# An efficient algorithm to decide periodicity of $b$-recognisable sets using MSDF convention


Victor Marsault

joint work with Bernard Boigelot , Isabelle Mainz and Michel Rigo

Montefiore Institute and Department of Mathematics,
University of Liège, Belgium



ICALP 2017
Warsaw

# Plan

# Integer base numeration systems



- $b > 1$
- *Alphabet used to represent numbers:* $\{0, 1, \ldots, b - 1\}$

# Integer base numeration systems



- $b > 1$
- *Alphabet used to represent numbers:* $\{0, 1, \ldots, b-1\}$

- $$\begin{array}{rcccl} \textsc{val} & : & \{0,1,\ldots,b-1\}^* & \longrightarrow & \mathbb{N} \\ & & x_n \cdots x_1 x_0 & \longmapsto & x_n b^n + \cdots + x_1 b^1 + x_0 b^0 \end{array}$$

In base $b = 2$, $\quad \textsc{val}(010011) \; = \; 0 + 2^3 + 0 + 0 + 2^1 + 2^0 = 19$ .

# Integer base numeration systems



- $b > 1$
- *Alphabet used to represent numbers:* $\{0, 1, \ldots, b-1\}$

- $$\begin{array}{lcccl} \mathrm{VAL} & : & \{0,1,\ldots,b-1\}^* & \longrightarrow & \mathbb{N} \\ & & x_n \cdots x_1 x_0 & \longmapsto & x_n b^n + \cdots + x_1 b^1 + x_0 b^0 \end{array}$$

In base $b = 2$, $\mathrm{VAL}(010011) = 0 + 2^3 + 0 + 0 + 2^1 + 2^0 = 19$ .

- $$\begin{array}{lcccl} \mathrm{REP} & : & \mathbb{N} & \longrightarrow & \{0,1,\ldots,b-1\}^* \\ & & 0 & \longmapsto & \varepsilon \\ & & n > 0 & \longmapsto & \mathrm{REP}(m)\, d \,, \end{array}$$

  *where* $(m, d)$ *is the Eucl. div of* $n$ *by* $b$.

In base 2, $\mathrm{REP}(19) = \mathrm{REP}(9)\,1 = \mathrm{REP}(4)\,11 = \cdots = 10011$ .

### Definition

*$X$: a set of integers.*

---

*$X$ is $b$-**recognisable** if $\mathrm{REP}(X)$ is a regular language.*

# *b*-recognisable sets



## Definition

*X: a set of integers.*

---

*X is b-**recognisable** if* $0^* \mathrm{REP}(X)$ *is a regular language.*

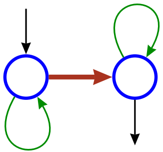
### Definition

*$X$: a set of integers.*

---

*$X$ is $b$-**recognisable** if $0^*\mathrm{REP}(X)$ is a regular language.*

Ex.: the powers of two form a 2-recognisable set:



Automaton accepting $0^*\mathrm{REP}(2^{\mathbb{N}})$


Legend

# $b$-recognisable sets (2)



**Theorem (folklore)**

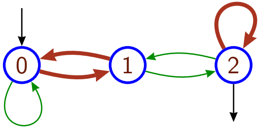
*Eventually periodic sets are $b$-recognisable in all base $b$.*

## Theorem (folklore)

*Eventually periodic sets are $b$-recognisable in all base $b$.*

## Example: $R + p\mathbb{N}$

- *Alph.:* $\{0, \ldots, b-1\}$
- *State set:* $\mathbb{Z}/p\mathbb{Z}$
- *Initial state:* 0
- *Transitions:*
  $\forall$ *state* $s$, $\forall$ *digit* $x$
  $$s \xrightarrow{x} sb + x$$
- *Final-state set:* $R$




Example 1: $p = 3$ , $R = \{2\}$

# $b$-recognisable sets (2)

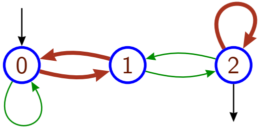


## Theorem (folklore)

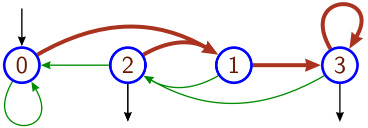
*Eventually periodic sets are $b$-recognisable in all base $b$.*

