Coinduction up-to

from concurrency to coalgebra and back

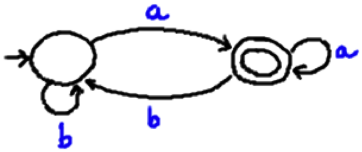
Filippo Bonchi and Alexandra Silva

ENS Lyon (FR) and Radboud University Nijmegen (NL)

June 18, 2014
OPCT 2014
Bertinoro, Italy

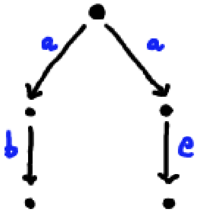
Context

- Automata are basic structures in Computer Science.
- Language equivalence: well-studied, several algorithms.
- Renewed attention (POPL'11, '13, '14).



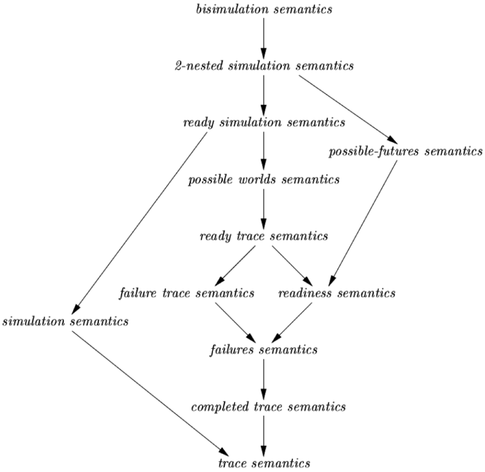
Context

- Concurrency: a spectrum of equivalences.
- Checking usually done by reducing to bisimilarity.



An alternative road

- Many efficient algorithms for equivalence of automata.
- Applications in concurrency?



From automata to concurrency

Various spectrum equivalences

=

Language equivalence of a *transformed* system

=

Automaton with outputs and structured state space (Moore automata).

Bonsangue, Bonchi, Caltais, Rutten, S. MFPS 12

From automata to concurrency

- Generalization of existing algorithms to Moore automata.
- Brzozowski's and Hopcroft/Karp algorithms for van Glabbeek's spectrum.
- Cleaveland and Hennessy's acceptance graphs for must/may testing = Moore automata.
- Brzozowski's and Hopcroft/Karp algorithms algorithm for must/may testing.

Bonchi, Caltais, Pous, Silva. APLAS 2013

From automata to concurrency

- Generalization of existing algorithms to Moore automata.
- Brzozowski's and Hopcroft/Karp algorithms for van Glabbeek's spectrum.
- Cleaveland and Hennessy's acceptance graphs for must/may testing = Moore automata.
- Brzozowski's and Hopcroft/Karp algorithms algorithm for must/may testing.

Bonchi, Caltais, Pous, Silva. APLAS 2013

The approach

Roadmap

1. Automata algorithms applied to concurrency.
2. For the rest of the talk: up-to techniques applied to automata.

Compositionality

$[\![X + Y]\!] = [\![X]\!] + [\![Y]\!]$

Coinduction

Proof principle for infinite structures

Roadmap

1. Automata algorithms applied to concurrency.
2. For the rest of the talk: up-to techniques applied to automata.

Compositionality	Coinduction
$[\![X + Y]\!] = [\![X]\!] + [\![Y]\!]$	Proof principle for infinite structures

The rest of the talk

- Deterministic Automata
  - Naive algorithm (for language equivalence)
  - Hopcroft & Karp's algorithm
- Non-Deterministic Automata
  - Powerset Construction
  - On the fly algorithm
  - **H&K-up-to-congruence algorithm**
- Discussion and Future Work

The rest of the talk

- Deterministic Automata
  - Naive algorithm (for language equivalence)
  - Hopcroft & Karp's algorithm
- Non-Deterministic Automata
  - Powerset Construction
  - On the fly algorithm
  - **H&K-up-to-congruence algorithm**
- Discussion and Future Work

Deterministic Automata

(S,o,t)

$$\begin{tikzcd} x \arrow[r, "a"] & \overline{y} \arrow[r, bend left, "a"] & z \arrow[l, bend left, "a"] \end{tikzcd}$$

set of states **S**

output function **o: S-->2** $\{0,1\}$

transition function **t: S-->S^A**

Accepted Language

$$[\![-]\!] : S \to 2^{A^*}$$

for all $x \in S$:

$$[\![x]\!](\epsilon) = o(x)$$

$$[\![x]\!](a \cdot w) = [\![t(x)(a)]\!](w)$$

Language Equivalence

$$x \sim y$$

$\underline{\texttt{Naive}(x,y)}$

(1) R is empty; $todo$ is empty;

