V_\omega$. Eve wins a play $\pi$ if $\pi \in W$. 
Games

Two Players: Eve ✑ Adam

Arena: finite graph $G = (V, E)$, with $V = V_0 \cup V_1$.

Initial vertex: $v_0 \in V$.

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

Arena: finite graph $G = (V, E)$, with $V = V_0 \cup V_1$.

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 $V = V_\circ \cup V_\square$.

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 $\pi$ if $\pi \in W$. 
$\omega$-regular games

Winning condition $W$: an $\omega$-regular language.

$$W = (a^* ba^* c)^\omega$$
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 $[1, 2]$ \hspace{1cm} CoBüchi=Parity $[0, 1]$. 
\( \omega \)-regular games

Winning condition \( W \): an \( \omega \)-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.

\begin{align*}
\text{Büchi} = \text{Parity } [1, 2] & \quad \text{CoBüchi} = \text{Parity } [0, 1].
\end{align*}

**Theorem (Positional Determinacy [Emerson, Jutla ’91])**

In a parity game, Eve or Adam has a **positional** winning strategy.
Solving an $\omega$-regular game

Input: $G$ game with $\omega$-regular winning condition $W \subseteq V^\omega$.

Question: Who wins $G$? How?
Solving an \( \omega \)-regular game

**Input:** \( G \) game with \( \omega \)-regular winning condition \( W \subseteq V^\omega \).

**Question:** Who wins \( G \)? How?

**Solution:**

1. Build Det Parity automaton \( A_{\text{Det}} \) for \( W \),
2. Solve the parity game \( G' = A_{\text{Det}} \circ G \).

**Theorem**

\( G' \) has same winner as \( G \).

\( \sigma_{\text{pos}} \) in \( G' \) gives \( \sigma \) in \( G \) with memory \( A_{\text{Det}} \).

\( \rightarrow \) \( \omega \)-regular games are finite-memory determined.

Application: Church Synthesis

Automatically build a program from a specification \( L \iff W \).
Solving an \( \omega \)-regular game

**Input:** \( G \) game with \( \omega \)-regular winning condition \( W \subseteq V^\omega \).

**Question:** Who wins \( G \)? How?

**Solution:**

1. Build Det Parity automaton \( A_{\text{Det}} \) for \( W \),
2. Solve the parity game \( G' = A_{\text{Det}} \circ G \).

**Theorem**

\( G' \) has same winner as \( G \).

\( \sigma_{\text{pos}} \) in \( G' \) gives \( \sigma \) in \( G \) with memory \( A_{\text{Det}} \).

\( \rightarrow \ \omega \)-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 $\omega$-regular game

**Input:** $G$ game with $\omega$-regular winning condition $W \subseteq V^\omega$.

**Question:** Who wins $G$? How?

**Solution:**
1. Build Det Parity automaton $A_{Det}$ for $W$,
2. Solve the parity game $G' = A_{Det} \circ G$.

**Theorem**
$G'$ has same winner as $G$.
$\sigma_{pos}$ in $G'$ gives $\sigma$ in $G$ with memory $A_{Det}$.
$\rightarrow$ $\omega$-regular games are finite-memory determined.

**Application: Church Synthesis**
Automatically build a program from a specification $L$
$\iff$ 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

\( \sigma : A^* \rightarrow Q \)
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of** $\mathcal{A}$:

**Adam** plays letters:

**Eve**: resolves non-deterministic choices for transitions

$w \in L \Rightarrow$ Run accepting.