## Example: $R + p\mathbb{N}$

- *Alph.:* $\{0, \ldots, b{-}1\}$
- *State set:* $\mathbb{Z}/p\mathbb{Z}$
- *Initial state:* 0
- *Transitions:*
  $\forall$ *state* $s$, $\forall$ *digit* $x$
  $$s \xrightarrow{x} sb + x$$
- *Final-state set:* $R$




Example 1: $p = 3$ , $R = \{2\}$




Example 2: $p = 4$ , $R = \{2, 3\}$

# $b$-recognisable sets (2)



**Theorem (Cobham, 1969)**

*$b, c$ : two integer bases, multiplicatively independent*$^\dagger$.
*$X$ : a set of integers.*

$$\left.\begin{array}{l} X \text{ is } b\text{-recognisable} \\ X \text{ is } c\text{-recognisable} \end{array}\right\} \Longrightarrow X \text{ is eventually periodic}$$

$^\dagger$such that $b^i \neq c^j$ for all $i, j > 0$.

$$\left\{ \text{ Eventually periodic sets } \right\} \quad = \quad \left\{ \text{ Sets } b\text{-recognisable for all } b \right\}$$

# PERIODICITY problem



## PERIODICITY

- ***Parameter***: *an integer base* $b > 1$.
- ***Input***: *a deterministic finite automaton* $\mathcal{A}$
  *(hence the* $b$*-recognisable set* $X$ *accepted by* $\mathcal{A}$*).*
- ***Question***: *is* $X$ *eventually periodic ?*

## Theorem (Honkala, 1986)

PERIODICITY *is decidable.*

# Restating PERIODICITY in terms of logic



**Theorem**

*X: a set of integers*

---

*X is eventually periodic* $\iff$ *X is definable in FO*$[\mathbb{N}, +]$

# Restating PERIODICITY in terms of logic



## Theorem

*X: a set of integers*

*X is eventually periodic* $\iff$ *X is definable in FO*$[\mathbb{N}, +]$

## Definition

$V_b$ : *function* $\mathbb{N} \to \mathbb{N}$ *that maps n to the greatest* $b^j$ *that divides n*

Ex. $V_2(2017) = 1$ and $V_2(2016) = V_2(32 \times 63) = 32$

# Restating PERIODICITY in terms of logic



**Theorem**

*$X$: a set of integers*

$X$ *is eventually periodic* $\iff$ $X$ *is definable in* $FO[\mathbb{N}, +]$

**Definition**

$V_b$ : *function* $\mathbb{N} \to \mathbb{N}$ *that maps* $n$ *to the greatest* $b^j$ *that divides* $n$

Ex. $V_2(2017) = 1$ and $V_2(2016) = V_2(32 \times 63) = 32$

**Theorem [Büchi 1960] [Bruyère 1985]**

*$X$: a set of integers*

$X$ *is b-recognisable* $\iff$ $X$ *is definable in* $FO[\mathbb{N}, +, V_b]$

# Restating PERIODICITY in terms of logic (2)



### PRESBUGER-DEFINABLE

- ***Parameter***: *an integer base* $b > 1$.
- ***Input***: *a formula* $\Phi$ *in* $FO[\mathbb{N}, +, V_b]$.
- ***Question***: *is there a formula of* $FO[\mathbb{N}, +]$ *equivalent to* $\Phi$ *?*

# Restating PERIODICITY in terms of logic (2)



**PRESBUGER-DEFINABLE**

- ***Parameter***: *an integer base* $b > 1$.
- ***Input***: *a formula* $\Phi$ *in* $FO[\mathbb{N}, +, V_b]$.
- ***Question***: *is there a formula of* $FO[\mathbb{N}, +]$ *equivalent to* $\Phi$ *?*

PERIODICITY is equivalent to **1**-PRESBURGER-DEFINABLE
($\Phi$ has **1** free variable)

# Best algorithms to solve PERIODICITY



**Least Significant Digit First (LSDF)** convention: the input automaton reads its entry "from right to left".

# Best algorithms to solve PERIODICITY



**Least Significant Digit First (LSDF)** convention: the input automaton reads its entry "from right to left".

### Theorem

*With LSDF convention,*

- PRESBUGER-DEFINABLE *is P-TIME* *[Leroux 2005]*
- PERIODICITY *is Linear-TIME if the input is minimal* *[M-Sakarovitch 2013].*

