CIRM winter school - Discrete mathematics and logic: between mathematics and the computer science

Algebraic approach to register minimization in streaming string transducers

Yahia Idriss BENALIOUA

Nathan LHOTE & Pierre-Alain REYNIER

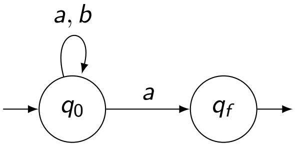
Aix Marseille université Socialement engagée

LABORATOIRE D'INFORMATIQUE & SYSTÈMES UMR 7020

January 18, 2023

Table of contents

Regular languages

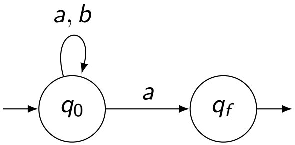


Automaton recognizing $(a+b)^*a$

Other characterizations:

- Algebra : syntactic congruence
- Regular expressions
- Logic : MSO

Regular languages

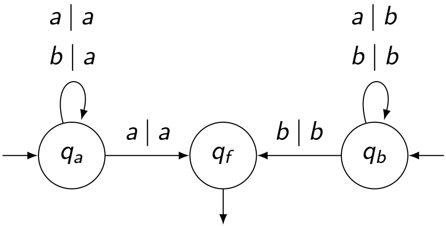


Automaton recognizing $(a+b)^*a$

Other characterizations:

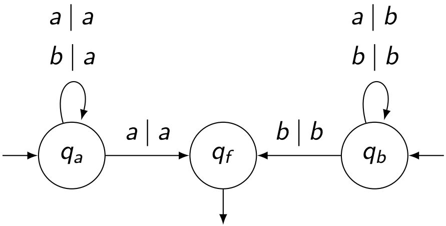
- Algebra : syntactic congruence
- Regular expressions
- Logic : MSO

Transducers



Transducer realizing the function *last*

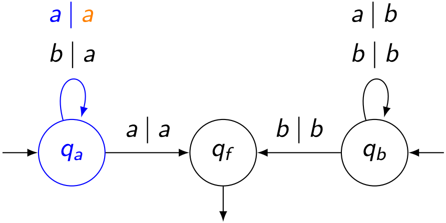
Transducers



Transducer realizing the function *last*

aba

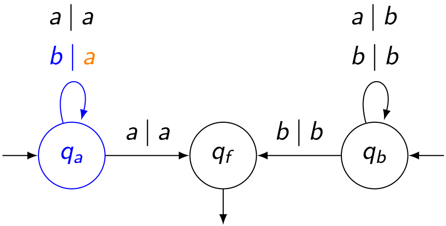
Transducers



Transducer realizing the function *last*

$$aba \longmapsto a$$

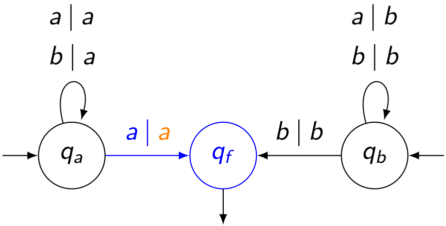
Transducers



Transducer realizing the function *last*

$$aba \longmapsto aa$$

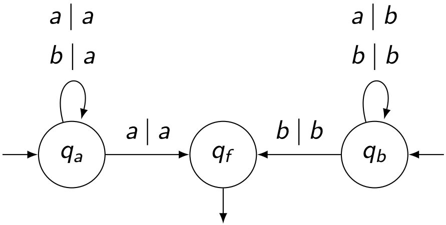
Transducers



Transducer realizing the function *last*

$$aba \longmapsto aaa$$

Transducers



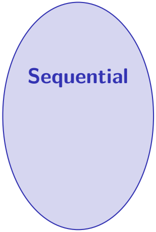
Transducer realizing the function *last*