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

Not a parity game! Only $\omega$-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of $\mathcal{A}$:**
Adam plays letters: $a$
Eve: resolves non-deterministic choices for transitions

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

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

Not a parity game! Only $\omega$-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of \( A \):**
- **Adam** plays letters: \( a \)
- **Eve**: resolves non-deterministic choices for transitions

\[ w \in L \Rightarrow \text{Run accepting.} \]

A GFG \( \Leftrightarrow \) Eve wins the Letter game on \( A \) \( \Leftrightarrow \) there is a strategy \( \sigma \) GFG:

\[ A^* \rightarrow Q \text{ accepting all words of } L(A). \]

Not a parity game! Only \( \omega \)-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of $A$:**
Adam plays letters: $a \ a$

Eve: resolves non-deterministic choices for transitions
Definition of GFG via a game

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

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

Not a parity game! Only $\omega$-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of A:**

Adam plays letters: $a$, $a$, $b$

Eve: resolves non-deterministic choices for transitions

![Diagram of a nondeterministic automaton with transitions labeled $a$, $b$, and $c$.]
Definition of GFG via a game

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

Not a parity game! Only $\omega$-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of \( A \):**
- **Adam** plays letters: \( a \ a \ b \ c \)
- **Eve**: resolves non-deterministic choices for transitions

\[
\text{Eve wins if: } w \in L \Rightarrow \text{Run accepting.}
\]

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

Not a parity game! Only \( \omega \)-regular.
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of $A$:**

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

Eve: resolves non-deterministic choices for transitions

---

![Diagram](image)
Definition of GFG via a game

A ND automaton on finite or infinite words.

**Letter game of A:**
Adam plays letters: \( a \ a \ b \ c \ c \)
Eve: resolves non-deterministic choices for transitions

Not a parity game! Only \( \omega \)-regular.
Definition of GFG via a game

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

$\mathcal{A}$ is GFG $\iff$ Eve wins the Letter game on $\mathcal{A} \iff$ there is a strategy $\sigma_{\text{GFG}} : \mathcal{A}^\omega \rightarrow Q$ accepting all words of $\mathcal{L}(\mathcal{A})$.

Not a parity game! Only $\omega$-regular.
Definition of GFG via a game

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.
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

\[
\begin{align*}
& a, b, c \\
& a \quad a \\
& b, c \\
& b \\
& c \\
& a, b, c
\end{align*}
\]

Eve wins if: \( w \in L \Rightarrow \text{Run accepting.} \)

\( \mathcal{A} \) GFG \iff Eve wins the Letter game on \( \mathcal{A} \)

\iff there is a strategy \( \sigma_{\text{GFG}} : \mathcal{A}^* \rightarrow Q \) accepting all words of \( L(\mathcal{A}) \).
Definition of GFG via a game

A ND automaton on finite or infinite words.

Letter game of $A$:
Adam plays letters: $a\ a\ b\ c\ c\ \ldots\ =\ w$
Eve: resolves non-deterministic choices for transitions

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

Not a parity game! Only $\omega$-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 $\sigma_{GFG}$. 
Why “Good-for-games”?

**Theorem (Henzinger, Piterman ’06)**

Let $A$ a parity GFG automaton, and $G$ game with winning condition $L(A)$.

Then $A \circ G$ (where Eve controls $A$) is a parity game with same winner as $G$.

**Proof:** Eve can drive $A$ according to $\sigma_{GFG}$.

$\rightarrow$ 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 $\sigma_{GFG}$.

→ 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 $\sigma_{GFG}$.

$\rightarrow$ 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.*
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 \( \sigma_{\text{GFG}} \).

→ 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 \( \text{ExpTime-complete} \) for nondet specification, and \( 2\text{ExpTime-complete} \) for LTL specification.
Good-for-Trees

A automaton on infinite words $\leftrightarrow A_T$ automaton on infinite trees

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

\[
p \xrightarrow{a} q_1 \quad q_2
\]
Good-for-Trees

$A$ automaton on infinite words $\mapsto A_T$ automaton on infinite trees

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

```
  p
 /\   /
\  / \  /
 q1 q2
```

Definition

$A$ is Good-for-Trees (GFT) if

$$L(A_T) = \{ t \mid \text{all branches of } t \text{ are in } L(A) \}$$
Good-for-Trees

