Good-for-Games Automata versus Deterministic Automata.

Denis Kuperberg$^{1,2}$ Michał Skrzypczak$^1$

$^1$University of Warsaw
$^2$IRIT/ONERA (Toulouse)

Séminaire MoVe
12/02/2015
LIF, Luminy
Deterministic automata are a central tool in automata theory:

▶ Polynomial algorithms for inclusion, complementation.
▶ Safe composition with games, trees.
▶ Solutions of the synthesis problem (verification).
▶ Easily implemented.

Problems:

▶ exponential state blow-up
▶ technical constructions (Safra)

Can we weaken the notion of determinism while preserving some good properties?
**Good-for-Games automata**

**Idea**: Nondeterminism can be resolved without knowledge about the future.
Good-for-Games automata

**Idea**: Nondeterminism can be resolved without knowledge about the future.

Introduced independently in
- symbolic representation (Henzinger, Piterman ’06) → simplification
- qualitative models (Colcombet ’09) → replace determinism

**Applications**
- synthesis
- branching time verification
- tree languages (Boker, K, Kupferman, S ’12)
**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$

System:
**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

**Environment**: $I_1$

**System**: $O_1$
Evaluating a game

**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1 \ I_2$
System: $O_1$
Synthesis: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$
System: $O_1$ $O_2$
Evaluating a game

**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$ $I_3$
System: $O_1$ $O_2$
Evaluating a game

**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$ $I_3$
System: $O_1$ $O_2$ $O_3$
**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$ $I_3$ $\cdots$
System: $O_1$ $O_2$ $O_3$ $\cdots$

System wins iff $(I_1, O_1), (I_2, O_2), (I_3, O_3), \ldots \models \varphi$. 
Synthesis: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$ $I_3$ $\cdots$
System: $O_1$ $O_2$ $O_3$ $\cdots$

System wins iff $(I_1, O_1), (I_2, O_2), (I_3, O_3), \ldots \models \varphi$.

Classical approach: $\varphi \rightsquigarrow A_{det}$ then solve game on $A_{det}$.
Synthesis: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1$ $I_2$ $I_3$ $\cdots$
System: $O_1$ $O_2$ $O_3$ $\cdots$

System wins iff $(I_1, O_1), (I_2, O_2), (I_3, O_3), \ldots \models \varphi$.

Classical approach: $\varphi \leadsto A_{det}$ then solve game on $A_{det}$.
Wrong approach: $\varphi \leadsto A_{non-det}$: no player can guess the future.
**Synthesis**: design a system responding to environment, while satisfying a constraint $\varphi$.

Environment: $I_1 I_2 I_3 \cdots$
System: $O_1 O_2 O_3 \cdots$

System wins iff $(I_1, O_1), (I_2, O_2), (I_3, O_3), \ldots \models \varphi$.

**Classical approach**: $\varphi \rightarrow A_{det}$ then solve game on $A_{det}$.

**Wrong approach**: $\varphi \rightarrow A_{non-det}$: no player can guess the future.

**New approach**: $\varphi \rightarrow A_{GFG}$. 
A automaton on finite or infinite words.

- **Refuter** plays letters:
- **Prover:** controls transitions

![Diagram of automaton]

- States: $q_0, q_1, q_2$
- Transitions:
  - $q_0$ to $q_1$: $a, b, c$
  - $q_1$ to $q_0$: $b, c$
  - $q_1$ to $q_2$: $a$
  - $q_2$ to $q_1$: $c$
  - $q_2$ to $q_0$: $a, b, c$
Definition via a game

A automaton on finite or infinite words.
Refuter plays letters: $a$
Prover: controls transitions
Definition via a game

A automaton on finite or infinite words.

**Refuter** plays letters: $a \ a$

**Prover:** controls transitions

![Diagram of a finite automaton](image-url)

- $q_0$ with transitions: $a, b, c$
- $q_1$ with transitions: $a$, $b, c$ (green transition)
- $q_2$ with transitions: $a, b, c$
A automaton on finite or infinite words.

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

Prover: controls transitions

\[
\begin{array}{c}
q_0 \overset{a}{\rightarrow} q_1 \overset{b}{\rightarrow} q_2 \\
\downarrow b, c \quad \downarrow c \\
a, b, c \quad a, b, c
\end{array}
\]
A automaton on finite or infinite words.