$$last(aba) = aaa$$

Classes of functions

Sequential functions

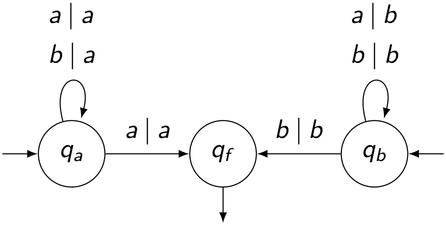
Deterministic underlying automaton

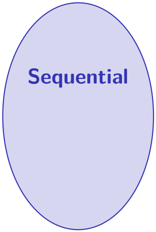


Classes of functions

Sequential functions

Deterministic underlying automaton

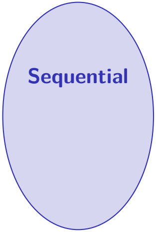


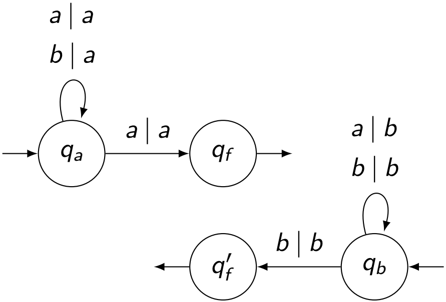


Classes of functions

Sequential functions

Deterministic underlying automaton

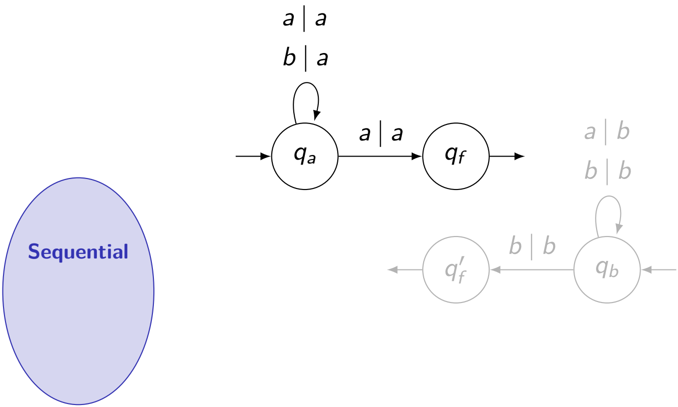




Classes of functions

Sequential functions

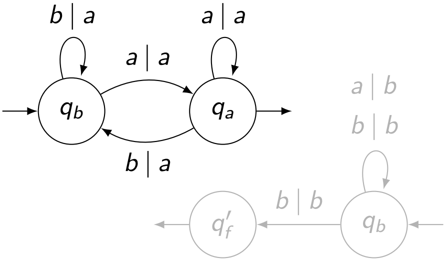
Deterministic underlying automaton

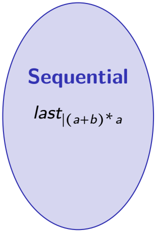


Classes of functions

Sequential functions

Deterministic underlying automaton

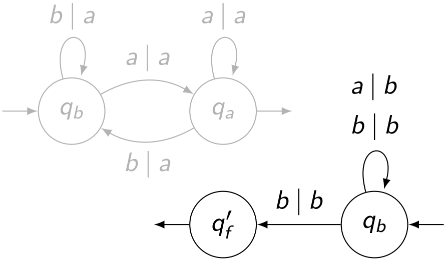


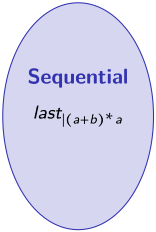


Classes of functions

Sequential functions

Deterministic underlying automaton

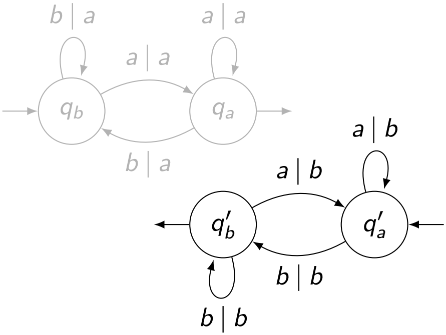




Classes of functions

Sequential functions

Deterministic underlying automaton



Sequential

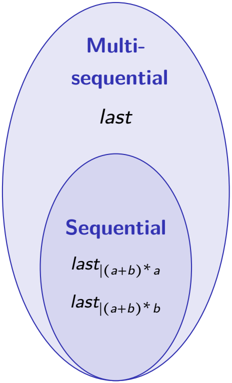
$last_{|(a+b)^*a}$

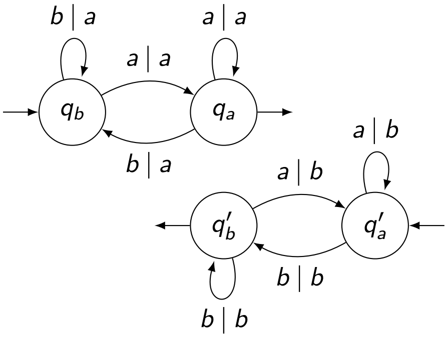
$last_{|(a+b)^*b}$

Classes of functions

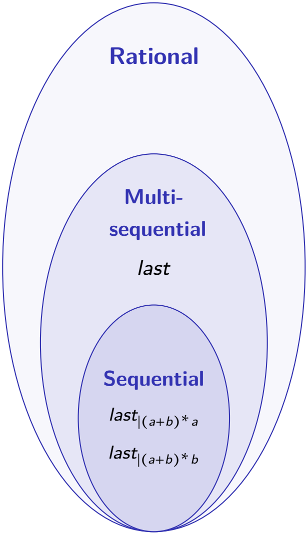
Multi-sequential functions

Finite union of sequential transducers





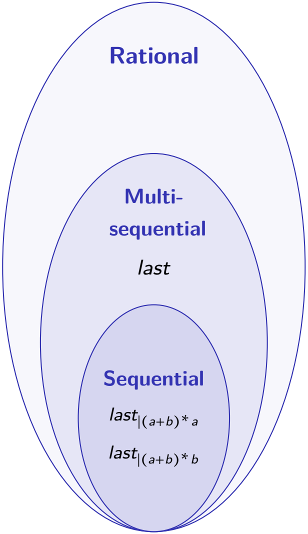
Classes of functions



Rational functions

Any finite-state transducer

Classes of functions



Rational functions

Any finite-state transducer

$$last^* :$$

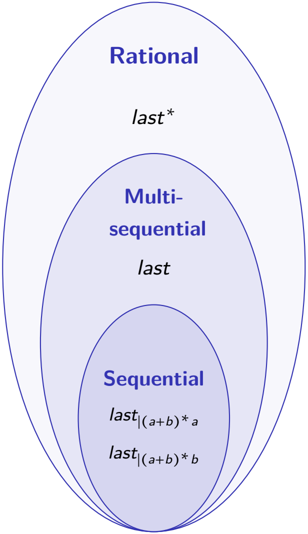
$$w_1 \# w_2 \# \dots \# w_n$$

$$\downarrow$$

$$last(w_1) \# last(w_2) \# \dots \# last(w_n)$$

any $n \in \mathbb{N}^*$

Classes of functions



Rational functions

Any finite-state transducer

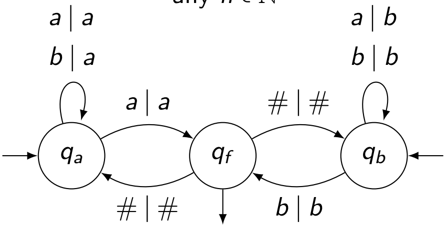
$$last^* :$$

$$w_1 \# w_2 \# \dots \# w_n$$

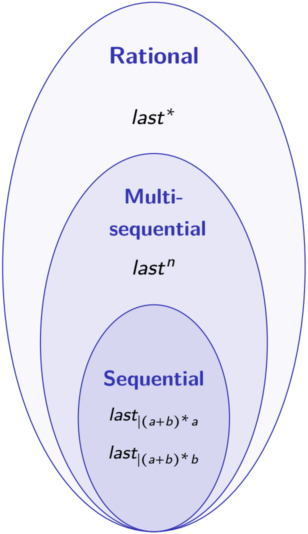
$$\downarrow$$

$$last(w_1) \# last(w_2) \# \dots \# last(w_n)$$

any $n \in \mathbb{N}^*$



Classes of functions



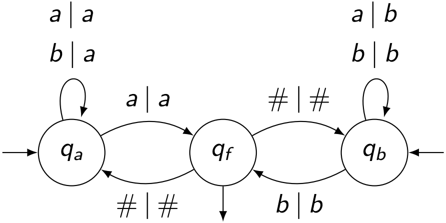
Rational functions

Any finite-state transducer

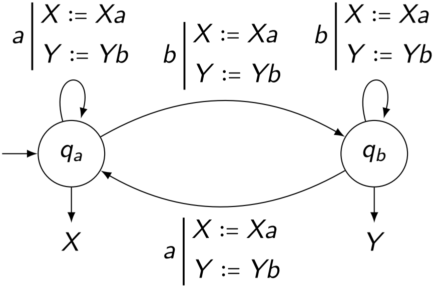
$last^n$:

$$w_1 \# w_2 \# \dots \# w_n$$
$$\downarrow$$
$$last(w_1) \# last(w_2) \# \dots \# last(w_n)$$

fixed $n \in \mathbb{N}^*$



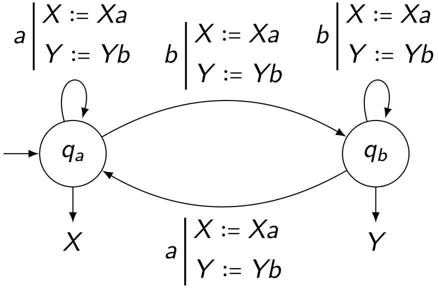
Streaming String Transducers[1] (SST)



SST realizing *last*

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

Streaming String Transducers[1] (SST)

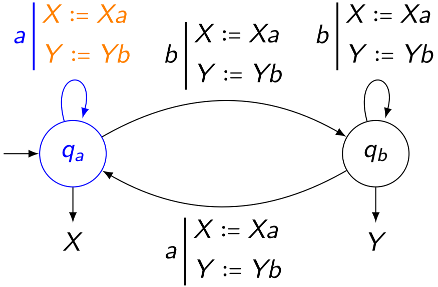


SST realizing *last*

$$\begin{cases} X = \epsilon \\ Y = \epsilon \end{cases} \qquad aba \mapsto$$

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

Streaming String Transducers[1] (SST)

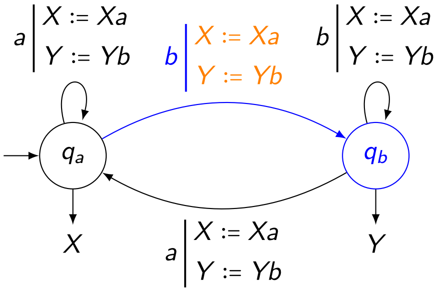


SST realizing *last*

$$\begin{cases} X = a \\ Y = b \end{cases} \qquad aba \mapsto$$

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

Streaming String Transducers[1] (SST)

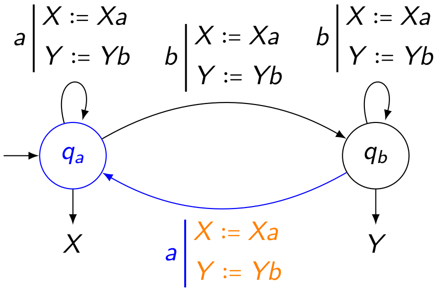


SST realizing *last*

$$\begin{cases} X = aa \\ Y = bb \end{cases} \qquad aba \mapsto$$

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

Streaming String Transducers[1] (SST)

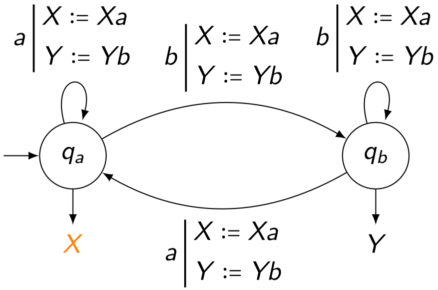


SST realizing *last*

$$\begin{cases} X = aaa \\ Y = bbb \end{cases} \qquad aba \mapsto$$

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

Streaming String Transducers[1] (SST)

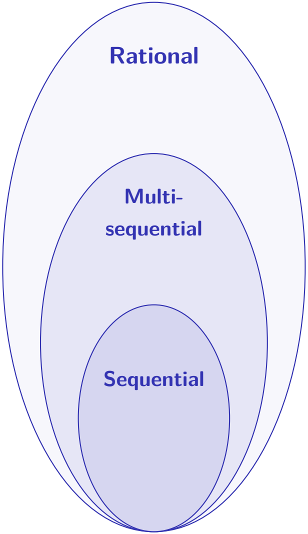


SST realizing *last*

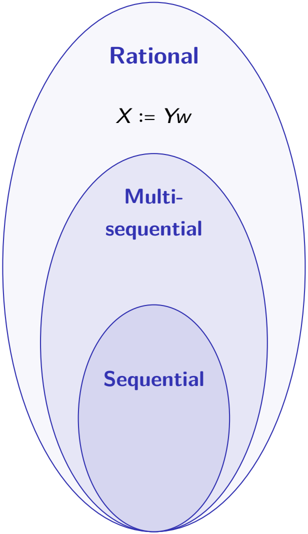
$$\begin{cases} X = aaa \\ Y = bbb \end{cases} \qquad aba \mapsto aaa$$

[1] Alur, R., & Cerný, P. (2011). Streaming transducers for algorithmic verification of single-pass list-processing programs. *POPL 2011*.

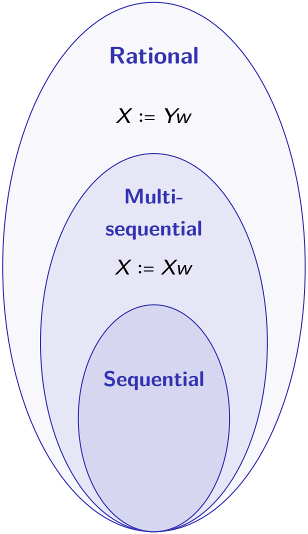
Classes of SST



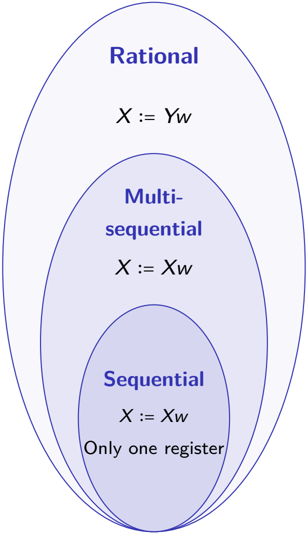
Classes of SST



Classes of SST



Classes of SST



Register minimization

Problem

Input : A rational function f realized by an SST $\mathcal{T}$, an integer k

Question : $\exists$? an SST $\mathcal{T}'$, with k registers, that is equivalent to $\mathcal{T}$ (i.e. realizing f).