A automaton on infinite words $\mapsto A_T$ automaton on infinite trees

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

Definition

$A$ is Good-for-Trees (GFT) if

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

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

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

$$\text{GFG} = \text{GFT}$$
**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 \( \mathcal{A}_T \):

\[
\begin{array}{c}
p \xrightarrow{a} \\
\quad \downarrow \\
q_1 \quad q_2
\end{array}
\]

**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*

\[ \text{GFG} = \text{GFT} \]

Hypothesis is necessary!
Fact

Every deterministic automaton is GFG.
Fact

Every deterministic automaton is GFG.

Some (unamb.) non-GFG automaton:

\[ L = (a + b)(a + b) \]
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”.
\[ \rightarrow \text{Every DBP automaton is GFG.} \]
Some GFG automata

**Theorem**

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

**Proof:** $\sigma_{\text{GFG}}$: always go to state accepting the maximal language.
Some **GFG** automata

**Theorem**

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

**Proof:** $\sigma_{\text{GFG}}$: always go to state accepting the maximal language.

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

*On infinite words, $\text{DBP} \subsetneq \text{GFG}$.***

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

Theorem (Easy inclusion checking)
If $A$ is nondet and $B$ is GFG, we can decide whether $L(A) \subseteq L(B)$ in \textbf{PTime}.

\textbf{Proof:} Simulation game.
Algorithmic properties of GFG automata

**Theorem (Easy inclusion checking)**

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

**Proof:** Simulation game.

**Theorem (Easy Union, Intersection)**

*If $A$ and $B$ are GFG, then their union and intersection using cartesian product are GFG.*
Algorithmic properties of GFG automata

Theorem (Easy inclusion checking)
If $A$ is nondet and $B$ is GFG, we can decide whether $L(A) \subseteq L(B)$ in $\text{PTime}$.

Proof: Simulation game.

Theorem (Easy Union, Intersection)
If $A$ and $B$ are GFG, then their union and intersection using cartesian product are GFG.

Theorem (Hard complementation)
If $A$ is GFG for $L$ and $B$ is GFG for $\overline{L}$, then we can build in $\text{PTime}$ a deterministic automaton for $L$ based on $A \times B$. 
Algorithmic properties of GFG automata

**Theorem (Easy inclusion checking)**

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

**Proof:** Simulation game.

**Theorem (Easy Union, Intersection)**

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

**Theorem (Hard complementation)**

If $A$ is GFG for $L$ and $B$ is GFG for $\overline{L}$, then we can build in $\text{PTime}$ a deterministic automaton for $L$ based on $A \times B$.

**Proof:** Pos. Strategy in Letter game of $U = A \times B$ accepting all words.
How much memory is needed in the GFG strategy?

**Lemma**

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

**Proof**: Solve the letter game the classical way.
How much memory is needed in the GFG strategy?

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

Proof: Solve the letter game the classical way.

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

Important Remark: $GFG$ automata can be used in algorithms without knowing $\sigma_{GFG}$. The strategy $\sigma_{GFG}$ “hides” the determinism.
How much memory is needed in the GFG strategy?

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

**Proof**: Solve the letter game the classical way.

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

**Conclusion**:
Size of memory in $\sigma_{GFG} \approx$ size of equivalent det. automaton.
How much memory is needed in the GFG strategy?

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

**Proof**: Solve the letter game the classical way.

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

**Conclusion**:  
Size of memory in $\sigma_{GFG} \approx$ 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 $A$ is GFG and $D$ is a det. automaton for $L(A)$, then there is a strategy $\sigma_{GFG}$ with memory $D$.

**Proof**: Solve the letter game the classical way.

**Fact**

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

**Conclusion:**
Size of memory in $\sigma_{GFG} \approx$ 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 $\sigma_{GFG}$. The strategy $\sigma_{GFG} \text{ "hides" the determinism.}
State-blow-up for determinization

Finite words:
\( \text{GFG} = \text{DBP} \), Determinization in \( \text{PTime} \) [Löding].
State-blow-up for determinization

Finite words:
\( \text{GFG} = \text{DBP} \), Determinization in \( \text{PTime} \) [Löding].

