Coinductive algorithms for Büchi automata

Denis Kuperberg   Laureline Pinault   Damien Pous

LIP, Ens de Lyon

DLT, Warsaw
August 5th 2019
Introduction and motivations

Büchi automata (NBWs):

\[
\langle S, T \rangle \text{ with } T : A \rightarrow 3^{S^2}
\]

\[\mathcal{L}(A) : A^\omega \rightarrow 2\]

Example:

\[
\begin{array}{c}
\text{a, b} \\
\text{0} \\
\text{b} \\
\text{1} \\
\text{a}
\end{array}
\]

\[\mathcal{L} = (a + b)^* b a^\omega\]

Problem:

\[\mathcal{L}(A) = \mathcal{L}(B)?\]

PSPACE-Complete

Modelize:

- Programs and systems
- Specifications of systems (eg. LTL formulae)

Application:

Verification of systems
Past works
## Existing algorithms

<table>
<thead>
<tr>
<th>Problem</th>
<th>NFAs</th>
<th>NBWs</th>
</tr>
</thead>
</table>
| **Inclusion** | Antichain-based Algorithms  
[Wulf, Doyen, Henzinger, Raskin ’06]  
[Abdulla, Chen, Holik, et al ’11]  
[Doyen, Raskin ’10]  
Tools: Powerset construction and subsumption techniques | Antichain-based Algorithms  
[Fogarty, Vardi ’09 ’10]  
[Doyen, Raskin ’09 ’10]  
[Abdulla, Chen, Clemente, et al ’10 ’11]  
Tools: Ranked-based or Ramsey-based complementation |
| **Equality** | Coinduction-based Algorithm  
HKC  [Bonchi, Pous ’13]  
Tools: Powerset construction and up-to techniques | ? |
**HKC [Bonchi,Pous '13]**

**Principle:** Try to compute a bisimulation *up to congruence*.

**Input:** A NFA $\mathcal{A}$ and two sets of states $X, Y$.

**Output:** $true$ if $X$ and $Y$ recognize the same language; $false$ otherwise.

- Explore the powerset construction on the fly.
- Examine pairs of sets of states:
  - Check if the accepting conditions match, if not return $false$.
  - If the pair is in the congruence closure of already seen pairs then skip it.
  - Add successors to the list of pairs to check.
- If no more pair to look at, return $true$. 

![Diagram](https://via.placeholder.com/150)
**HKC [Bonchi,Pous '13]**

**Principle:** Try to compute a bisimulation up to congruence.

**Input:** A NFA $A$ and two sets of states $X, Y$.

**Output:** $true$ if $X$ and $Y$ recognize the same language; $false$ otherwise.

- Explore the powerset construction on the fly
- Examine pairs of sets of states
  - Check if the accepting conditions match, if not return $false$
  - If the pair is in the congruence closure of already seen pairs then skip it
  - Add successors to the list of pairs to check
- If no more pair to look at, return $true$
**Principle:** Try to compute a bisimulation up to congruence.

**Input:** A NFA $\mathcal{A}$ and two sets of states $X, Y$.

**Output:** $true$ if $X$ and $Y$ recognize the same language; $false$ otherwise.

- Explore the powerset construction on the fly
- Examine pairs of sets of states
  - Check if the accepting conditions match, if not return $false$
  - If the pair is in the congruence closure of already seen pairs then skip it
  - Add successors to the list of pairs to check
- If no more pair to look at, return $true$
Principle: Try to compute a bisimulation up to congruence.

Input: A NFA $\mathcal{A}$ and two sets of states $X, Y$.
Output: $true$ if $X$ and $Y$ recognize the same language; $false$ otherwise.

- Explore the powerset construction on the fly
- Examine pairs of sets of states
  - Check if the accepting conditions match, if not return $false$
  - If the pair is in the congruence closure of already seen pairs then skip it
  - Add successors to the list of pairs to check
- If no more pair to look at, return $true$
**HKC [Bonchi,Pous '13]**

**Principle:** Try to compute a bisimulation *up to congruence*.

**Input:** A NFA $\mathcal{A}$ and two sets of states $X, Y$.

**Output:** *true* if $X$ and $Y$ recognize the same language; *false* otherwise.

- Explore the powerset construction on the fly
- Examine pairs of sets of states
  - Check if the accepting conditions match, if not return *false*
  - If the pair is in the congruence closure of already seen pairs then skip it
  - Add successors to the list of pairs to check
- If no more pair to look at, return *true*
Principle: Try to compute a bisimulation up to congruence.

