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

Introduction

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)
  $\rightarrow$ simplification
- qualitative models (Colcombet '09) $\rightarrow$ 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 φ.

Environment: I_1
System:

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

Environment: I_1
System: O_1

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

Environment: I_1 I_2
System: O_1

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

Environment: I_1 I_2
System: O_1 O_2

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

Environment: $I_1 \quad I_2 \quad I_3$
System: $O_1 \quad O_2$

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

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

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

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

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

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

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

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

Classical approach: $\varphi \rightsquigarrow \mathcal{A}_{det}$ then solve game on $\mathcal{A}_{det}$.

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

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

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

Classical approach: $\varphi \rightsquigarrow \mathcal{A}_{det}$ then solve game on $\mathcal{A}_{det}$.
Wrong approach: $\varphi \rightsquigarrow \mathcal{A}_{non-det}$: no player can guess the future.

Evaluating a game

Synthesis : design a system responding to environment, while satisfying a constraint φ.

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

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

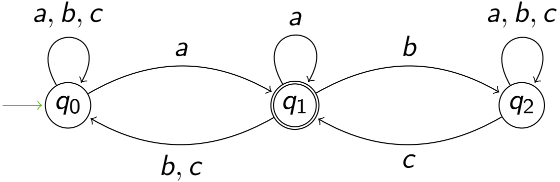
Classical approach: $\varphi \rightsquigarrow \mathcal{A}_{det}$ then solve game on $\mathcal{A}_{det}$.
Wrong approach: $\varphi \rightsquigarrow \mathcal{A}_{non-det}$: no player can guess the future.
New approach: $\varphi \rightsquigarrow \mathcal{A}_{GFG}$.

Definition via a game

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

Refuter plays letters:

Prover: controls transitions

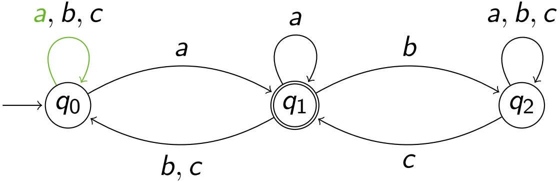


Definition via a game

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

Refuter plays letters: a

Prover: controls transitions

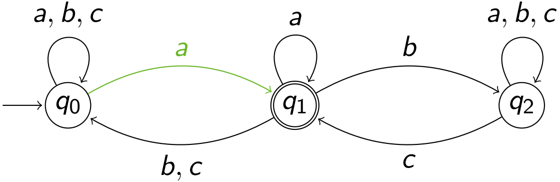


Definition via a game

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

Refuter plays letters: a a

Prover: controls transitions

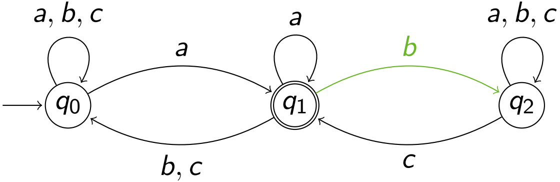


Definition via a game

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

Refuter plays letters: a a b

Prover: controls transitions

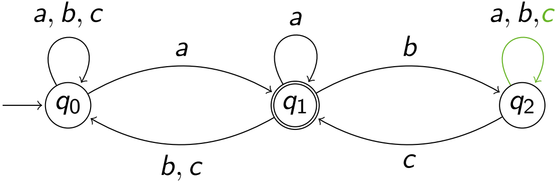


Definition via a game

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

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

Prover: controls transitions

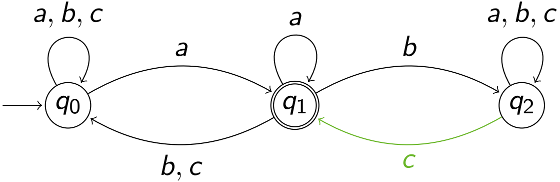


Definition via a game

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

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

Prover: controls transitions

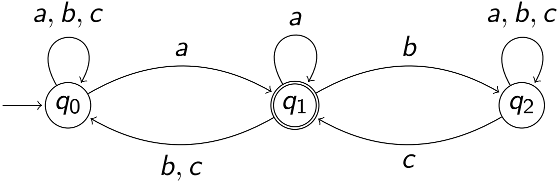


Definition via a game

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

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

Prover: controls transitions



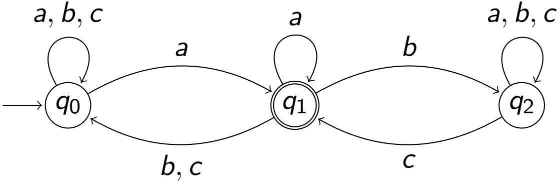
Player GFG wins if: $w \in L \Rightarrow$ Run accepting.

Definition via a game

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

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

Prover: controls transitions



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

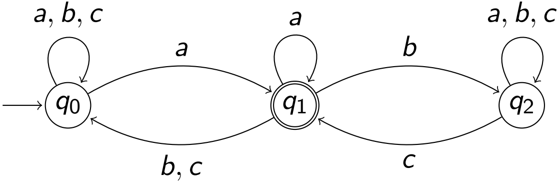
$\mathcal{A}$ **GFG** means that there is a strategy $\sigma : A^* \to Q$, for accepting words of $L(\mathcal{A})$.

Definition via a game

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

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

Prover: controls transitions



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

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

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 $\{t$: all branches of t are in $L\}$ if and only if $\mathcal{A}$ is **GFG**.

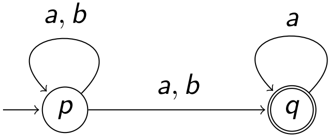
Fact

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^ω.

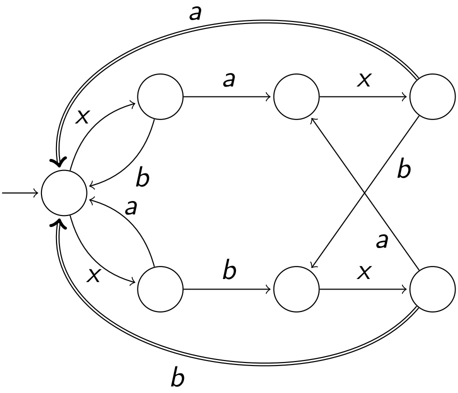
Fact

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

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

A GFG Büchi example

Büchi condition: Run is accepting if infinitely many Büchi transitions are seen.



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

Determinization of Büchi GFG

Theorem

Let $\mathcal{A}$ a **GFG** *Büchi automaton. There exists a deterministic 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.

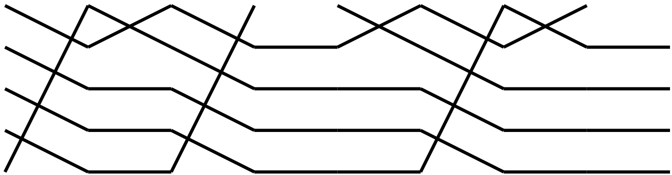
Is it true for all ω-regular conditions?

The language L_n

n paths: σ, π permute paths, $\sharp$ cuts the current 0-path.
Here for $n = 5$:

α: σ π σ $\sharp$ π σ π $\sharp$

DAG: 0 1 2 3 4



time: 0 1 2 3 4 5 6 7 8

The word α is in L_n if it contains an infinite path.

Automaton for L_n

GFG coBüchi automaton for L_n with n states:

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

Strategy σ: 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 σ and π permute states deterministically.
- letter $\sharp$:
  - state $0 \longrightarrow$ go anywhere but pay a coBüchi (must be finitely many times)
  - states $1, \ldots, n$: do nothing

Strategy σ: 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** $\cap$ *co***NP**) *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 **P**, but Complementation $\sim$ Determinisation.
- Conditions Büchi and lower: **GFG** $\approx$ Deterministic.
- Conditions coBüchi and higher: exponential succinctness.
- Recognizing **GFG** coBüchi is in **P**.

Open Problems

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