\text{Büchi} (Parity \([1, 2]\)):
- Simple exponential determinization: powerset not Safra.
- Determinization: \( O(n^2) \) states and \( \text{NP} \) [K., Skrzypczak ’15].
- Conjecture \( O(n) \) states and in \( \text{PTime} \).
State-blow-up for determinization

Finite words:
\( \text{GFG} = \text{DBP} \), Determinization in \( \text{PTIME} \) [Löding].

\( \text{Büchi} \) (Parity \([1, 2]\)):
- Simple exponential determinization: powerset not Safra.
- Determinization: \( O(n^2) \) states and \( \text{NP} \) [K., Skrzypczak ’15].
- Conjecture \( O(n) \) states and in \( \text{PTIME} \).

\( \text{CoBüchi} \) (Parity \([0, 1]\)):
\( \text{GFG} \) automata can be exponentially more succinct than deterministic ones [K., Skrzypczak ’15]
Exponential succinctness

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

$$a : + 1 \mod n \quad b : 0 \leftrightarrow 1 \quad \# : \text{“cuts” 0}$$
Exponential succinctness

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

- $a : \quad +1 \mod n$
- $b : \quad 0 \leftrightarrow 1$
- $\# : \text{"cuts" } 0$

Graph($w$):

<table>
<thead>
<tr>
<th>$w$</th>
<th>$a$</th>
<th>$b$</th>
<th>$#$</th>
<th>$a$</th>
<th>$#$</th>
<th>$a$</th>
<th>$b$</th>
<th>$#$</th>
<th>$\ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

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

\[ a : +1 \mod n \quad b : 0 \leftrightarrow 1 \quad \# : \text{“cuts” 0} \]

Graph$(w)$:

\[
\begin{align*}
0 & \quad a & b & \# & a & \# & a & b & \# & \ldots \\
1 & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \ldots \\
2 & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \ldots \\
3 & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \quad & \ldots \\
\end{align*}
\]

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

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

Graph($w$): 

```
0
1
2
3
```

$w$: a b ♯ a ♯ a b ♯ ...
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]

▶ $\sigma_{\text{Eve}}$ and $\sigma_{\text{Adam}}$

▶ 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]
  ▶ $\sigma_{\text{Eve}}$ and $\sigma_{\text{Adam}}$
  ▶ exponential succinctness versus GFG and versus Det
  ▶ notion of half-GFG (open problems !)

($\omega$-)Pushdown automata [Guha, Jecker, Lehtinen, Zimmermann]
  ▶ Strictly between DPDA and PDA
  ▶ Exponential ($<\text{DPDA}$) and doubly exponential ($>\text{PDA}$) 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]

- $\sigma_{\text{Eve}}$ and $\sigma_{\text{Adam}}$
- 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]
- $\sigma_{\text{Eve}}$ and $\sigma_{\text{Adam}}$
- exponential succinctness versus GFG and versus Det
- notion of half-GFG (open problems !)

(\(\omega\)-)**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]

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

(\( \omega \)-)**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]

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

(\(\omega\)-)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 [losti, K. '19]
- Minimization of GFG CoBüchi automata in PTime [Abu Radi, Kupferman '20]

Det CoBüchi $\mapsto$ 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 $\rightarrow$ Minimal GFG coBüchi

Less ambitious goal: recognizing GFG automata.
Recognizing GFG automata

GFGness problem: input $A$ a ND automaton, is it GFG?
Recognizing GFG automata

**GFGness problem**: input $A$ a ND automaton, is it GFG?

**Theorem ([Henzinger, Piterman ’06])**

*We can solve the Letter game in $\text{ExpTime}$.***
Recognizing GFG automata

**GFGness problem:** input $A$ a ND automaton, is it GFG?

**Theorem ([Henzinger, Piterman ’06])**

*We can solve the Letter game in* \( \text{ExpTime} \).