Aix*Marseille université
Socialement engagée

Register minimization

Problem

Input : A rational function f realized by an SST $\mathcal{T}$, an integer k

Question : $\exists$? an SST $\mathcal{T}'$, with k registers, that is equivalent to $\mathcal{T}$ (i.e. realizing f).

For rational functions :

- Daviaud, L., Reynier, P., & Talbot, J. (2016).A generalised twinning property for minimisation of cost register automata. *LICS 2016*

For multi-sequential functions :

- Daviaud, L., Jecker, I., Reynier, P., & Villevalois, D. (2017).Degree of sequentiality of weighted automata. *FOSSACS 2017*

Algebraic characterization

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.

Definition

An equivalence relation $\sim$ over Σ^* is a right *congruence* if

$$\forall u, v \in \Sigma^* \; \forall \sigma \in \Sigma \qquad u \sim v \Rightarrow u\sigma \sim v\sigma$$

Definition

(right) *syntactic congruence* of L:

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$$

Algebraic characterization

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.

Definition

(right) *syntactic congruence* of L:

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$$

Theorem (Myhill-Nerode)

L is a regular language iff $\sim_L$ has a finite index (i.e. $|\Sigma^* / \sim_L| < \infty$)

Algebraic characterization

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.

Definition

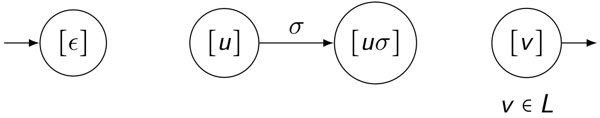
(right) *syntactic congruence* of L:

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$$

Theorem (Myhill-Nerode)

L is a regular language iff $\sim_L$ has a finite index (i.e. $|\Sigma^*/\sim_L| < \infty$)

- Allows to define the minimal automaton of L



Algebraic characterization

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.

Definition

(right) *syntactic congruence* of L:

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$$

Theorem (Myhill-Nerode)

L is a regular language iff $\sim_L$ has a finite index (i.e. $|\Sigma^*/\sim_L| < \infty$)

- Allows to define the minimal automaton of L
- Allows to decide if a language belongs to a subclass (ex: L is definable in first-order logic iff $\sim_L$ is aperiodic)

Algebraic characterization

Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language.

Definition

(right) *syntactic congruence* of L:

$$u \sim_L v \Leftrightarrow \forall w \in \Sigma^*, uw \in L \Leftrightarrow vw \in L$$

Theorem (Myhill-Nerode)

L is a regular language iff $\sim_L$ has a finite index (i.e. $|\Sigma^*/\sim_L| < \infty$)

- Allows to define the minimal automaton of L
- Allows to decide if a language belongs to a subclass (ex: L is definable in first-order logic iff $\sim_L$ is aperiodic)
- Used in learning algorithms (Angluin, 1987)
- ...

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} \end{cases}$$

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} u \sim_{dom(f)} v \\ \end{cases}$$

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, uw \in dom(f) \Leftrightarrow vw \in dom(f) \\ \end{cases}$$

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, uw \in dom(f) \Leftrightarrow vw \in dom(f) \\ \forall w \in u^{-1}dom(f),\ \widehat{f}(u)^{-1} f(uw) = \widehat{f}(v)^{-1} f(vw) \end{cases}$$

where $\widehat{f}(u) = \bigwedge f(uw)$ and $\bigwedge$ is the longest common prefix

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, uw \in dom(f) \Leftrightarrow vw \in dom(f) \\ \forall w \in u^{-1}dom(f),\ \widehat{f}(u)^{-1} f(uw) = \widehat{f}(v)^{-1} f(vw) \end{cases}$$

where $\widehat{f}(u) = \bigwedge f(uw)$ and $\bigwedge$ is the longest common prefix

Theorem (Choffrut[1])

f is a sequential function iff $\sim_f$ has a finite index

[1] Choffrut, C. (1977). Une caracterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationnelles. *Theor. Comput. Sci.*

Sequential functions : Syntactic congruence

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Syntactic congruence of f:

$$u \sim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, uw \in dom(f) \Leftrightarrow vw \in dom(f) \\ \forall w \in u^{-1}dom(f),\ \widehat{f}(u)^{-1} f(uw) = \widehat{f}(v)^{-1} f(vw) \end{cases}$$

where $\widehat{f}(u) = \bigwedge f(uw)$ and $\bigwedge$ is the longest common prefix

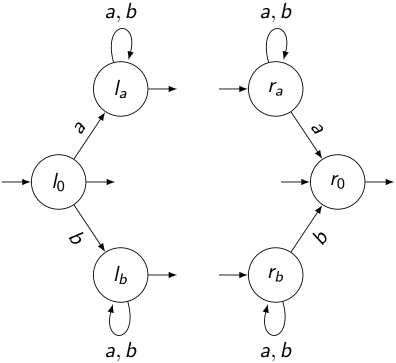
Theorem (Choffrut[1])

f is a sequential function iff $\sim_f$ has a finite index

- Allows to define the minimal sequential transducer of f

[1] Choffrut, C. (1977).Une caracterisation des fonctions sequentielles et des fonctions sous-sequentielles en tant que relations rationnelles. *Theor. Comput. Sci.*

