Good-for-Games Automata versus Deterministic Automata.

Denis Kuperberg       Michał Skrzypczak

IRIT/ONERA (Toulouse) and University of Warsaw

Séminaire du LACL
05/01/2015
Crteil
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?
**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)
Evaluating a game

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

Environment: $I_1$

System:
Evaluating a game

**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: $O_1$
Evaluating a game

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

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

Environment: $l_1$  $l_2$  $l_3$
System: $O_1$  $O_2$
**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**: $\varphi \leadsto A_{det}$ 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:** $\varphi \rightarrow A_{det}$ then solve game on $A_{det}$.

**Wrong approach:** $\varphi \rightarrow A_{non-det}$ : 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: $\varphi \rightarrow \mathcal{A}_{\text{det}}$ then solve game on $\mathcal{A}_{\text{det}}$.
Wrong approach: $\varphi \rightarrow \mathcal{A}_{\text{non-det}}$: no player can guess the future.
New approach: $\varphi \rightarrow \mathcal{A}_{\text{GFG}}$. 

Definition via a game

A automaton on finite or infinite words.

Refuter plays letters:

Prover: controls transitions
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
A automaton on finite or infinite words.

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

Prover: controls transitions
Definition via a game

A automaton on finite or infinite words.

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

**Prover:** controls transitions
Definition via a game

A automaton on finite or infinite words.

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

Prover: controls transitions
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 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 \( \sigma : A^* \to Q \), for accepting words of \( L(A) \).
Definition via a game

A automaton on finite or infinite words.

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

Prover: controls transitions

\[
\begin{array}{c}
\xrightarrow{a, b, c} \quad a \\
\xrightarrow{a} \quad b, c \\
\xrightarrow{b} \quad c \\
\xrightarrow{a, b, c} \quad q_0 \quad q_1 \quad q_2
\end{array}
\]

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

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

How close is this to determinism?
Some properties of GFG automata

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

**Theorem (Boker, K, Kupferman, S ’12)**

Let $\mathcal{A}$ be an automaton for $L \subseteq A^\omega$. Then the tree version of $\mathcal{A}$ recognizes $\{t : \text{all branches of } t \text{ are in } L\}$ if and only if $\mathcal{A}$ is GFG.

**Fact**

Let $\mathcal{A}$ be GFG on finite words. Then $\mathcal{A}$ contains an equivalent deterministic automaton.

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

Language: \([(xa + xb)^* (xaxa + xbxb)]^\omega\)
Theorem

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

Proof scheme:

- Use brutal powerset determinisation,
- rank signatures of Walukiewicz
- iterative normalization of $A$
- 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?
The language $L_n$

$n$ paths: $\sigma, \pi$ permute paths, $\#$ cuts the current 0-path.
Here for $n = 5$:

$$\alpha: \quad \begin{array} \sigma & \pi & \sigma & \# & \pi & \sigma & \pi & \# \\ \end{array}$$

$$\text{DAG:} \quad \left\{ \begin{array} \hline 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ \hline \end{array} \right.$$  

$$\text{time:} \quad \begin{array} \hline 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \hline \end{array}$$

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 $1, \ldots, n$: 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.
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

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

**Open Problems**

- Is it in $P$ 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 $\text{P}$, but Complementation $\sim$ Determinisation.
- Conditions Büchi and lower: **GFG** $\approx$ Deterministic.
- Conditions coBüchi and higher: exponential succinctness.
- Recognizing **GFG** coBüchi is in $\text{P}$.

Open Problems

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