# Best algorithms to solve PERIODICITY



**Least Significant Digit First (LSDF)** convention: the input automaton reads its entry "from right to left".

### Theorem

*With LSDF convention,*

- PRESBUGER-DEFINABLE *is P-TIME* *[Leroux 2005]*
- PERIODICITY *is Linear-TIME if the input is minimal* *[M-Sakarovitch 2013].*

### Remark

*Making an automaton reads from right to left*
*requires a transposition and a determinisation*
$\Rightarrow$ *Exponential blow-up*

# Our contribution



**Theorem**

*PERIODICITY is decidable in $O(b\, n \log(n))$ time*

*(where $n$ is the state-set cardinal.)*

# Plan

# Pseudo-morphisms (1)



## Definition

*$\mathcal{A}, \mathcal{M}$: two complete DFA*

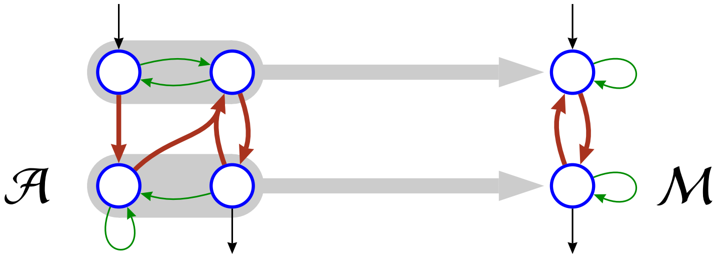
*$\varphi$: a function {states of $\mathcal{A}$} $\rightarrow$ {states of $\mathcal{M}$}*

---

*$\varphi$ is a* ***pseudo-morphism*** *$\mathcal{A} \rightarrow \mathcal{M}$ if*

- *$\varphi$ maps the initial state of $\mathcal{A}$ to the initial state of $\mathcal{M}$*
- $s \xrightarrow{a} s'$ *in* $\mathcal{A}$ $\iff$ $\varphi(s) \xrightarrow{a} \varphi(s')$ *in* $\mathcal{M}$

(A pseudo-morphism is a morphism with no condition on final states.)



# Pseudo-morphisms (1)

## Definition

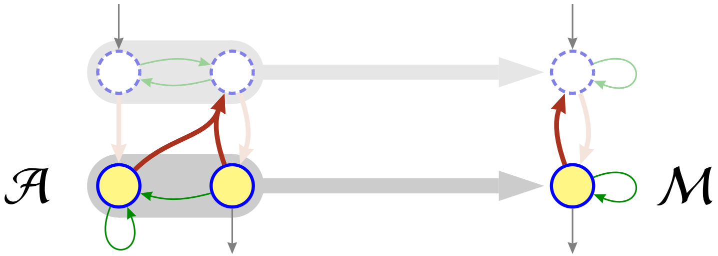
*$\mathcal{A}, \mathcal{M}$: two complete DFA*
*$\varphi$: a function {states of $\mathcal{A}$} $\to$ {states of $\mathcal{M}$}*

---

*$\varphi$ is a* ***pseudo-morphism*** *$\mathcal{A} \to \mathcal{M}$ if*

- *$\varphi$ maps the initial state of $\mathcal{A}$ to the initial state of $\mathcal{M}$*
- *$s \xrightarrow{a} s'$ in $\mathcal{A}$ $\iff$ $\varphi(s) \xrightarrow{a} \varphi(s')$ in $\mathcal{M}$*

(A pseudo-morphism is a morphism with no condition on final states.)



# Pseudo-morphisms (1)



## Definition

$\mathcal{A}, \mathcal{M}$: *two complete DFA*

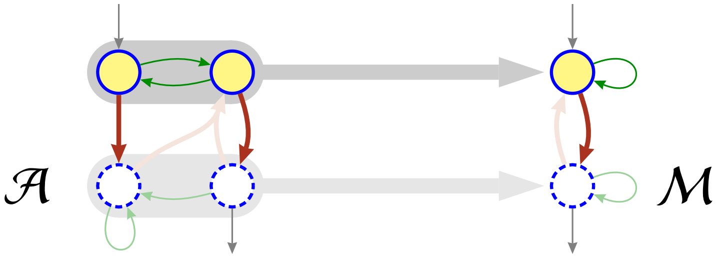
$\varphi$: *a function* $\{$*states of* $\mathcal{A}\} \to \{$*states of* $\mathcal{M}\}$