**Proof:**

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

Recognizing GFG automata

**GFGness problem**: input $A$ a ND automaton, is it GFG?

**Theorem ([Henzinger, Piterman ’06])**

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

**Proof:**

- Compute a deterministic parity automaton for $L(A)$,
- 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 $\text{ExpTime}$?
Abstracting the Letter game: finite words

Theorem (Löding)

The GFGness problem is in PTime for finite words automata.
Abstracting the Letter game: finite words

**Theorem (Löding)**

The GFGness problem is in \( \text{PTime} \) for finite words automata.

The game \( G_1 \):
Adam plays letters:

Eve: moves one token \( \bigcirc \),      Adam: moves one token \( \blacksquare \)

The game \( G_1 \) is a safety game, solvable in polynomial time.

\[ a, b, c \]
\[ a \]
\[ b \]
\[ c \]
**Theorem (Löding)**

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

The game $G_1$:
- **Adam** plays letters: $a$
- **Eve** moves one token $\bullet$, **Adam** moves one token $\blacksquare$

Eve wins if at all times: $\text{accepting} \Rightarrow \text{accepting}$.

**Theorem**: Eve wins $G_1 \iff A$ is GFG.

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

Pos. Strategy $\mapsto$ Det. Automaton.
Abstracting the Letter game: finite words

Theorem (Löding)

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

The game $G_1$:
Adam plays letters: $a$
Eve: moves one token $\blacklozenge$, Adam: moves one token $\blacksquare$

Eve wins if at all times: $\text{accepting} \Rightarrow \text{accepting}$.

Theorem: Eve wins $G_1 \iff A$ is GFG.

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

Pos. Strategy $\mapsto$ Det. Automaton.
Abstracting the Letter game: finite words

Theorem (Löding)
The GFGness problem is in $\text{PTime}$ for finite words automata.

The game $G_1$:
Adam plays letters: $a$
Eve: moves one token $\bullet$, Adam: moves one token $\blacksquare$

$G_1$ is a safety game, solvable in polynomial time.
Abstracting the Letter game: finite words

Theorem (Löding)

The GFGness problem is in \textbf{PTime} for finite words automata.

The game $G_1$:

Adam plays letters: $a$ $a$

Eve: moves one token $\bigcirc$, Adam: moves one token $\blacksquare$

\begin{center}
\begin{tikzpicture}[node distance={15mm}, thick, main/.style = {draw, circle}, initial text={}]  

  \node[main, initial, initial where=left] (1) {$a, b, c$};  
  \node[main] (2) [right of=1] {$a$};  
  \node[main] (3) [right of=2] {$b$};  
  \node[main, accepting] (4) [right of=3] {$a, b, c$};  

  \path[->]  
  (1) edge [loop above] node {$a$} (1)  
  (1) edge node {$a$} (2)  
  (2) edge node {$b$} (3)  
  (3) edge [loop above] node {$c$} (3)  
  \end{tikzpicture}
\end{center}

\begin{center}
Eve wins if at all times: $\text{accepting} \Rightarrow \text{accepting}$.
\end{center}

Theorem: Eve wins $G_1$ $\iff$ Adam is GFG.

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

Pos. Strategy $\mapsto$ Det. Automaton.
Abstracting the Letter game: finite words

**Theorem (Löding)**

*The GFGness problem is in \( \text{PTIME} \) for finite words automata.*

The game \( G_1 \):

**Adam** plays letters: \( a \quad a \)

**Eve**: moves one token \( \bullet \), **Adam**: moves one token \( \blacksquare \)

\[ a, b, c \]

\[ a, b, c \]
Abstracting the Letter game: finite words

Theorem (Löding)

The GFGness problem is in \( \text{PTime} \) for finite words automata.

