On Determinisation of History-Deterministic Automata.

Denis Kuperberg    Michał Skrzypczak

University of Warsaw

YR-ICALP 2014
Copenhagen
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.
History-deterministic 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](#))
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
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
A automaton on finite or infinite words.

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

Prover: controls transitions
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 H-D wins if: $w \in L \Rightarrow$ Run accepting.
A automaton on finite or infinite words.

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

Prover: controls transitions

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

A H-D 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: $a \ a \ b \ c \ c \ \ldots = w$
Prover: controls transitions

Player H-D wins if: $w \in L \implies \text{Run accepting.}$

A H-D 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 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 $L \subseteq \mathcal{A}^\omega$. 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 \( ba^\omega \).

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

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

Let $A$ a H-D 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 $n = 5$:

\[
\begin{array}{ccccccccc}
\alpha: & \sigma & \pi & \sigma & \# & \pi & \sigma & \pi & \#
\end{array}
\]

\[
\begin{aligned}
&DAG: \begin{cases}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
0 & & & & & & & & \\
1 & & & & & & & & \\
2 & & & & & & & & \\
3 & & & & & & & & \\
4 & & & & & & & & \\
\end{cases}
\end{aligned}
\]

The word $\alpha$ 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 $\sigma$ 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) **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 $\sigma$ 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) **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 $A$, is it H-D?

**Theorem**

The complexity of deciding H-D-ness is in

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

**Open Problems**

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

Open Problems

- Can we build small $\mathbf{H-D}$ automata in a systematic way?
- Complexity of deciding $\mathbf{H-D}$-ness for parity automata?