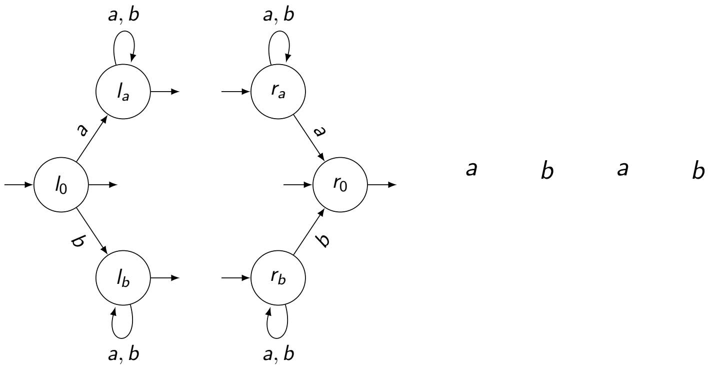
Rational functions : Bimachines



$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

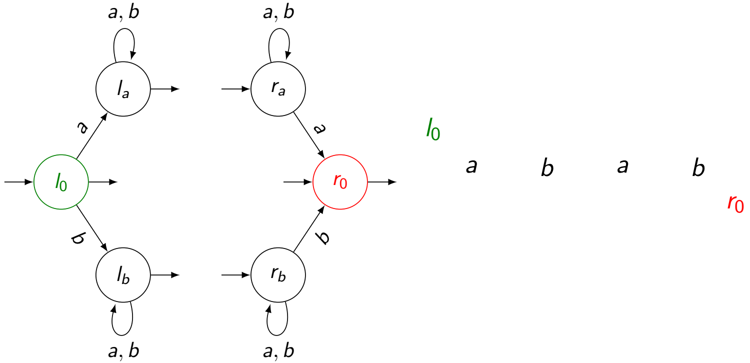
Rational functions : Bimachines



$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

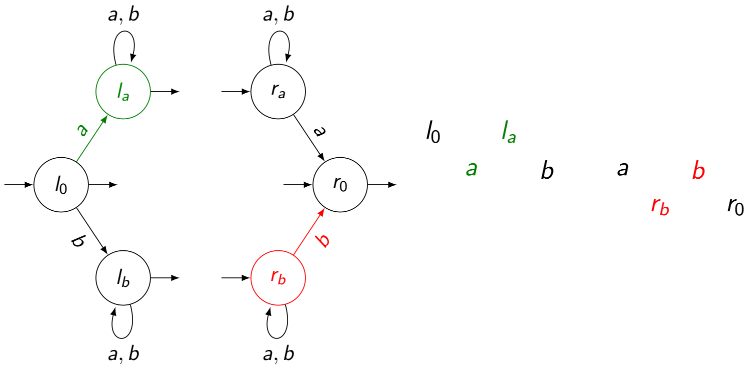
Rational functions : Bimachines



$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

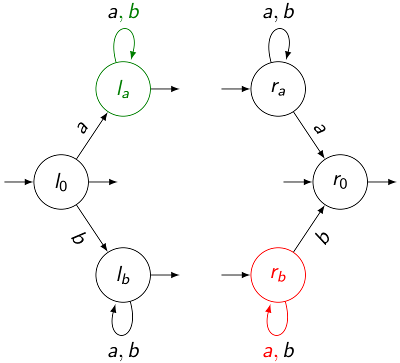
Rational functions : Bimachines



$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

Rational functions : Bimachines



l_0 l_a l_a

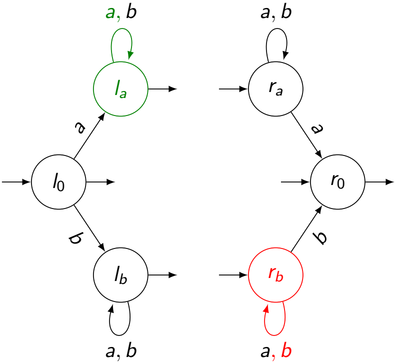
a b a b

r_b r_b r_0

$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

Rational functions : Bimachines

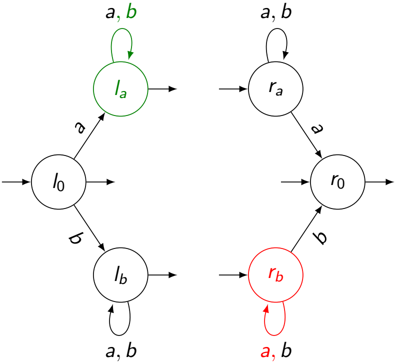


l_0		l_a		l_a		l_a		
	a		b		a		b	
		r_b		r_b		r_b		r_0

$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$
$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap* ($swap(\sigma w \tau) = \tau w \sigma$)

Rational functions : Bimachines



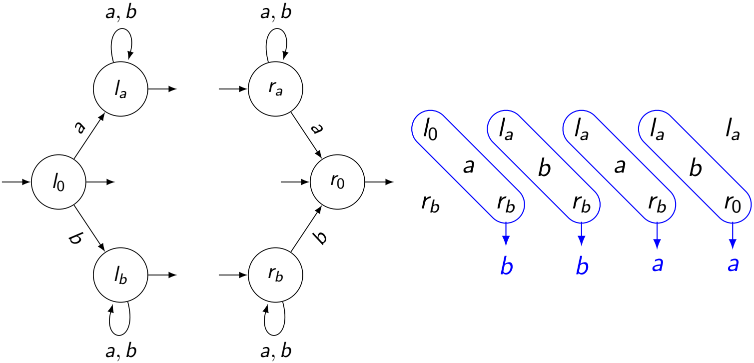
l_0		l_a		l_a		l_a		l_a
	a		b		a		b	
r_b		r_b		r_b		r_b		r_0

$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$

$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

Rational functions : Bimachines



$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$

$\omega(l, \sigma, r) = \sigma$ for any other (l, σ, r)

Bimachine realizing *swap*
$(swap(\sigma w \tau) = \tau w \sigma)$

Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Left syntactic congruence of f:

$$u \leftsquigarrow_f v \Leftrightarrow \left\{ \right.$$

Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Left syntactic congruence of f:

$$u \backsim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, wu \in dom(f) \Leftrightarrow wv \in dom(f) \\ \\ \end{cases}$$

Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Left syntactic congruence of f:

$$u \backsim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, wu \in dom(f) \Leftrightarrow wv \in dom(f) \\ \sup\limits_{w \in dom(f)u^{-1}} d_p(f(wu), f(wv)) < \infty \end{cases}$$

where $d_p(u, v) = |u| + |v| - 2|lcp(u, v)|$ is the prefix distance

Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Left syntactic congruence of f:

$$u \backsim_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, wu \in dom(f) \Leftrightarrow wv \in dom(f) \\ \sup\limits_{w \in dom(f)u^{-1}} d_p(f(wu), f(wv)) < \infty \end{cases}$$

where $d_p(u,v) = |u| + |v| - 2|lcp(u,v)|$ is the prefix distance

Theorem (Reutenauer and Schützenberger[1])

If f is a rational function then $\backsim_f$ has a finite index

[1] Reutenauer, C., & Schützenberger, M. P. (1991). Minimization of rational word functions. *SIAM J. Comput.*

Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Left syntactic congruence of f:

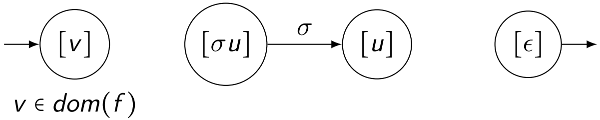
$$u \leftharpoondown_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, wu \in dom(f) \Leftrightarrow wv \in dom(f) \\ \sup\limits_{w \in dom(f)u^{-1}} d_p(f(wu), f(wv)) < \infty \end{cases}$$

where $d_p(u,v) = |u| + |v| - 2|lcp(u,v)|$ is the prefix distance

Theorem (Reutenauer and Schützenberger)

If f is a rational function then $\leftharpoondown_f$ has a finite index

- Allows to define the minimal right automaton of f



Rational functions : Bimachines

Let Σ be a finite alphabet and $f : \Sigma^* \to \Sigma^*$ be a function.

Definition

Right syntactic congruence of f:

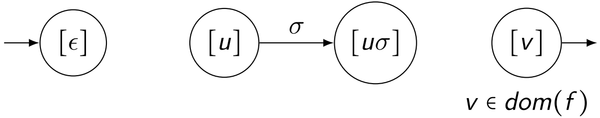
$$u \rightsquigarrow_f v \Leftrightarrow \begin{cases} \forall w \in \Sigma^*, uw \in dom(f) \Leftrightarrow vw \in dom(f) \\ \sup\limits_{w \in u^{-1}dom(f)} d_s(f(uw), f(vw)) < \infty \end{cases}$$

where $d_s(u,v) = |u| + |v| - 2|lcs(u,v)|$ is the suffix distance

Theorem (Reutenauer and Schützenberger)

If f is a rational function then $\rightsquigarrow_f$ has a finite index

- Allows to define the minimal left automaton of f



Register minimization

Known results

For rational functions :

- Daviaud, L., Reynier, P., & Talbot, J. (2016).A generalised twinning property for minimisation of cost register automata. *LICS 2016*
- Daviaud, L., Jecker, I., Reynier, P., & Villevalois, D. (2017).Degree of sequentiality of weighted automata. *FOSSACS 2017*

Register minimization

Known results

For rational functions :

- Daviaud, L., Reynier, P., & Talbot, J. (2016).A generalised twinning property for minimisation of cost register automata. *LICS 2016*
- Daviaud, L., Jecker, I., Reynier, P., & Villevalois, D. (2017).Degree of sequentiality of weighted automata. *FOSSACS 2017*

Based on finding patterns in the transducer realizing the function

Register minimization

Known results

For rational functions :

- Daviaud, L., Reynier, P., & Talbot, J. (2016).A generalised twinning property for minimisation of cost register automata. *LICS 2016*
- Daviaud, L., Jecker, I., Reynier, P., & Villevalois, D. (2017).Degree of sequentiality of weighted automata. *FOSSACS 2017*

Based on finding patterns in the transducer realizing the function

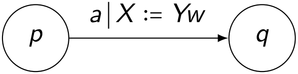
Objective : Solving the register minimization problem for rational functions using their algebraic characterization

Register flows

If $X := Yw$ we say that the register Y **flows** to X

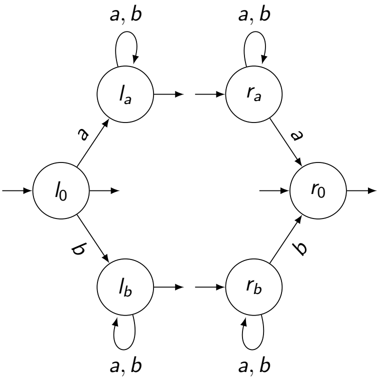
Independent flows

If



then Y depends only on X and a

Bimachine ≃ SST with independent flows

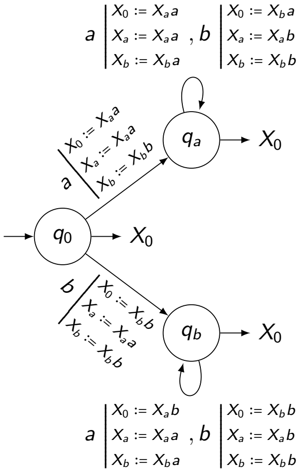


$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$

$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

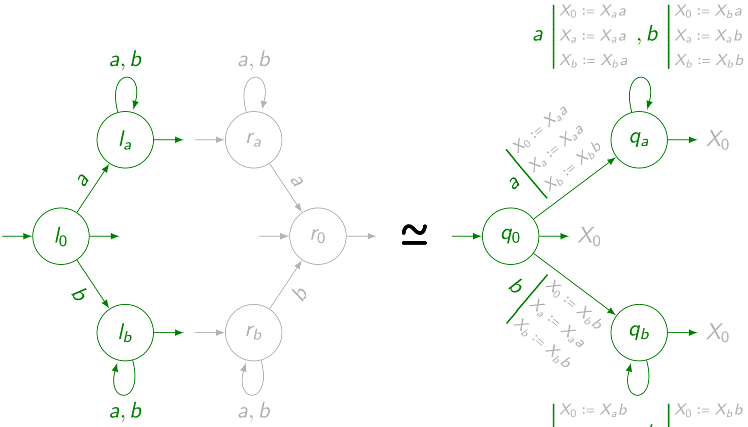
Bimachine

$\simeq$



SST

Bimachine ≃ SST with independent flows

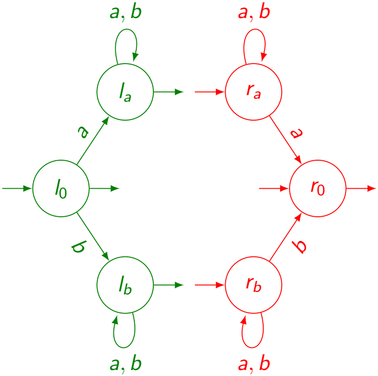


$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$

$\omega(l, \sigma, r) = \sigma$ for any other (l, σ, r)