(2) insert (x, y) in $todo$;

(3) while $todo$ is not empty

{

(3.1) extract (x', y') from $todo$;

(3.2) if $(x', y') \in R$ then skip, else {

(3.3) if $o(x') \neq o(y')$ then return $false$, else {

(3.4) for all $a \in A$, insert $(t(x')(a), t(y')(a))$ in $todo$;

(3.5) insert (x', y') in R; }}

}

(4) return $true$;

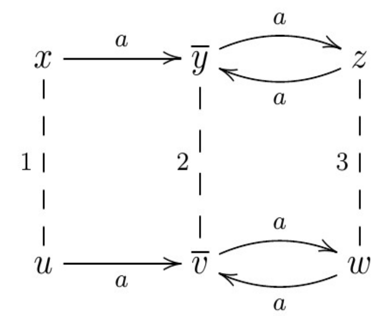
$$\begin{tikzcd} x \arrow[r, "a"] & \overline{y} \arrow[r, bend left, "a"] & z \arrow[l, bend left, "a"] \end{tikzcd}$$

$$todo = \{(z, w)\} \qquad (z, w)$$

$$R = \{(x, u), (y, v), (z, w)\}$$

$$\begin{tikzcd} u \arrow[r, "a"'] & \overline{v} \arrow[r, bend left, "a"] & w \arrow[l, bend left, "a"] \end{tikzcd}$$

Language Equivalence via Bisimulations



Given an automaton <o,t>:S-->2xS^{A},

B:Rel_S-->Rel_S is defined

for all R⊆S×S as

B(R)= {(x,y) | o(x)=o(y) & ∀a∈A, (t(x)(a),t(y)(a))∈R }

νB is language equivalence

<u>Def:</u> A *bisimulation* is a relation R such that R⊆**B**(R)

<u>Coinduction Proof Principle:</u>

$[\![x]\!] = [\![y]\!]$ iff (x,y)∈R, for some bisimulation R

Naive(x,y)

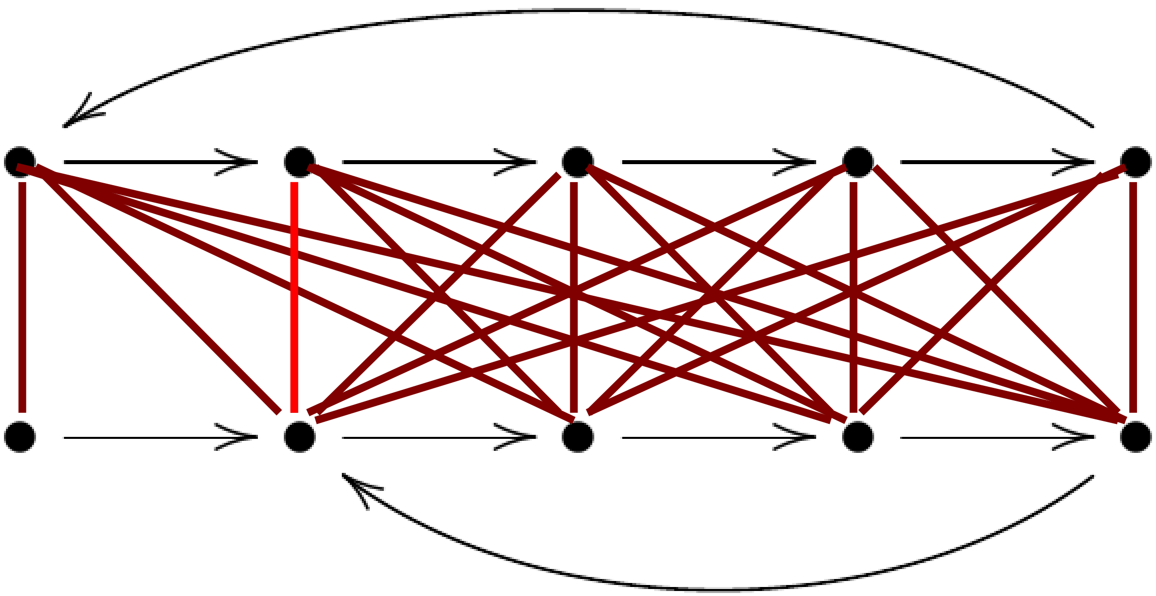
```
(1)   R is empty; todo is empty;
(2)   insert (x, y) in todo;
(3)   while todo is not empty
 {
      (3.1)   extract (x', y') from todo;
      (3.2)   if (x', y') ∈ R then skip, else {
      (3.3)   if o(x') ≠ o(y') then return false, else {
      (3.4)   for all a ∈ A, insert (t(x')(a), t(y')(a)) in todo;
      (3.5)   insert (x', y') in R; }}
 }
(4)   return true;
```