---

$\varphi$ *is a* ***pseudo-morphism*** $\mathcal{A} \to \mathcal{M}$ *if*

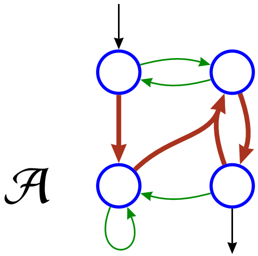
- $\varphi$ *maps the initial state of* $\mathcal{A}$ *to the initial state of* $\mathcal{M}$
- $s \xrightarrow{a} s'$ *in* $\mathcal{A} \iff \varphi(s) \xrightarrow{a} \varphi(s')$ *in* $\mathcal{M}$

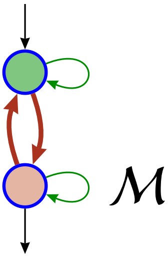
(A pseudo-morphism is a morphism with no condition on final states.)



**Lemma**

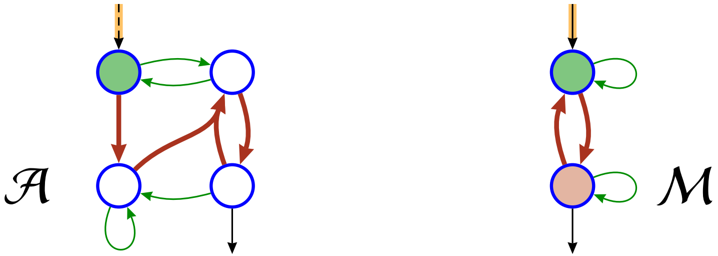
*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*



### Lemma

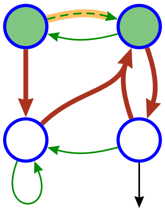
*Computing the pseudo-morphism* $\varphi : \mathcal{A} \to \mathcal{M}$*, if it exists, may be done in* $O(b\,n)$ *time.*

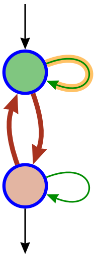


**Lemma**

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*

$\mathcal{A}$



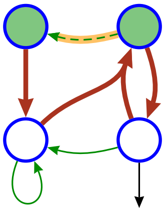


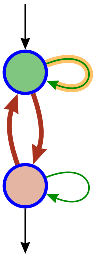
$\mathcal{M}$

**Lemma**

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*

$\mathcal{A}$

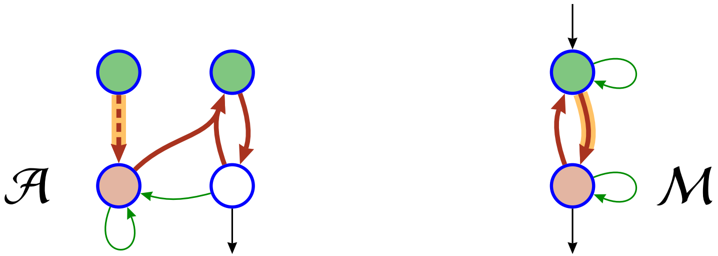




$\mathcal{M}$

**Lemma**

*Computing the pseudo-morphism $\varphi : \mathcal{A} \rightarrow \mathcal{M}$, if it exists, may be done in $O(b\, n)$ time.*