Bimachine **SST**

Bimachine ≃ SST with independent flows

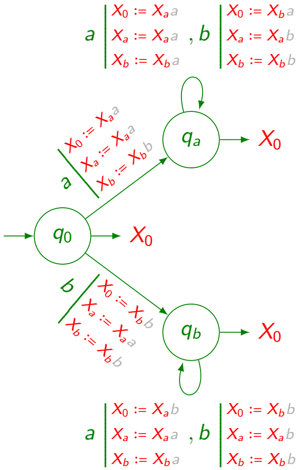


$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$

$\omega(l, \sigma, r) = \sigma$ for any other (l, σ, r)

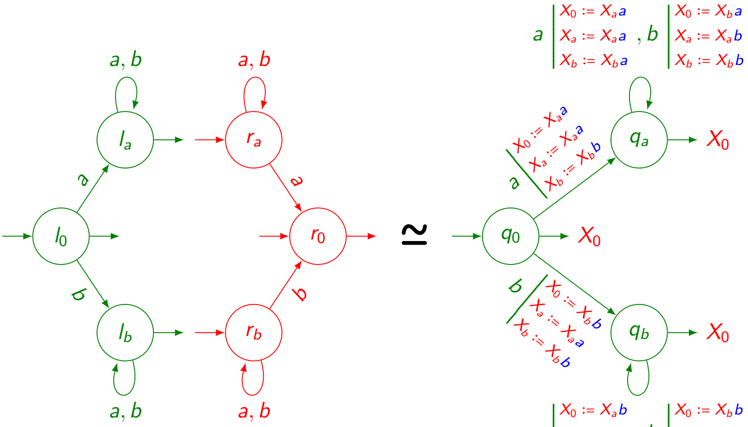
Bimachine

$\simeq$



SST

Bimachine ≃ SST with independent flows



$$\omega(l_0, \sigma, r_\tau) = \omega(l_\tau, \sigma, r_0) = \tau$$

$$\omega(l, \sigma, r) = \sigma \text{ for any other } (l, \sigma, r)$$

Bimachine **SST**

Bimachine minimization

Filiot, E., Gauwin, O., & Lhote, N. (2019). Logical and algebraic characterizations of rational transductions. *Log. Methods Comput. Sci.*

- Bimachines can be minimized in PTIME
- Left / right minimization → Tradeoff between the size of the left and the right automaton

Bimachine minimization

Filiot, E., Gauwin, O., & Lhote, N. (2019). Logical and algebraic characterizations of rational transductions. *Log. Methods Comput. Sci.*

- Bimachines can be minimized in PTIME
- Left / right minimization → Tradeoff between the size of the left and the right automaton

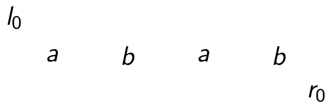
Corollary

For SST with independent flows :

⇒ Minimal number of registers = Number of states of the minimal right automaton

⇒ Tradeoff between the size of the underlying automaton and the number of registers

Asynchronous bimachines



Execution of a bimachine realizing *swap*

Asynchronous bimachines

l_0 l_a

a b a b

r_b r_0

Execution of a bimachine realizing *swap*

Asynchronous bimachines

l_0 l_a l_a

a b a b

r_b r_b r_0

Execution of a bimachine realizing *swap*

Aix*Marseille université Socialement engagée

Asynchronous bimachines

$$\begin{array}{ccccccccc} l_0 & & l_a & & l_a & & l_a & & \\ & a & & b & & a & & b & \\ & & r_b & & r_b & & r_b & & r_0 \end{array}$$

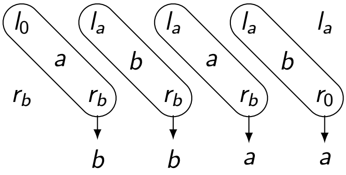
Execution of a bimachine realizing *swap*

Asynchronous bimachines

l_0		l_a		l_a		l_a		l_a
	a		b		a		b	
r_b		r_b		r_b		r_b		r_0

Execution of a bimachine realizing *swap*

Asynchronous bimachines



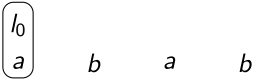
Execution of a bimachine realizing ***swap***

Asynchronous bimachines

a b a b

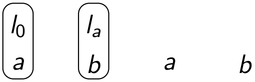
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines

Yahia Idriss BENALIOUA

l_0	l_a	l_a	
a	b	a	b

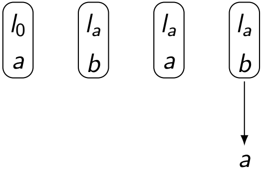
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines

$\begin{pmatrix} l_0 \\ a \end{pmatrix}$ $\begin{pmatrix} l_a \\ b \end{pmatrix}$ $\begin{pmatrix} l_a \\ a \end{pmatrix}$ $\begin{pmatrix} l_a \\ b \end{pmatrix}$

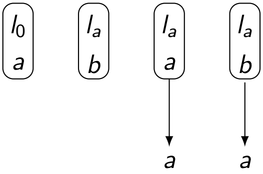
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



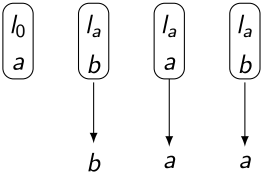
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



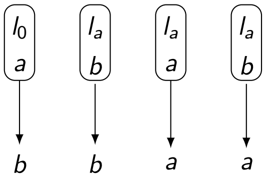
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