R⊆**B**(R∪todo)

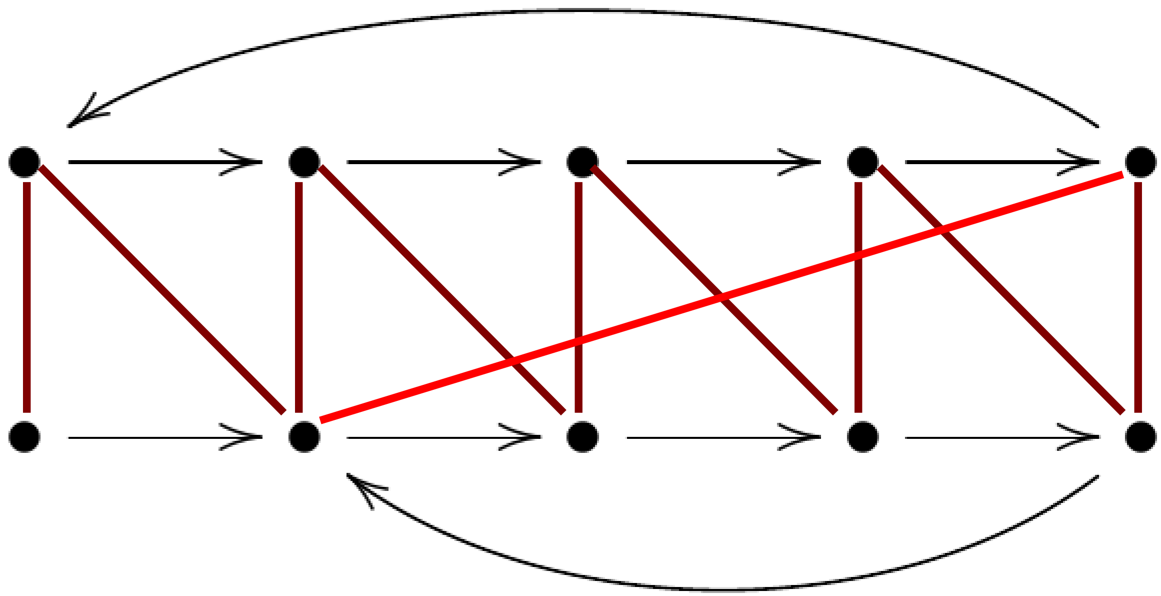
$x \sim y$ *iff* Naive(x,y)

After (3), R⊆**B**(R)

Hopcroft and Karp's Algorithm (1971)



Hopcroft and Karp's Algorithm (1971)



HK(x,y)

(1) R is empty; $todo$ is empty;

(2) insert (x, y) in $todo$;

(3) while $todo$ is not empty

$\{$

(3.1) extract (x', y') from $todo$;

(3.2) if (x', y') $\in \mathbf{E}(R)$ then skip, else $\{$

(3.3) if $o(x') \neq o(y')$ then return $false$, else $\{$

(3.4) for all $a \in A$, insert ($t(x')(a)$, $t(y')(a)$) in $todo$;

(3.5) insert (x', y') in R; $\}\}$

$\}$

(4) return $true$;

R⊆**B**(**E**(R)∪todo)

At most n iterations!