The game \( G_1 \): Adam plays letters: \( a \ a \)

Eve: moves one token \( \circ \), Adam: moves one token \( \square \)

Eve wins if at all times: accepting \( \Rightarrow \) accepting.

Theorem: Eve wins \( G_1 \) \( \iff \) \( A \) is GFG.

\( G_1 \) is a safety game, solvable in polynomial time.

Pos. Strategy \( \mapsto \) Det. Automaton.
Abstracting the Letter game: finite words

**Theorem (Löding)**

The GFGness problem is in \( \text{PTIME} \) for finite words automata.

The game \( G_1 \):
Adam plays letters: \( a \ a \ b \)

Eve: moves one token \( \bigcirc \), Adam: moves one token \( \square \)

\[\begin{array}{c}
a, b, c \\
a, b, c \\
a, b, c \\
\end{array}\]

\[\begin{array}{c}
a \\
a \\
b \\
\end{array}\]

\[\begin{array}{c}
b, c \\
b, c \\
c \\
\end{array}\]
Abstracting the Letter game: finite words

**Theorem (Löding)**

*The GFGness problem is in \( \text{PTime} \) for finite words automata.*

The game \( G_1 \):

Adam plays letters: \( a \ a \ b \)

Eve: moves one token \( \bigcirc \), Adam: moves one token \( \blacksquare \)

Eve wins if at all times: \( \text{accepting} \implies \text{accepting} \).

**Theorem:** Eve wins \( G_1 \) \iff A is GFG.

\( G_1 \) is a safety game, solvable in polynomial time.

**Pos. Strategy \( \mapsto \) Det. Automaton.**
Abstracting the Letter game: finite words

**Theorem (Löding)**

*The GFGness problem is in \( \text{PTime} \) for finite words automata.*

The game \( G_1 \):

*Adam plays letters: \( a \ a \ b \)*

*Eve: moves one token \( \bullet \), Adam: moves one token \( \blacksquare \)*

\( G_1 \) is a safety game, solvable in polynomial time.

Eve wins if at all times:

\[ \text{accepting} \Rightarrow \text{accepting}. \]

Theorem: Eve wins \( G_1 \) \( \Leftrightarrow \) Adam is \( \text{GFG} \).
Abstracting the Letter game: finite words

Theorem (Löding)

*The GFGness problem is in \( \text{PTime} \) for finite words automata.*

The game \( G_1 \):

Adam plays letters: \( a \ a \ b \ c \)

Eve: moves one token  

Adam: moves one token

\[ G_1 \text{ is a safety game, solvable in polynomial time.} \]

\[ \text{Pos. Strategy} \mapsto \text{Det. Automaton.} \]
Abstracting the Letter game: finite words

Theorem (Löding)

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

The game $G_1$:
- Adam plays letters: $a$, $b$, $c$
- Eve: moves one token $\bigcirc$, Adam: moves one token $\blacksquare$

Eve wins if at all times:

$\text{accepting} \Rightarrow \text{accepting}$. 

Theorem: Eve wins $G_1$ $\iff$ $A$ is GFG.

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

Positive Strategy $\mapsto$ Deterministic Automaton.
Abstracting the Letter game: finite words

**Theorem (Löding)**

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

The game $G_1$:
- **Adam** plays letters: $a$ $a$ $b$ $c$
- **Eve**: moves one token $\bigcirc$, **Adam**: moves one token $\blacksquare$

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

$\text{Pos. Strategy} \mapsto \text{Det. Automaton}$
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 $\bigcirc$, Adam: moves one token $\blacksquare$*

Eve wins if at all times: $\blacksquare$ accepting $\Rightarrow \bigcirc$ accepting.
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 $\bullet$, Adam: moves one token $\blacksquare$

Eve wins if at all times: $\blacksquare$ accepting $\Rightarrow$ $\bullet$ accepting.