Input: A NFA $A$ and two sets of states $X, Y$.
Output: true if $X$ and $Y$ recognize the same language; false otherwise.

- Explore the powerset construction on the fly
- Examine pairs of sets of states
  - Check if the accepting conditions match, if not return false
  - If the pair is in the congruence closure of already seen pairs then skip it
  - Add successors to the list of pairs to check
- If no more pair to look at, return true
Difficulties for adapting HKC to NBWs

- Non local acceptation conditions
- No proper determinization operation
Ultimately periodic words [Calbrix, Nivat, Podelski '93]

\[ UP(\mathcal{L}) = \{ uv^\omega \mid uv^\omega \in \mathcal{L} \} \quad \mathcal{L}^\$ = \{ u$v \mid uv^\omega \in \mathcal{L} \} \]

**Fact 1**

If \( \mathcal{L}_1 \) and \( \mathcal{L}_2 \) are \( \omega \)-regular then \( \mathcal{L}_1 = \mathcal{L}_2 \) if and only if \( UP(\mathcal{L}_1) = UP(\mathcal{L}_2) \)

▶ **Proof** Closure properties of rational \( \omega \)-languages + non-empty languages contain at least one ultimately periodic word.

**Fact 2**

If \( \mathcal{L} \) is \( \omega \)-regular then \( \mathcal{L}^\$ \) is regular.

▶ **Proof** Construction of Calbrix, Nivat and Podelski.
This paper
Big picture

\[ \omega\text{-regular} \quad \text{ultimately periodic} \quad \text{regular} \]

\[ \mathcal{L} \subseteq A^\omega \quad UP(\mathcal{L}) \subseteq A^\omega \quad \mathcal{L}^\$ \subseteq (A \cup \{\$\})^* \]

\[ \mathcal{L}_1 = \mathcal{L}_2 \quad UP(\mathcal{L}_1) = UP(\mathcal{L}_2) \quad \mathcal{L}_1^\$ = \mathcal{L}_2^\$ \]

\[ \text{Fact 1} \quad \iff \quad \text{Fact 2} \]

\[ \frac{\text{NBW}}{A} \quad \frac{\text{(revisited) construction of Calbrix, Nivat and Podelski}}{[x]A = [y]A} \quad \frac{\text{NFA}}{A^\$} \]

\[ |A| = n \quad |A^\$| = n + n3n^2 \]

\[ [x]A = [y]A \quad [x]A^\$ = [y]A^\$ \]

\[ \Rightarrow \text{HKC on } A^\$ + \text{Exploiting the structure of the construction} \]
Construction of $A^\$ 

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoid of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$L(A) = (a + b)^*ba^\omega$
Construction of $\mathcal{A}^\$ 

Consider a NBW $\mathcal{A}$ with $n$ states.

- Construct the (Büchi) transition monoid of $\mathcal{A}$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$$T_v(x, y) = \begin{cases} 
0 & \text{if } x \overset{v}{\not\rightarrow} y \\
1 & \text{if } x \overset{v}{\rightarrow} y \\
\star & \text{if } x \overset{v}{\Rightarrow} y
\end{cases}$$

$$L(\mathcal{A}) = (a + b)^*ba^\omega$$
Construction of $A^\$$

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoïd of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$x \in \omega(T_v)$ iff $v^\omega \in [x]_A$

iff $x \xrightarrow{v^k} y \xrightarrow{v^{k'}} y$

$L(A) = (a + b)^*ba^\omega$
Construction of $A$

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoïd of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$A$ without accepting conditions
Construction of $A$

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoid of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$
$\omega(I) = \emptyset$

$L(A) = (a + b) \cdot (ba)^* \cdot (a + b)$
Construction of $A^S$

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoid of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

\[
I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \omega(I) = \emptyset
\]

\[
T_a = \begin{pmatrix} 1 & 0 \\ 0 & \ast \end{pmatrix}, \quad \omega(T_a) = \{1\}
\]

\[
T_b = \begin{pmatrix} 1 & \ast \\ 0 & 0 \end{pmatrix}, \quad \omega(T_b) = \emptyset
\]
Construction of $A^S$

Consider a NBW $A$ with $n$ states.

- Construct the (Büchi) transition monoid of $A$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$(x, T_v)$ is accepting iff $x \in \omega(T_v)$
Construction of $\mathcal{A}^\$ 

Consider a NBW $\mathcal{A}$ with $n$ states.