$x \sim y$ iff HK(x,y)

The complexity is n log(n)

After (3), R⊆**B**(**E**(R))

i.e, R is a <u>bisimulation up-to equivalence</u>

Mistakes in Milner's book

Weak Bisimulation up-to Equivalence

Weak Bisimulation

up-to Weak Bisimilarity

Plan of the Talk

- Deterministic Automata
  - Naive algorithm (for language equivalence)
  - Hopcroft & Karp's algorithm
- Non-Deterministic Automata
  - Powerset Construction
  - On the fly algorithm
  - **H&K-up-to-congruence algorithm**
- Discussion and Future Work

Semi-Lattices

$$(X, +, 0)$$

X a set

$+ : X \times X \to X$ Associative-Commutative-Idempotent

$0 \in X$ the identity element

Examples

$2 = \{0, 1\}$, $+ =$ "or", 0

2^{A^*}, $+ =$ "union of languages", $0 =$ "empty language"

$\mathcal{P}(S)$, $+ =$ "union of subsets", $0 =$ "empty set"

Semi-Lattices

$$(X, +, 0)$$

X a set

$+ \colon X \times X \to X$ Associative-Commutative-Idempotent

$0 \in X$ the identity element

Homomorphisms

$$f \colon (X_1, +_1, 0_1) \to (X_2, +_2, 0_2)$$

for all $x, y \in X_1$,

$$f(x +_1 y) = f(x) +_2 f(y) \text{ and}$$

$$f(0_1) = 0_2.$$

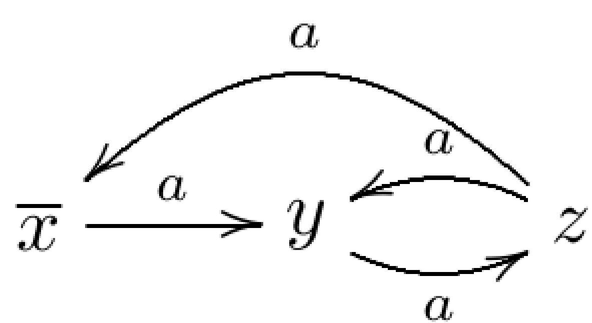
Non-Deterministic Automata

(S, o, δ)

S set of states

$o: S \dashrightarrow 2$ output function

$\delta: S \dashrightarrow P(S)^A$ transition relation



Determinization

$$(S, o, \delta) \longrightarrow (\mathcal{P}(S), o^\sharp, \delta^\sharp)$$

$o^\sharp : \mathcal{P}(S) \to 2$

$$o^\sharp(X) = \begin{cases} o(x) & \text{if } X = \{x\} \text{ with } x \in S \\ 0 & \text{if } X = 0 \\ o^\sharp(X_1) + o^\sharp(X_2) & \text{if } X = X_1 + X_2 \end{cases}$$

$\delta^\sharp : \mathcal{P}(S) \to \mathcal{P}(S)^A$

$$\delta^\sharp(X)(a) = \begin{cases} \delta(x)(a) & \text{if } X = \{x\} \text{ with } x \in S \\ 0 & \text{if } X = 0 \\ \delta^\sharp(X_1)(a) + \delta^\sharp(X_2)(a) & \text{if } X = X_1 + X_2 \end{cases}$$

$$\begin{tikzcd} \overline{x} \arrow[r, "a"] & y \arrow[r, bend right, "a"'] & z \arrow[l, bend right, "a"'] \arrow[ll, bend right=60, "a"'] \end{tikzcd}
\qquad
\begin{tikzcd} \overline{x} \arrow[r, "a"] & y \arrow[r, "a"] & z \arrow[r, "a"] & \overline{x+y} \arrow[r, "a"] & y+z \arrow[r, "a"] & \overline{x+y+z} \arrow[loop above, "a"] \end{tikzcd}$$

Accepted Language

$$(S,o,\delta) \longrightarrow (P(S),o^{\#},\delta^{\#}) \xrightarrow{[\![-]\!]} 2^{A^*}$$

A *bisimulation* is a relation $R \subseteq P(S) \times P(S)$ such that $R \subseteq \mathbf{B}(R)$ where **B:Rel_P(S)-->Rel_P(S)** is defined as

For all $R \subseteq P(S) \times P(S)$,

$$\mathbf{B}(R) = \{(X,Y) \mid o^{\#}(X) = o^{\#}(Y) \;\&\; \forall a \in A,\ (\delta^{\#}(X)(a), \delta^{\#}(Y)(a)) \in R\}$$

<u>Coinduction Proof Principle:</u>

$[\![X]\!] = [\![Y]\!]$ iff $(X,Y) \in R$, for some bisimulation R

HKNFA (X, Y)