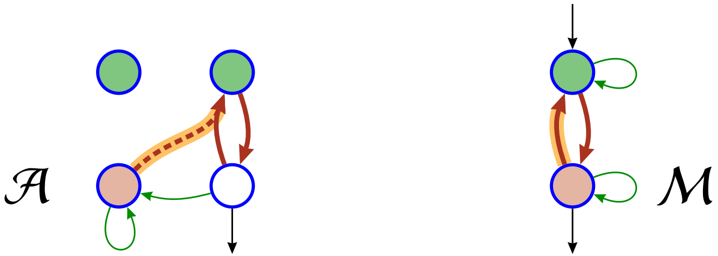


# Pseudo-morphisms (2)



## Lemma

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*

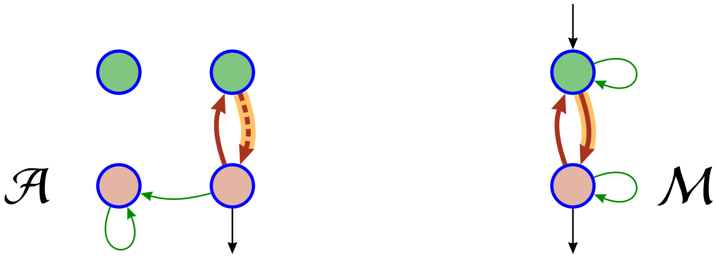


# Pseudo-morphisms (2)



**Lemma**

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*

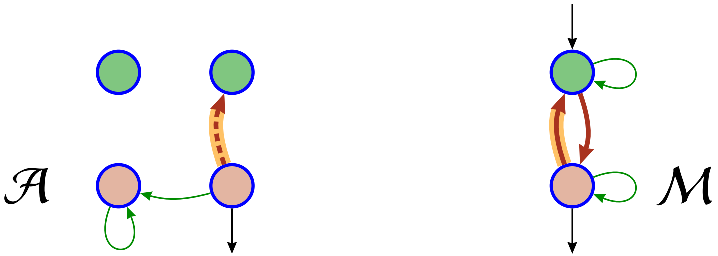


# Pseudo-morphisms (2)



**Lemma**

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*

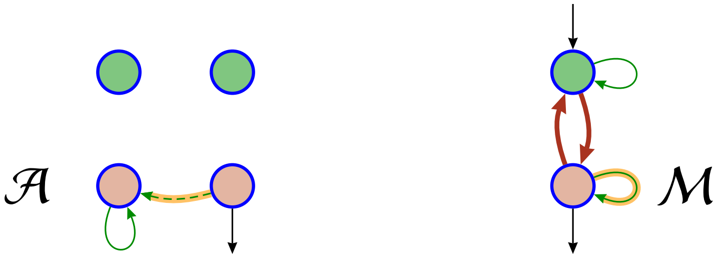


# Pseudo-morphisms (2)



**Lemma**

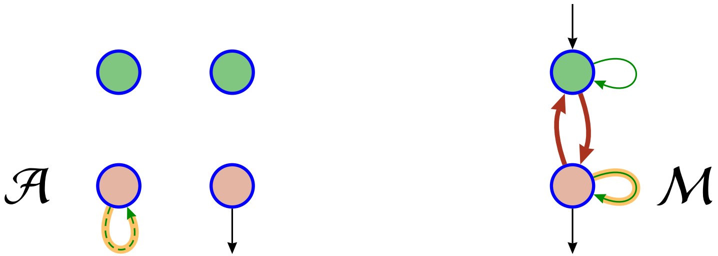
*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*



# Pseudo-morphisms (2)

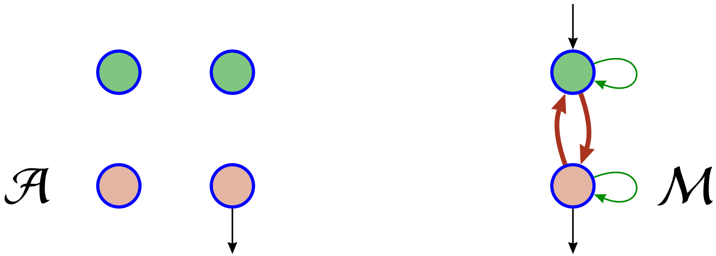
## Lemma

*Computing the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{M}$, if it exists, may be done in $O(b\,n)$ time.*



**Lemma**

*Computing the pseudo-morphism* $\varphi : \mathcal{A} \to \mathcal{M}$*, if it exists, may be done in* $O(b\,n)$ *time.*



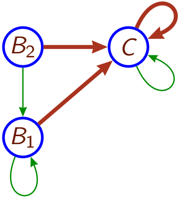
# Ultimate Equivalence (1)



## Definition

*$\mathcal{A}$: a complete DFA.*
*$s, s'$: states of $\mathcal{A}$.*
*$m$: an integer.*

*$s$ and $s'$ are $m$-ultimately-equivalent (w.r.t. $\mathcal{A}$),*
*if $\forall$word $u$ of length $m$, [ $s \xrightarrow{u} t$ and $s' \xrightarrow{u} t'$ implies $t = t'$ ].*




- $B_1$ and $B_2$ are 1-ult.-equiv.
- All others pairs are not ult.-equiv., as witnessed by the family $0^*$.