- Construct the (Büchi) transition monoid of $\mathcal{A}$
- Compute the loop structure
- Construct the prefix layer
- Add a $\$ transition from each state
- Copy $n$ times the transition monoid
- Define the final states

$L(\mathcal{A}^\$) = (a + b)^* ba^* a^+$
Exploiting the structure of the construction

\[ L^\$ = \{ u\$v \mid uv^\omega \in \mathcal{L} \} \]

- Second layer deterministic \( \Rightarrow \) no need for congruence on it.
- Second layer consists in \( n \) times the same structure, only final states change \( \Rightarrow \) want to share the computation.
- First layer can be seen as a weighted automaton \( \mathcal{A}^\mathcal{L} \) recognizing \( \mathcal{L}^\mathcal{L} : u \mapsto \{ v \mid uv^\omega \in \mathcal{L} \} \) \( \Rightarrow \) apply a variant of HKC on it.
Changes: Postpone the verifications and return list of non-skipped pairs.

**Input:** A NBW $A$ and two sets of states $X, Y$.

**Output:** A pre-bisimulation up to congruence.

$[0]_A = [1]_A = \text{infinitely many } a\text{'s}$

Pairs: \[\langle \{0\}, \{1\} \rangle ; \langle \{1\}, \{1, 2\} \rangle \]

Skipped pairs: \[\langle \{0, 2\}, \{0\} \rangle ; \langle \{1, 2\}, \{1, 2\} \rangle ; \langle \{0\}, \{0, 2\} \rangle \]
Changes: Postpone the verifications and return list of non-skipped pairs.

**Input:** A NBW $\mathcal{A}$ and two sets of states $X, Y$.

**Output:** A pre-bisimulation up to congruence.

In the congruence closure:
- $\langle \{0\}, \{1, 2\} \rangle$ by transitivity
- $\langle \{2\}, \{2\} \rangle$ by reflexivity
- $\langle \{0, 2\}, \{1, 2\} \rangle$ by union
- $\langle \{1, 2\}, \{1\} \rangle$ and $\langle \{1\}, \{0\} \rangle$ by symmetry
- $\langle \{0, 2\}, \{1\} \rangle$ by transitivity
- $\langle \{0, 2\}, \{0\} \rangle$ by transitivity

$[0]\mathcal{A} = [1]\mathcal{A} = \text{infinitely many } a \text{'s}$

Pairs: $\langle \{0\}, \{1\} \rangle$ ; $\langle \{1\}, \{1, 2\} \rangle$

Skipped pairs: $\langle \{0, 2\}, \{0\} \rangle$ ; $\langle \{1, 2\}, \{1, 2\} \rangle$ ; $\langle \{0\}, \{0, 2\} \rangle$
Comparing the outputs: discriminating sets

A priori need to compare language of words \( \Rightarrow \) infinitely many of them.

**But** All the information needed in the set of \( \omega(T_v) \) \( \Rightarrow \) finite.

**Input:** A NBW \( \mathcal{A} = \langle S, T \rangle \).

**Output:** The set of discriminating sets \( \mathcal{D} = \{ \omega(T_v) | v \in A^* \} \).

\[
\begin{align*}
0 & \xleftarrow{a} 1 \\
0 & \xleftarrow{b} 2 \\
2 & \xleftarrow{b} 1
\end{align*}
\]

- Go through the transition monoid
- Keep in memory the different \( \omega(T_v) \)'s encountered

Transition monoid has 13 elements

\[
\begin{align*}
\mathcal{D} = \emptyset ; \ {0, 1} ; \ {0, 1, 2} \\
\omega(T_b) &= \emptyset \quad \omega(T_a) = \{0, 1\} \quad \omega(T_{ba}) = \{0, 1, 2\}
\end{align*}
\]
Global Algorithm

Compute the Pairs \( \parallel \) Compute the discriminating sets \( \mathcal{D} \)

For all pair \( \langle X, Y \rangle \) and all discriminating set \( \mathcal{D} \), check that:

\[
X \cap \mathcal{D} = \emptyset \iff Y \cap \mathcal{D} = \emptyset
\]

Pairs = \( \langle \{0\}, \{1\} \rangle ; \langle \{1\}, \{1, 2\} \rangle \)

\( \mathcal{D} = \emptyset ; \{0, 1\} ; \{0, 1, 2\} \)
Conclusion

A coinductive-based algorithm to solve the language equality problem for Büchi automata.


Advantages:

▶ Two independent and parallel parts
▶ Advanced up-to techniques to study the prefix layer
▶ Sharing of information to study the periodic layer

Drawback:

▶ Need to explore the whole transition monoid to find discriminating sets
Future work

- Design up-to techniques to take advantage of the structure of the periodic layer

- Adapt known optimization for already existing algorithms on NBW