**Refuter** plays letters: \( a \ a \ b \ c \)

**Prover**: controls transitions

![Diagram of an automaton with states and transitions](image-url)
A automaton on finite or infinite words.

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

Prover: controls transitions

\[
\begin{array}{c}
q_0 \\
\quad \quad \quad \quad \quad \quad a, b, c \\
\quad \quad \quad \quad \quad \quad a \\
\quad \quad \quad \quad \quad \quad b, c \\
\quad \quad \quad \quad \quad \quad b \\
\quad \quad \quad \quad \quad \quad c \\
\quad \quad \quad \quad \quad \quad a, b, c \\
q_1 \\
\quad \quad \quad \quad \quad \quad a \\
\quad \quad \quad \quad \quad \quad a \\
\quad \quad \quad \quad \quad \quad b \\
\quad \quad \quad \quad \quad \quad c \\
\quad \quad \quad \quad \quad \quad a, b, c \\
q_2 \\
\end{array}
\]
A automaton on finite or infinite words.
Refuter plays letters: $a \ a \ b \ c \ c \ldots = w$
Prover: controls transitions

Player GFG wins if: $w \in L \Rightarrow$ Run accepting.
Definition via a game

A automaton on finite or infinite words.

Refuter plays letters: \(a \ a \ b \ c \ c \ldots = w\)

Prover: controls transitions

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

A GFG means that there is a strategy \(\sigma : A^* \rightarrow Q\), for accepting words of \(L(A)\).
Definition via a game

A automaton on finite or infinite words.
Refuter plays letters: $a\ a\ b\ c\ c\ c\ldots = w$
Prover: controls transitions

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

A GFG means that there is a strategy $\sigma : A^* \rightarrow Q$, for accepting words of $L(A)$.
How close is this to determinism?
Some properties of GFG automata

Composition with games: $A \circ G$ has same winner as $G$ with condition $L(A)$.
Some properties of GFG automata

Composition with games: \( \mathcal{A} \circ G \) has same winner as \( G \) with condition \( L(\mathcal{A}) \).

**Theorem**

*If \( \mathcal{A} \) is nondeterministic and \( \mathcal{B} \) is GFG, it is in \( \mathsf{P} \) to decide whether \( L(\mathcal{A}) \subseteq L(\mathcal{B}) \).*
Some properties of GFG automata

Composition with games: $\mathcal{A} \circ G$ has same winner as $G$ with condition $L(\mathcal{A})$.

**Theorem**

If $\mathcal{A}$ is nondeterministic and $\mathcal{B}$ is GFG, it is in $\mathbf{P}$ to decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

**Theorem (Boker, K, Kupferman, Skrzypczak, ICALP '12)**

If $\mathcal{A}$ and $\mathcal{B}$ are GFG for $L$ and $\bar{L}$, there is a deterministic automaton for $L$ of size $|\mathcal{A}| \cdot |\mathcal{B}|$. 
Some properties of GFG automata

Composition with games: $\mathcal{A} \circ G$ has same winner as $G$ with condition $L(\mathcal{A})$.

**Theorem**

If $\mathcal{A}$ is nondeterministic and $\mathcal{B}$ is GFG, it is in $\mathsf{P}$ to decide whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$.

**Theorem (Boker, K, Kupferman, Skrzypczak, ICALP ’12)**

If $\mathcal{A}$ and $\mathcal{B}$ are GFG for $L$ and $\overline{L}$, there is a deterministic automaton for $L$ of size $|\mathcal{A}| \cdot |\mathcal{B}|$.

**Theorem (Boker, K, Kupferman, Skrzypczak, ICALP ’12)**

Let $\mathcal{A}$ be an automaton for $L \subseteq A^\omega$. Then the tree version of $\mathcal{A}$ recognizes \{ $t : all \ branches \ of \ t \ are \ in \ L$ \} if and only if $\mathcal{A}$ is GFG.
An automaton that is not GFG

This automaton for \( L = (a + b)^* a^\omega \) is not GFG:

![Automaton Diagram]

Opponent strategy: play \( a \) until Eve goes in \( q \), then play \( ba^\omega \).
An automaton that is not GFG

This automaton for $L = (a + b)^* a^\omega$ is not GFG:

Opponent strategy: play $a$ until Eve goes in $q$, then play $ba^\omega$.