(1) R is empty; *todo* is empty;

(2) insert (X, Y) in *todo*;

(3) while *todo* is not empty

{

(3.1) extract (X', Y') from *todo*;

(3.2) if $(X', Y') \in \mathbf{E}(R)$ then skip, else $\{$

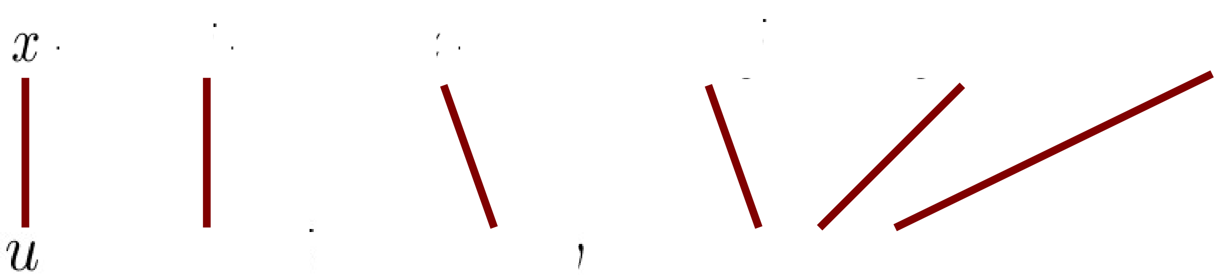
(3.3) if $o^\sharp(X') \neq o^\sharp(Y')$ then return *false*, else $\{$

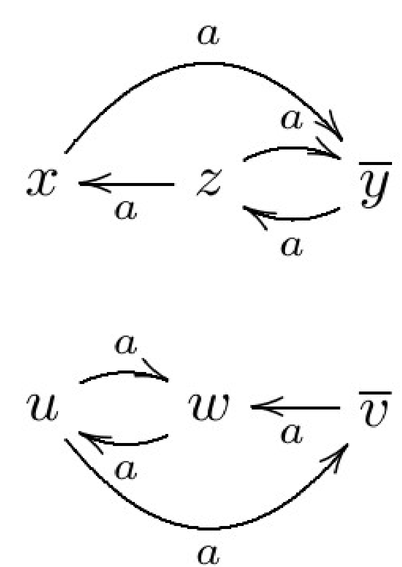
(3.4) for all $a \in A$, insert $(\delta^\sharp(X')(a), \delta^\sharp(Y')(a))$ in *todo*;

(3.5) insert (X', Y') in R; $\}\}$

}

(4) return *true*;



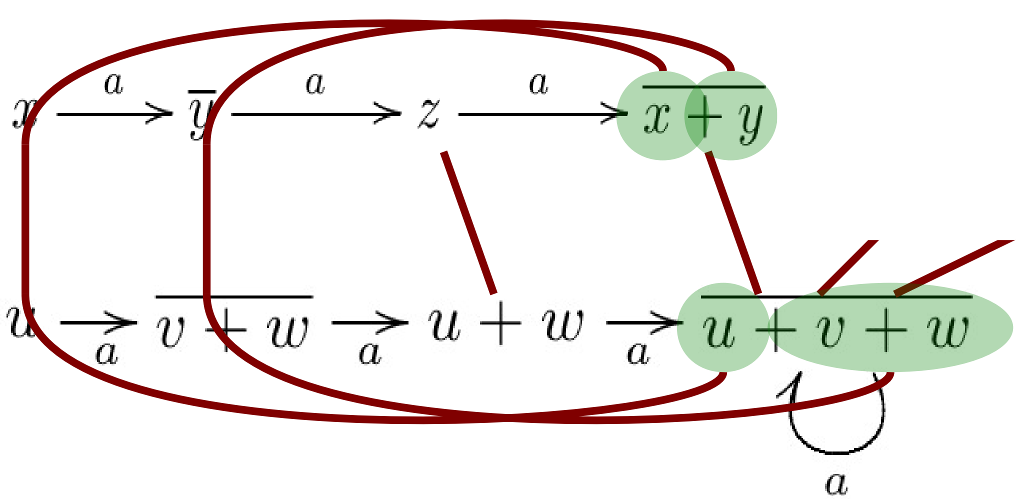


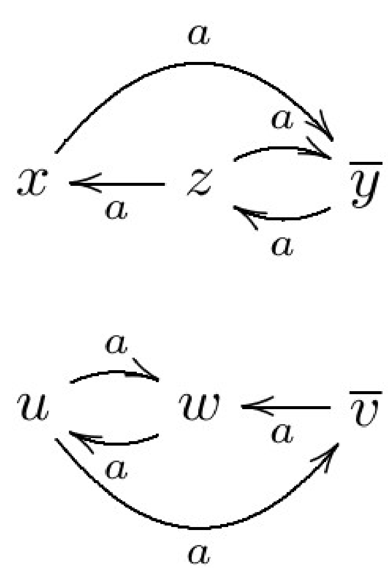
Our Idea...