Theorem: Eve wins $G_1 \iff \mathcal{A}$ is GFG.
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 $\bullet$, Adam moves one token $\blacksquare$

Eve wins if at all times: $\blacksquare$ accepting $\Rightarrow$ $\bullet$ accepting.

**Theorem:** Eve wins $G_1 \iff A$ 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 $\bigcirc$, **Adam**: moves one token $\blacksquare$
- Eve wins if at all times: $\blacksquare$ accepting $\Rightarrow \bigcirc$ accepting.

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

$G_1$ is a safety game, solvable in polynomial time.
On infinite words

Fact

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

not GFG: $\left( (a + b)^* a^{\omega} \right)$

\[ a, b \quad a \]

\[ \xrightarrow{a} 1 \quad \xrightarrow{a} 2 \]
On infinite words

Fact

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

\[
\begin{align*}
\text{not GFG:} &\quad \xymatrix{1 \ar[r]^a & 2 \ar@/_1pc/[l]^{-b} \ar@/_1pc/[l]^{-a}} \quad (a + b)^* a^\omega \\
\end{align*}
\]

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.

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 $\bigcirc$, Adam: moves two tokens $\text{1}, \text{2}$

$a, b, c$

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a$

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

Eve wins if in the long run: $\text{accepts} \Rightarrow \text{accepts}$.

The $G_2$ Conjecture

For all automata $A$, Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a$

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

$a, b, c$

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a$

Eve: moves one token \(),\quad Adam: moves two tokens \(1, 2\)

$\begin{align*}
a, b, c & \quad \quad \quad \quad a
\quad b, c & \quad \quad \quad \quad b
\quad c & \quad \quad \quad \quad c
\end{align*}$
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a \ a$

Eve: moves one token $\bullet$, Adam: moves two tokens 1, 2

$a, b, c$, $a$, $b$, $c$, $a, b, c$
Abstracting the Letter game: infinite words

The game \( G_2 \):

Adam plays letters: \( a \ a \)

Eve: moves one token \( \bullet \), Adam: moves two tokens \( \textcolor{red}{1}, \textcolor{red}{2} \)

\( a, b, c \)

\( a \)

\( b, c \)

\( b \)

\( c \)

\( a, b, c \)

Eve wins if in the long run:

\( \textcolor{red}{1}, \textcolor{red}{2} \) accepts

The \( G_2 \) Conjecture

For all automata \( A \) : Eve wins \( G_2 \) \iff \( A \) is GFG.

Solving \( G_2 \) is polynomial \implies \text{Efficient algorithm for GFGness.}
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a \ a$

Eve: moves one token $\bullet$, Adam: moves two tokens $1, 2$

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

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is $GFG$.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for $GFG$ness.
Abstracting the Letter game: infinite words

The game $G_2$:

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

Eve: moves one token $\bigcirc$, Adam: moves two tokens $\blacksquare$, $\blacksquare$

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

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

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

Eve wins if in the long run: $\Rightarrow$ accepts.

The $G_2$ Conjecture
For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

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

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

$G_2$ Conjecture

For all automata $A$: Eve wins $G_2$ $\iff A$ is GFG.

Solving $G_2$ is polynomial $\implies$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

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

Eve: moves one token $\bigcirc$,  Adam: moves two tokens $\begin{array}{c} 1 \\ 2 \end{array}$

Eve wins if in the long run:

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a$ $a$ $b$ $c$
Eve: moves one token $\circ$, Adam: moves two tokens $\mathbf{1}$, $\mathbf{2}$

Eve wins if in the long run:

\[ \Rightarrow \text{accepts}. \]

The $G_2$ Conjecture
For all automata $A$: Eve wins $G_2 \iff A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

Adam plays letters: $a \ b \ c$

Eve: moves one token $\blacklozenge$, Adam: moves two tokens $\blacksquare, \blacksquare$

Adam wins if in the long run:

1 or 2 accepts $\Rightarrow$ accepts.