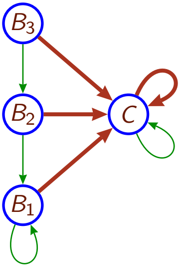
# Ultimate Equivalence (1)



**Definition**

*$\mathcal{A}$: a complete DFA.*
*$s, s'$: states of $\mathcal{A}$.*
*$m$: an integer.*

*$s$ and $s'$ are $m$-ultimately-equivalent (w.r.t. $\mathcal{A}$),*
*if $\forall$word $u$ of length $m$, [ $s \xrightarrow{u} t$ and $s' \xrightarrow{u} t'$ implies $t = t'$ ].*

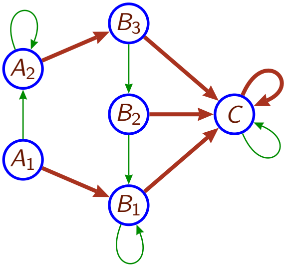



- $B_1$ and $B_2$ are 1-ult.-equiv.
- $B_2$ and $B_3$ are 2-ult.-equiv.
- $B_3$ and $B_1$ are 2-ult.-equiv.
- All others pairs are not ult.-equiv., as witnessed by the family $0^*$.

## Definition

*$\mathcal{A}$: a complete DFA.*
*$s, s'$: states of $\mathcal{A}$.*
*$m$: an integer.*

*$s$ and $s'$ are $m$-ultimately-equivalent (w.r.t. $\mathcal{A}$),*
*if $\forall$word $u$ of length $m$, $[\ s \xrightarrow{u} t$ and $s' \xrightarrow{u} t'$ implies $t = t'\ ]$.*




- $B_1$ and $B_2$ are 1-ult.-equiv.
- $B_2$ and $B_3$ are 2-ult.-equiv.
- $B_3$ and $B_1$ are 2-ult.-equiv.
- $A_1$ and $A_2$ are 3-ult.-equiv.
- All others pairs are not ult.-equiv., as witnessed by the family $0^*$.

# Ultimate Equivalence (2)



$\mathcal{A}$: a DFA.
$n$: the number of states in $\mathcal{A}$.
$b$: the size of the alphabet.

By using the automaton product $\mathcal{A} \times \mathcal{A}$, it is known that:

**Lemma (folklore)**

*Ultimate-equivalence relation of $\mathcal{A}$ can be computed in $O(bn^2)$ time.*

There exists a better algorithm:

**Theorem (Béal-Crochemore, 2007)**

*Ultimate-equivalence relation of $\mathcal{A}$ can be computed in $O(b\, n \log(n))$ time.*

# Plan

# Characterisation theorem



$\mathcal{A}_p$ denotes the naïve automaton accepting $p\mathbb{N}$.

## Theorem

*$\mathcal{A}$: a minimal DFA.*
*$X$: the b-recognisable set accepted by $\mathcal{A}$.*
*$\ell$: the total number of states in 0-circuits.*

*$X$ is purely periodic if and only if*

- *$\exists$ a pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_{(\ell,\emptyset)}$;*
- *states $s, s'$ such that $\varphi(s) = \varphi(s')$, are ultimately equivalent;*
- *the initial state of $\mathcal{A}$ bears a 0-loop.*

# Execution on an example



0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits.

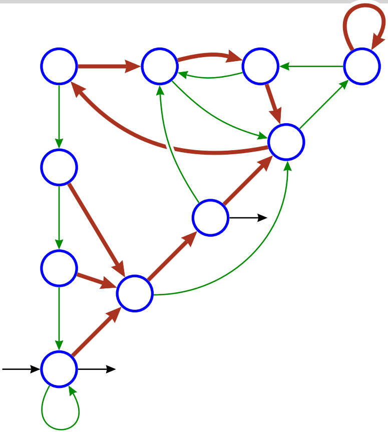
2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example





0 Start from a minimal complete DFA $\mathcal{A}$.

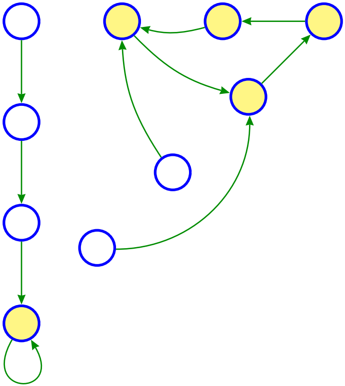
1 Count the number $\ell$ of states in 0-circuits.