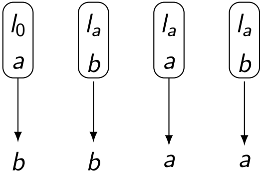
Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines



Execution of an asynchronous bimachine realizing *swap*

Asynchronous bimachines ≃ SST

Minimization of asynchronous bimachines

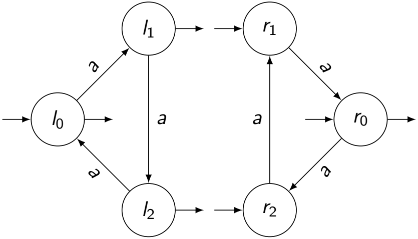
(ongoing work)

$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$

Minimization of asynchronous bimachines

(ongoing work)

$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$

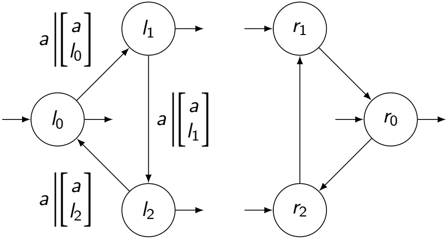


ω	l_0	l_1	l_2
r_0	b	b	a
r_1	b	a	b
r_2	a	b	b

Minimization of asynchronous bimachines

(ongoing work)

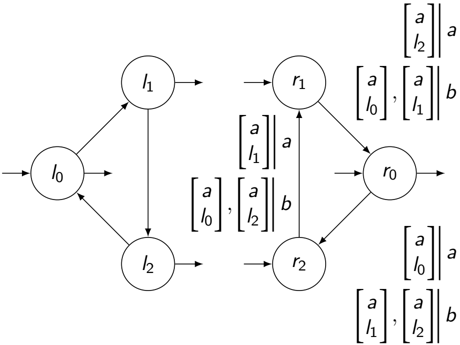
$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$



Minimization of asynchronous bimachines

(ongoing work)

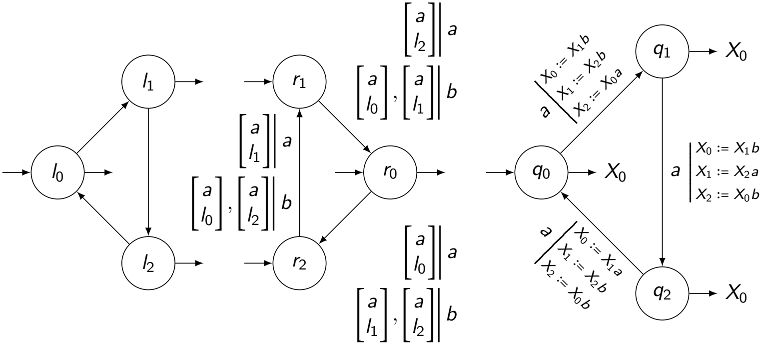
$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$



Minimization of asynchronous bimachines

(ongoing work)

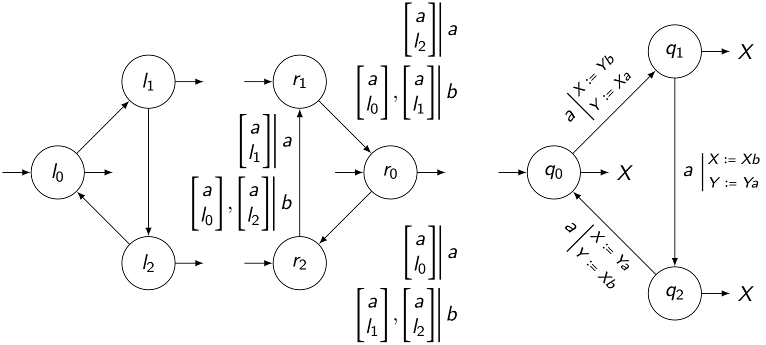
$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$



Minimization of asynchronous bimachines

(ongoing work)

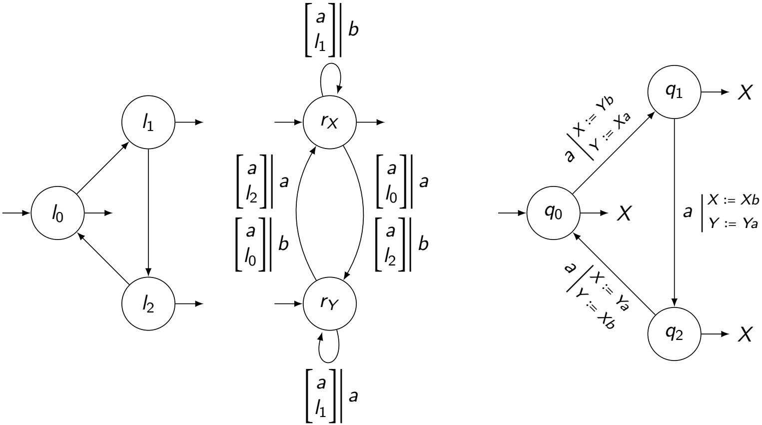
$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$



Minimization of asynchronous bimachines

(ongoing work)

$$a^n \mapsto \begin{cases} a^n & \text{if } n \equiv 0 \mod 3 \\ b^n & \text{otherwise} \end{cases}$$



Minimization of asynchronous bimachines

(ongoing work)

Problem

Input : A partial sequential function f, an integer k

Question : $\exists$? a sequential transducer, with k states, realizing an extension of f.

Minimization of asynchronous bimachines

(ongoing work)

Problem

Input : A partial sequential function f, an integer k

Question : $\exists$? a sequential transducer, with k states, realizing an extension of f.

- NP-complete for letter-to-letter transducers

Pfleeger, C. P. (1973). State reduction in incompletely specified finite-state machines. *IEEE Trans. Computers*

Conclusion

Principal results:

- Bimachine ≃ SST with independent flows → register minimization for this class
- Asynchronous bimachine ≃ SST

Conclusion

Principal results:

- Bimachine ≃ SST with independent flows → register minimization for this class
- Asynchronous bimachine ≃ SST

Perspectives:

- Minimization of asynchronous bimachines
- Links with other problems

Thank you for your attention