On Determinisation of History-Deterministic Automata.

Denis Kuperberg Michał Skrzypczak

University of Warsaw

YR-ICALP 2014
Copenhagen

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?

History-deterministic automata

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

History-deterministic 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

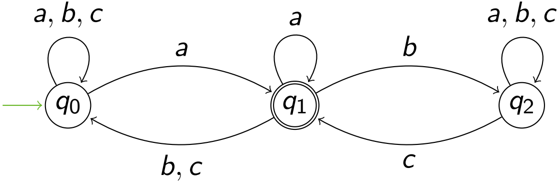
- synthesis
- branching time verification
- tree languages (Boker, K, Kupferman, S '12)

Definition via a game

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

Refuter plays letters:

Prover: controls transitions

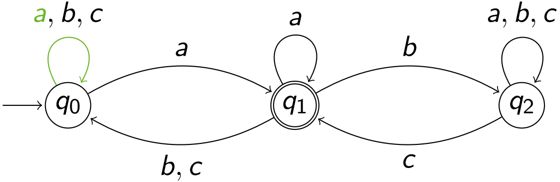


Definition via a game

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

Refuter plays letters: a

Prover: controls transitions

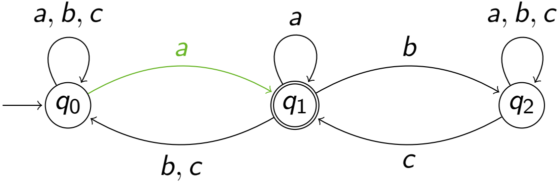


Definition via a game

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

Refuter plays letters: a a

Prover: controls transitions

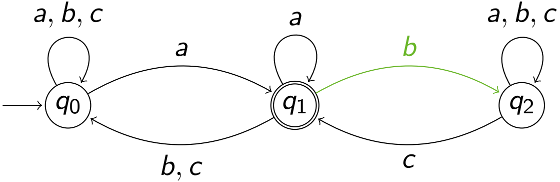


Definition via a game

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

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

Prover: controls transitions

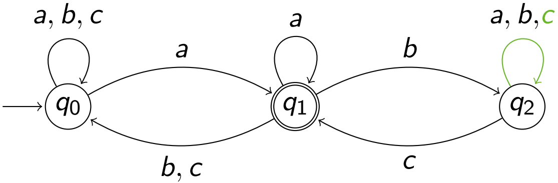


Definition via a game

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

Refuter plays letters: a a b c

Prover: controls transitions

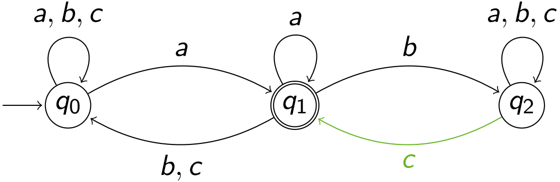


Definition via a game

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

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

Prover: controls transitions

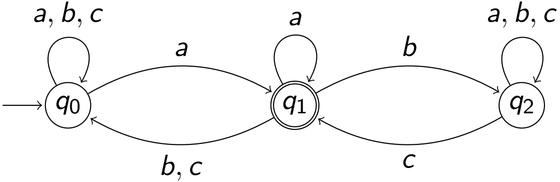


Definition via a game

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

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

Prover: controls transitions



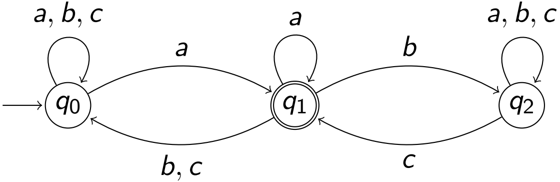
Player H-D wins if: $w \in L \Rightarrow$ Run accepting.

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 H-D wins if: $w \in L \Rightarrow$ Run accepting.

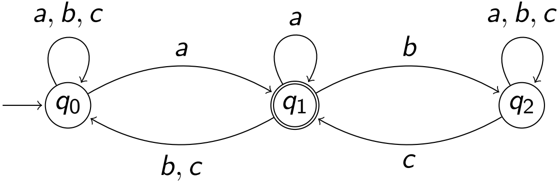
$\mathcal{A}$ **H-D** means that there is a strategy $\sigma : A^* \to Q$, for accepting words of $L(\mathcal{A})$.

Definition via a game

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

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

Prover: controls transitions



Player H-D wins if: $w \in L \Rightarrow$ Run accepting.

$\mathcal{A}$ **H-D** 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 H-D 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 **H-D**.

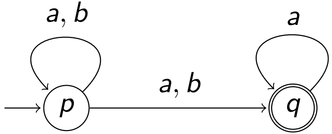
Fact

Let $\mathcal{A}$ be **H-D** *on finite words. Then $\mathcal{A}$ contains an equivalent deterministic automaton.*

Therefore, **H-D** is not a useful concept on finite words. What about infinite words ?

An automaton that is not H-D

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



Opponent strategy: play a until Eve goes in q, then play ba^ω.

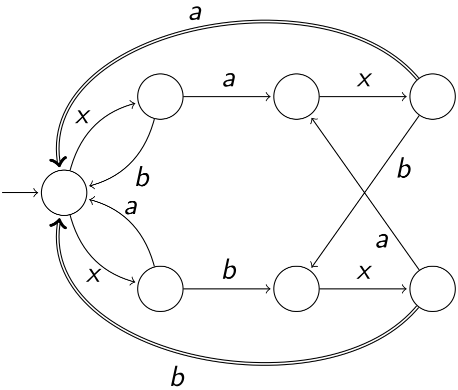
Fact

H-D *automata with condition* C *have same expressivity as deterministic automata with condition* C*.*

Therefore, **H-D** could improve succinctness but not expressivity.

A H-D Büchi example

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



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

Determinization of Büchi H-D

Theorem

Let $\mathcal{A}$ a **H-D** *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

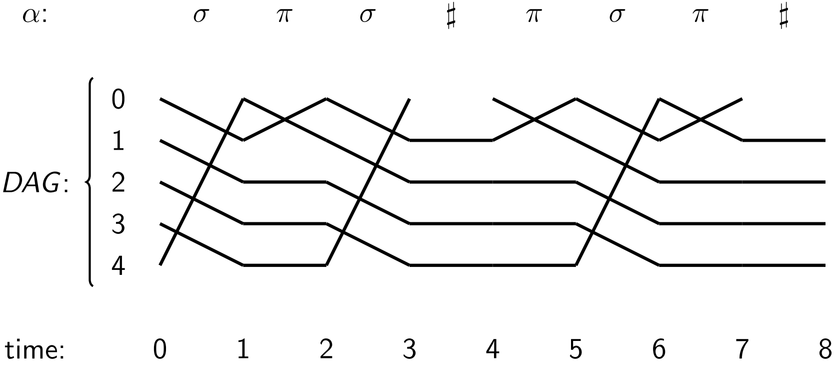
Conclusion: the automaton can use itself as memory structure $\Rightarrow$ quadratic blow-up only.

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 $n = 5$:



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

Automaton for L_n

H-D 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) **H-D** automata can provide both succinctness and sound behaviour with respect to games.

Automaton for L_n

H-D 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) **H-D** automata can provide both succinctness and sound behaviour with respect to games.

Question: Can we effectively use them?

Recognizing H-D automata

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

Theorem

*The complexity of deciding **H-D**-ness is in*

- **P** *for coBüchi automata*
- **NP** *for Büchi automata*
- *at least as hard as solving parity games* ($\mathbf{NP} \cap co\mathbf{NP}$) *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

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

Open Problems

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