Coinduction based algorithm to decide Büchi automata equivalence

Laureline Pinault
supervised by Denis Kuperberg and Damien Pous

M2 internship, ENS de Lyon
February 2017 - June 2017
Introduction: Büchi automata

→ Accepts words having infinitely many a's and infinitely many b's
Motivations

**PRINTER DRIVER**

- W: wait, r: request, c: cancel, g: grant, f: finish

**LTL FORMULA**

\[ G[r \Rightarrow XF \neg w] \]
# Table of Contents

1. Equivalence of finite automata [BP13]

2. From Büchi automata to finite automata [CNP93]

3. Equivalence of Büchi automata

4. Conclusion
Table of Contents

1. Equivalence of finite automata [BP13]

2. From Büchi automata to finite automata [CNP93]

3. Equivalence of Büchi automata

4. Conclusion
HK algorithm on DFAs

INPUT: 1 DFA, 2 states $x$ and $y$ ; OUTPUT: $x \sim y$?

I

II

III

$1 \xrightarrow{a, b} 2 \xrightarrow{a, b} 3 \xrightarrow{a, b}$

$1 \xrightarrow{a} II \xrightarrow{a, b} III \xrightarrow{a, b} III \xrightarrow{b}$
HK algorithm on DFAs

IDEA: Assume $x \sim y$ and see if there is a contradiction

[Diagram showing transitions between states with labels $a, b$]
IDEA: Assume $x \sim y$ and see if there is a contradiction
IDEA: Assume $x \sim y$ and see if there is a contradiction
IDEA: Assume \( x \sim y \) and see if there is a contradiction
HK algorithm on DFAs

IDEA: Assume $x \sim y$ and see if there is a contradiction
**IDEA:** Assume $x \sim y$ and see if there is a contradiction
IDEA: Assume $x \sim y$ and see if there is a contradiction

Relation closed under equivalence
HK algorithm on DFAs

IDEA: Assume $x \sim y$ and see if there is a contradiction

Relation closed under equivalence
HK algorithm on NFAs

→ Run the algorithm on the powerset

\[ \]

1 \rightarrow 2 \rightarrow 3

I

\[ \]

1 \rightarrow 2, 3 \rightarrow 1, 2 \rightarrow 1, 2, 3

I

1 \rightarrow a

1 \rightarrow a
HK algorithm on NFAs

→ Run the algorithm on the powerset
HK algorithm on NFAs

→ Run the algorithm on the powerset
HK algorithm on NFAs

→ Run the algorithm on the powerset
HK algorithm on NFAs

→ Run the algorithm on the powerset
HKC algorithm on NFAs

Laureline Pinault
HKC algorithm on NFAs

Laureline Pinault

Algorithm for Büchi automata equivalence
HKC algorithm on NFAs
HKC algorithm on NFAs

Laureline Pinault
HKC algorithm on NFAs
HKC algorithm on Büchi automata?
Table of Contents

1. Equivalence of finite automata [BP13]

2. From Büchi automata to finite automata [CNP93]

3. Equivalence of Büchi automata

4. Conclusion
Ultimately periodic words

Words of shape $u \cdot v^\omega$

$u = u_0 u_1 u_2 u_3 u_4 u_5 u_6 u_7 \ldots$ : accepted by a Büchi automaton

$q_0 q_1 q_2 q_f q_4 q_5 q_6 q_f q_8 \ldots$ : an accepting run

$u_0 u_1 u_2 (u_3 u_4 u_5 u_6)^\omega$ accepted by $q_0 q_1 q_2 q_f (q_4 q_5 q_6 q_f)^\omega$

→ Any non-empty rational language has a ultimately periodic word
Equivalence of language equivalence

Corollary

\[ L_1 = L_2 \iff UP(L_1) = UP(L_2) \quad \text{L}_1, \text{L}_2 \text{ rationals} \]

\[ UP((L_1 \cup L_2) \setminus (L_1 \cap L_2)) = \emptyset \]

\[ \Rightarrow (L_1 \cup L_2) \setminus (L_1 \cap L_2) = \emptyset \]

\[ \Rightarrow L_1 = L_2 \]
Rationality of ultimately periodic languages

\[ \text{UP}(\mathcal{L}) = u \cdot v^\omega \rightsquigarrow u \cdot \$ \cdot v = \mathcal{L}_\$ \]

\[ \mathcal{L}_\$ = \bigcup_y \mathcal{M}_{x,y} \cdot \$ \cdot \mathcal{N}_y \]

\[ A \]
Issues when constructing $\mathcal{A}_{N_y}$

Need to read $(ab)^3$

Need to read $abab$ and then $ab$
Construction of $\mathcal{A}_{N_y}$

\[
\begin{pmatrix}
1, 0 \\
2, 0 \\
3, 0 \\
4, 0 \\
5, 0 \\
\end{pmatrix} \xrightarrow{a} \begin{pmatrix}
2, 0 \\
\bot \\
1, 1 \\
5, 0 \\
\bot \\
\end{pmatrix} \xrightarrow{b} \begin{pmatrix}
3, 0 \\
\bot \\
4, 1 \\
1, 1 \\
\bot \\
\end{pmatrix}
\]

$1 \xrightarrow{ab} 3, 3 \xrightarrow{ab} 4, 4 \xrightarrow{ab} 1$
Construction of $A_\$
Construction of \( A_\$ \)
Construction of $A_\$
Construction of $A_\$$