Fact

GFG automata with condition $C$ have same expressivity as deterministic automata with condition $C$.

Therefore, GFG could improve succinctness but not expressivity.
An automaton that is not GFG

This automaton for \( L = (a + b)^* a^\omega \) is not GFG:

\[
\begin{align*}
\text{Opponent strategy}: \text{play } a \text{ until Eve goes in } q, \text{ then play } ba^\omega.
\end{align*}
\]

Fact

**GFG automata with condition C have same expressivity as deterministic automata with condition C.**

Therefore, **GFG** could improve succinctness but not expressivity.

But **GFG** on finite words \( \iff \) deterministic (+useless transitions).

What about infinite words?
Büchi condition: Run is accepting if infinitely many Büchi transitions are seen.

Language: \([(xa + xb)^{\omega} (xaxa + xbxb)^{\omega}]\)
Determinization of Büchi GFG

**Theorem**

Let $\mathcal{A}$ a GFG Büchi automaton. There exists a deterministic Büchi automaton $\mathcal{B}$ with $L(\mathcal{B}) = L(\mathcal{A})$ and $|\mathcal{B}| \leq |\mathcal{A}|^2$.

**Proof scheme:**

- Use brutal powerset determinisation,
- rank signatures of Walukiewicz
- iterative normalization of $\mathcal{A}$
- dependency graph over the automaton

**Conclusion:** the automaton can use itself as memory structure $\Rightarrow$ quadratic blow-up only.

Is it true for all $\omega$-regular conditions?
CoBüchi counter-example: the language $L_n$

$n$ paths: $\sigma, \pi$ permute paths, \# cuts the current 0-path.

Here for $n = 5$:

$$\alpha: \begin{array}{ccccccc}
\sigma & \pi & \sigma & \# & \pi & \sigma & \pi & \# \\
\hline
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8
\end{array}$$

The word $\alpha$ is in $L_n$ if it contains an infinite path.
Automaton for $L_n$

**GFG coBüchi automaton for $L_n$ with $n$ states:**

- letters $\sigma$ and $\pi$ permute states deterministically.
- letter $\#$:
  - state 0 $\rightarrow$ go anywhere but pay a coBüchi (must be finitely many times)
  - states 1, ..., $n$: do nothing

**Strategy $\sigma$:** try paths one after the other. Uses memory $2^n$, to ensure that all paths are visited.
Automaton for $L_n$

**GFG** coBüchi automaton for $L_n$ with $n$ states:

- letters $\sigma$ and $\pi$ permute states deterministically.
- letter $\#$:
  - state 0 $\rightarrow$ go anywhere but pay a coBüchi (must be finitely many times)
  - states 1, $\ldots$, $n$: do nothing

**Strategy $\sigma$:** try paths one after the other. Uses memory $2^n$, to ensure that all paths are visited.

**Theorem**

*Any deterministic automaton for $L_n$ has $\Omega(2^n)$ states.*

CoBüchi (and parity) **GFG** automata can provide both succinctness and sound behaviour with respect to games.

**Question:** Can we effectively use them?
Recognizing GFG automata

**Question:** Given an automaton $\mathcal{A}$, is it GFG?

**Theorem**

*The complexity of deciding GFG-ness is in*

- $\text{NP}$ for Büchi automata
- $\text{P}$ for coBüchi automata (involved proof)
- at least as hard as solving parity games ($\text{NP} \cap \text{coNP}$) for parity automata.

**Open Problems**

- Is it in $\text{P}$ for any **fixed** acceptance condition?
- Is it equivalent to parity games in the general case?
Summary and conclusion

Results

- **GFG** automata capture good properties of deterministic automata.
- Inclusion is in $\mathbf{P}$, but Complementation $\sim$ Determinisation.
- Conditions Büchi and lower: **GFG** $\approx$ Deterministic.
- Conditions coBüchi and higher: exponential succinctness.
- Recognizing **GFG** coBüchi is in $\mathbf{P}$.

Open Problems

- Can we build small **GFG** automata in a systematic way?
- Complexity of deciding **GFG**-ness for parity automata?
- Exact cost of Büchi **GFG** determinisation?