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2$ $\iff$ $A$ is GFG.

Solving $G_2$ is polynomial $\Rightarrow$ Efficient algorithm for GFGness.
Abstracting the Letter game: infinite words

The game $G_2$:

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

Eve: moves one token $\bullet$, Adam: moves two tokens $\square, \blacksquare$

Eve wins if in the long run: $\square$ or $\blacksquare$ accepts $\Rightarrow \bullet$ accepts.
Abstracting the Letter game: infinite words

The game $G_2$:

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

Eve: moves one token $\bullet$, Adam: moves two tokens $\square_1$, $\square_2$

Eve wins if in the long run: $\square_1$ or $\square_2$ accepts $\Rightarrow \bullet$ accepts.

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2 \iff A$ is GFG.
Abstracting the Letter game: infinite words

The game $G_2$:

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

Eve: moves one token $\bigcirc$, Adam: moves two tokens $1$, $2$

$A, B, C$

Eve wins if in the long run: $1$ or $2$ accepts $\Rightarrow$ $\bigcirc$ accepts.

The $G_2$ Conjecture

For all automata $A$: Eve wins $G_2 \iff A$ 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 □. Some $i$ accepts $\Rightarrow$ ● must accept.
Why $G_2$ is powerful: The game $G_k$

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

**Lemma**

$Eve$ wins $G_2$ $\iff$ $Eve$ wins $G_k$ for all $k \geq 2$. 
Why $G_2$ is powerful: The game $G_k$

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

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

Proof sketch: $G_2 \Rightarrow G_3$

- play a virtual token $\circ$ against $1$ and $2$.
- play $G_2$ strategy against $\circ$ and $3$.

Proof sketch diagram:
Explorable automata

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

Definition
\(A\) is \(k\text{-GFG}\) if Eve wins the \(k\)-Letter game.
\(A\) is \textbf{Explorable} if it is \(k\text{-GFG}\) 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**

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

**Theorem (Unpublished)**

If $A$ is explorable: Eve wins $G_2 \iff A$ 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]

→ GFGness is in $P\text{TIME}$ for these automata.

For now, even with 3 parity ranks, the GFGNess problem is only known to be in $P\text{TIME}$.
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]

→ GFGness is in $P\text{Time}$ for these automata.

For now, even with 3 parity ranks, the GFGNess problem is only known to be in $\text{ExpTime}$. 
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]

→ GFGness is in $\text{PTime}$ for these automata.

For now, even with 3 parity ranks, the GFGNess problem is only known to be in $\text{ExpTime}$.

**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 $G_2 \Rightarrow GFG$ on Büchi

Assume for contradiction:

- Eve wins $G_2$, so Eve wins $G_k$ with strategy $\sigma_k$, for a big $k$.
- Adam wins the Letter game with finite-memory strategy $\tau_{GFG}$.

Idea for a strategy against $\tau_{GFG}$ in the Letter game:

- Move $k$ virtual tokens uniformly
- Play $\sigma_k$ against these $k$ tokens

Trick:
- Word from $\tau_{GFG}$ ⇒ one Büchi for some $M$ steps.
- $\Rightarrow$ wins agains $\tau_{GFG}$, contradiction.
Main proof sketch for $G_2 \Rightarrow GFG$ on Büchi

Assume for contradiction:

- Eve wins $G_2$, so Eve wins $G_k$ with strategy $\sigma_k$, for a big $k$.
- Adam wins the Letter game with finite-memory strategy $\tau_{GFG}$.

Idea for a strategy against $\tau_{GFG}$ in the Letter game:

- move $k$ virtual tokens uniformly
- play $\sigma_k$ against these $k$ tokens

Trick: Word from $\tau_{GFG} \Rightarrow$ one Büchi for some ● every $M$ steps.

⇝ ● wins against $\tau_{GFG}$, 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.
- ...