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example



0 Start from a minimal complete DFA $\mathcal{A}$.

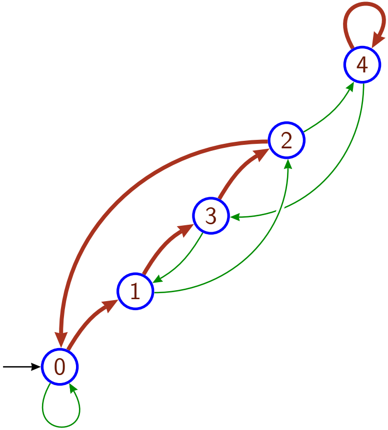
1 Count the number $\ell$ of states in 0-circuits. $\boldsymbol{\ell = 5}$

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example




0 Start from a minimal complete DFA $\mathcal{A}$.

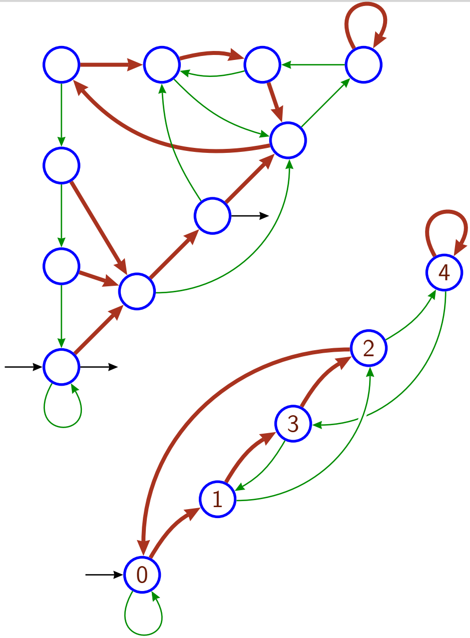
1 Count the number $\ell$ of states in 0-circuits. **$\ell = 5$**

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example




0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits. $\ell = 5$

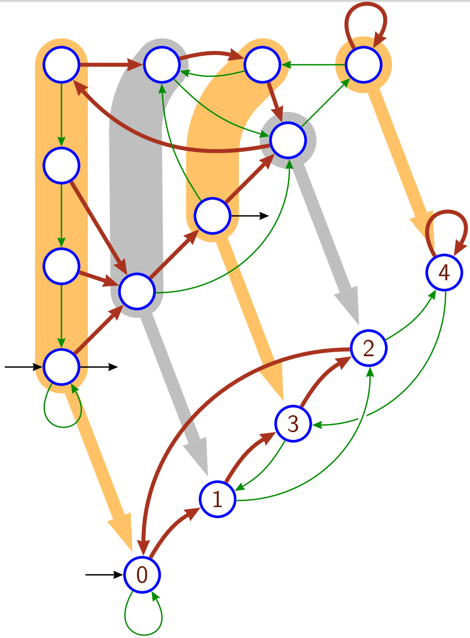
2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example






0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits. $\boldsymbol{\ell = 5}$

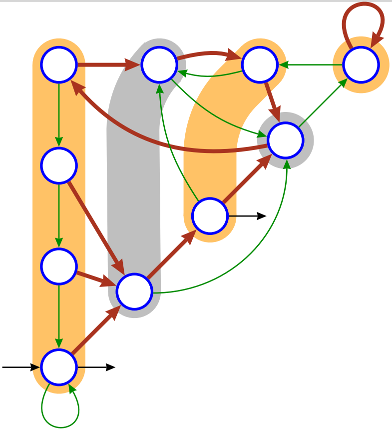
2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example





0 Start from a minimal complete DFA $\mathcal{A}$.

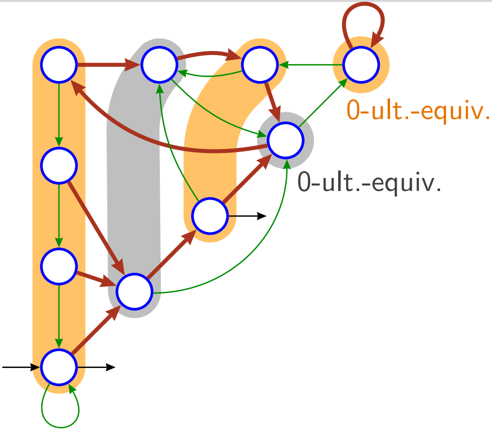
1 Count the number $\ell$ of states in 0-circuits. $\ell = 5$