$$[\![-]\!] : \mathcal{P}(S) \to 2^{A^*}$$

for all $X_1, X_2 \in \mathcal{P}(S)$,

$$[\![X_1 + X_2]\!] = [\![X_1]\!] + [\![X_2]\!] \quad \textit{and} \quad [\![0]\!] = 0$$





$\underline{\texttt{HKC}(X,Y)}$

(1) R is empty; *todo* is empty;

(2) insert $(X,\ Y)$ in *todo*;

(3) while *todo* is not empty

$\{$

(3.1) extract $(X',\ Y')$ from *todo*;

(3.2) if $(X',\ Y') \in \mathbf{C}(R)$ then skip, else $\{$

(3.3) if $o^\sharp(X') \neq o^\sharp(Y')$ then return *false*, else $\{$

(3.4) for all $a \in A$, insert $(\delta^\sharp(X')(a),\ \delta^\sharp(Y')(a))$ in *todo*;

(3.5) insert $(X',\ Y')$ in R; $\}\}$

$\}$

(4) return *true*;

R⊆**B**(**C**(R)∪todo)

$$X \sim Y \ \textit{iff} \ \texttt{HKC}(X,Y)$$

After (3), R⊆**B**(**C**(R))

namely, R is a <u>bisimulation up-to congruence</u>

Conclusions

- Implementation is available online

  (Googling HKC automata)

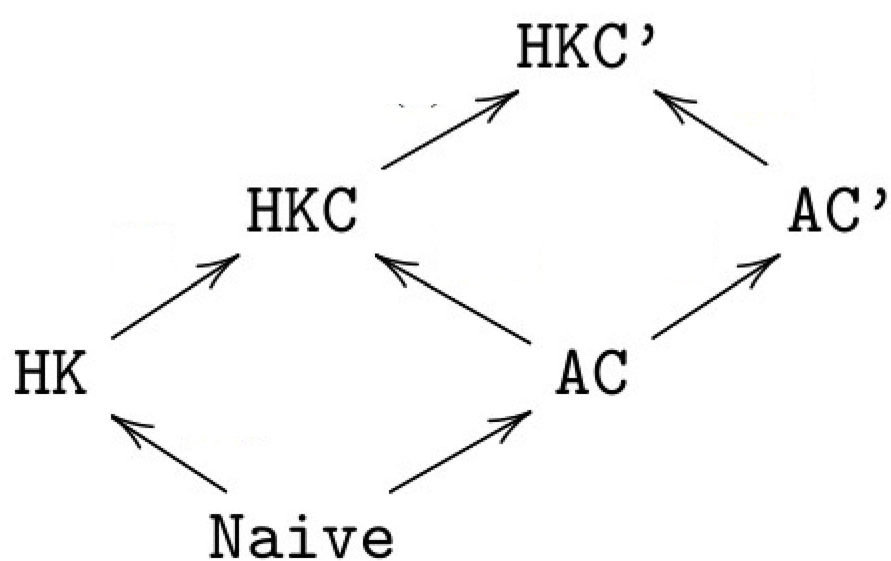
  and more and more used (already 24 citations, see e.g., www.languageinclusion.org)
- Interactive Applet & COQ proof scripts
- A follow-up will appear in LICS 2014
  - Weighted Automata, Nominal Automata, Process Calculi
  - Different sort of Coinductive Predicates like Termination, Similarity, Weak Bisimilarity

Antichain Approach

AC M. D. Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In Proc. CAV 2006.

AC' P. A. Abdulla, Y.-F. Chen, L. Holik, R. Mayr, and T. Vojnar. When simulation meets antichains. In Proc. TACAS 2010.

Following AC', we developed another algorithm called HKC'



Experimental Assessment