Same structure but different accepting conditions
Table of Contents

1. Equivalence of finite automata [BP13]

2. From Büchi automata to finite automata [CNP93]

3. Equivalence of Büchi automata

4. Conclusion
How the algorithm works

→ We can run HKC on $A_S$
1st improvement: pre-processing

\[ \begin{align*}
\forall i, \quad o(q_i) &= o(r_i) \quad ? \\
\end{align*} \]
2nd improvement: state compression

$A_N$: contains less states that we would think

$$
\begin{pmatrix}
1, 0 \\
2, 0 \\
3, 0 \\
4, 0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
2, 1 \\
3, 0 \\
3, 1 \\
3, 1
\end{pmatrix}
$$
2nd improvement: state compression

\[ A_N \]

contains less states that we would think

- \( (1, 0) \)
- \( (2, 0) \)
- \( (3, 0) \)
- \( (4, 0) \)

\[ \begin{array}{c}
(4, 0) \\
3, 0 \\
3, 1 \\
1, 0 \\
\end{array} \]

\[ \begin{array}{c}
(2, 1) \\
3, 0 \\
3, 1 \\
\end{array} \]

\[ \begin{array}{c}
3, 1 \\
\end{array} \]

\[ \begin{array}{c}
(4, 0) \\
3, 0 \\
3, 1 \\
3, 1 \\
\end{array} \]
2nd improvement: state compression

\( A_N \): contains less states that we would think

\[
\begin{bmatrix}
(1, 0) & (2, 0) & (3, 0) & (4, 0) \\
(2, 1) & (3, 0) & (3, 1) & (3, 1) \\
(3, 0) & (3, 1) & (3, 1) & (3, 1) \\
(4, 0) & (2, 1) & (3, 0) & (3, 1) \\
\end{bmatrix}
\]
2nd improvement: state compression

\[ A_N? \ :	ext{ contains less states that we would think } \quad 2^{(2^m)^m} \rightarrow (2^2)^m \]
1st issue: description of the automaton

\[(4, 0), (2, 1), (3, 1)\]  
\[(3, 0)\]  
\[(3, 1)\]  
\[(1, 0), (3, 1)\]
1st issue: description of the automaton

\[(4, 0), (2, 1), (3, 1), (3, 0), (3, 1), (1, 0), (3, 1)\]

1. Compute the strongly connected components

```plaintext
1 2 3 4
1 2 3 3
```
1st issue: description of the automaton

\[
\begin{pmatrix}
(4, 0), (2, 1), (3, 1) \\
(3, 0) \\
(3, 1) \\
(1, 0), (3, 1)
\end{pmatrix}
\]

1. Compute the strongly connected components
2. Keep the ones having a final edge
### 1st issue: description of the automaton

\[
\begin{pmatrix}
(4, 0), (2, 1), (3, 1) \\
(3, 0) \\
(3, 1) \\
(1, 0), (3, 1)
\end{pmatrix}
\]

1. Compute the strongly connected components
2. Keep the ones having a final edge
3. Reverse the edges
1st issue: description of the automaton

\[
\begin{pmatrix}
(4, 0), (2, 1), (3, 1) \\
(3, 0) \\
(3, 1) \\
(1, 0), (3, 1)
\end{pmatrix}
\]

1. Compute the strongly connected components
2. Keep the ones having a final edge
3. Reverse the edges
4. Compute the connected components
2nd issue: computation of the congruence

→ The problem becomes NP-Complete

→ Use of a SAT-Solver
### Table of Contents

1. Equivalence of finite automata [BP13]
2. From Büchi automata to finite automata [CNP93]
3. Equivalence of Büchi automata
4. Conclusion
Summary

\[ A \text{ : Büchi ND} \]
\[ m \text{ states} \]

\[ A_\$ \text{ : NFA} \]
\[ m + m(2m)^m \text{ states} \]
Summary

\[ A : \text{Büchi ND} \]
\[ m \text{ states} \]

\[ A_\$ : \text{NFA} \]
\[ m + m(2m)^m \text{ states} \]

\[ \rightarrow \text{HKC on} \]

\[ \text{Det}(A_\$) \]
\[ 2^m + m(2m)^m \text{ states} \]
Summary

\[ \mathcal{A} : \text{Büchi ND} \]
\[ m \text{ states} \]

\[ \mathcal{A}_S : \text{NFA} \]
\[ m + m(2m)^m \text{ states} \]

\[ \rightarrow \text{HKC on} \]
\[ \text{Det}(\mathcal{A}_S) \]
\[ 2^m + m(2m)^m \text{ states} \]

\[ \rightarrow \text{HKC on} \]
\[ \text{Det}(\mathcal{A}_N?) \]
\[ 2^{(2m)^m} \text{ states} \]
\[ + \text{HKC on} \]
\[ \text{Det}(\mathcal{A}) \]
\[ 2^m \text{ states} \]
Summary

$A : \text{Büchi ND}$

$m$ states

$\xrightarrow{\text{HKC on}}$

$\text{Det}(A)$

$2^{m+m(2m)^m}$ states

$A_{\text{ND}}$ : Büchi ND

$m$ states

$\xrightarrow{\text{HKC on}}$

$\text{Det}(A_{\text{ND}})$

$2^{(2m)^m}$ states

$\xrightarrow{\text{HKC on}}$

$A'_{\text{ND}} : \text{DFA}$

$2^{2m^2}$ states

$\xrightarrow{\text{HKC on}}$

$\text{Det}(A)$

$2^m$ states

$\xrightarrow{\text{HKC on}}$

$\text{Det}(A)$

$2^m$ states
Future Work

- Implementation and comparison with existing methods
- Further improvements of the algorithm
- Extension to other automata classes
Thank You