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example




0 Start from a minimal complete DFA $\mathcal{A}$.

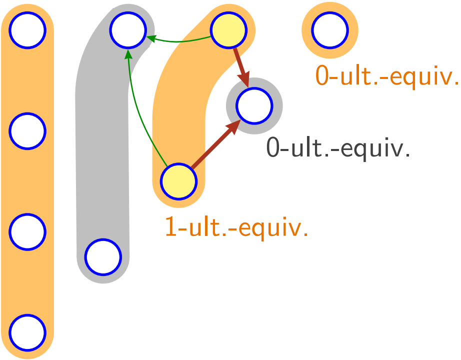
1 Count the number $\ell$ of states in 0-circuits. $\boldsymbol{\ell = 5}$

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example




0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits. $\ell = 5$

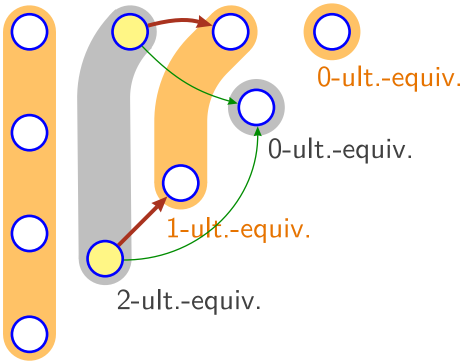
2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example






0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits. $\ell = 5$

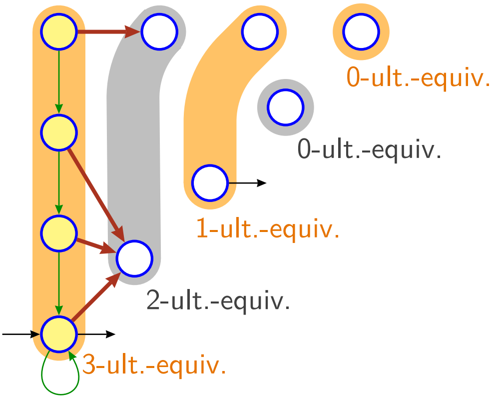
2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example






0 Start from a minimal complete DFA $\mathcal{A}$.

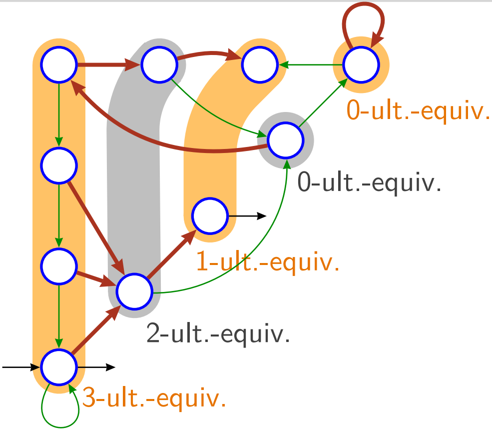
1 Count the number $\ell$ of states in 0-circuits. $\boldsymbol{\ell = 5}$

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Execution on an example




( Then, the period is $b^m \times \ell = 2^3 \times 5 = 40$ )

0 Start from a minimal complete DFA $\mathcal{A}$.

1 Count the number $\ell$ of states in 0-circuits. $\ell = 5$

2 Build $\mathcal{A}_\ell$.

3 Compute the pseudo-morphism $\varphi : \mathcal{A} \to \mathcal{A}_\ell$.

4 Check that states $s, t$ such that $\varphi(s) = \varphi(t)$ are ult-equiv.

# Conclusion



## Main theorem

PERIODICITY *is decidable in* $O(b\, n \log(n))$ *time*

*(where* $n$ *is the state-set cardinal.)*

# Conclusion



## Main theorem

PERIODICITY *is decidable in* $0(b\, n\, log(n))$ *time*

*(where n is the state-set cardinal.)*

## Possible future work

- Design efficient data structure for integer set.
- Consider sets of **real** numbers.
- Extend result to multi-dimensional sets of $\mathbb{N}^k$
- Represent integers with a non-standard numeration systems.

# $\mathcal{A}_{(12,\{5,7\})}$ as the product $\mathcal{A}_{(4,?)} \times \mathcal{A}_{(3,?)}$