On the hierarchy of block deterministic languages

Pascal CARON, Ludovic MIGNOT and Clément MIKLARZ

Université de Rouen
Laboratoire d'Informatique, du Traitement de l'Information et des Systèmes

August 20, 2015

History

- 1986: the language SGML is published. Its structure is specified by a Document Type Definition (DTD) using regular expressions where each next position can be decided without look-ahead.
  Examples:
  - <!ELEMENT x $((a|b)^*, c)$ > $\Longrightarrow$ $(a+b)^*c$: OK
  - <!ELEMENT x $((a|b)^*, a)$ > $\Longrightarrow$ $(a+b)^*a$: KO

History

- 1986: the language SGML is published. Its structure is specified by a Document Type Definition (DTD) using regular expressions where each next position can be decided without look-ahead.
  Examples:
  - <!ELEMENT x $((a|b)^*, c)$ > $\Longrightarrow$ $(a+b)^*c$: OK
  - <!ELEMENT x $((a|b)^*, a)$ > $\Longrightarrow$ $(a+b)^*a$: KO
- 1998: Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages

History

- 1986: the language SGML is published. Its structure is specified by a Document Type Definition (DTD) using regular expressions where each next position can be decided without look-ahead.
  Examples:
  - <!ELEMENT x $((a|b)^*, c)$ > $\Longrightarrow$ $(a+b)^*c$: OK
  - <!ELEMENT x $((a|b)^*, a)$ > $\Longrightarrow$ $(a+b)^*a$: KO
- 1998: Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages
- 2001: Giammarresi, D., Montalbano, R., Wood, D.: Block-deterministic regular languages

History

- 1986: the language SGML is published. Its structure is specified by a Document Type Definition (DTD) using regular expressions where each next position can be decided without look-ahead.
  Examples:
  - <!ELEMENT x $((a|b)^*, c)$ > $\Longrightarrow$ $(a+b)^*c$: OK
  - <!ELEMENT x $((a|b)^*, a)$ > $\Longrightarrow$ $(a+b)^*a$: KO
- 1998: Brüggemann-Klein, A., Wood, D.: One-unambiguous regular languages
- 2001: Giammarresi, D., Montalbano, R., Wood, D.: Block-deterministic regular languages
- 2008: Han, Y.-S., Wood, D.: Generalizations of 1-deterministic regular languages

Overview

1. Preliminaries
   - Basics
   - 1-unambiguous regular languages
   - Block deterministic regular languages
2. Previous results on block deterministic regular languages
   - Determining whether a language L is block deterministic
   - The hierarchy of k-block deterministic regular languages
3. New results
   - Flawed results
   - A witness for the infinite hierarchy

Regular Languages

Regular expression

$$E = \emptyset \qquad E = \varepsilon \qquad E = a$$
$$E = (F + G) \qquad E = (F \cdot G) \qquad E = (F^*)$$

Language denoted by E

$$\mathrm{L}(\emptyset) = \emptyset \qquad \mathrm{L}(\varepsilon) = \{\varepsilon\} \qquad \mathrm{L}(a) = \{a\}$$
$$\mathrm{L}(F + G) = \mathrm{L}(F) \cup \mathrm{L}(G) \qquad \mathrm{L}(F \cdot G) = \mathrm{L}(F) \cdot \mathrm{L}(G) \qquad \mathrm{L}(F^*) = \mathrm{L}(F)^*$$

Regular Languages

Regular expression

$E = \emptyset$ $\quad$ $E = \varepsilon$ $\quad$ $E = a$

$E = (F + G)$ $\quad$ $E = (F \cdot G)$ $\quad$ $E = (F^*)$

Language denoted by E

$\mathrm{L}(\emptyset) = \emptyset$ $\quad$ $\mathrm{L}(\varepsilon) = \{\varepsilon\}$ $\quad$ $\mathrm{L}(a) = \{a\}$

$\mathrm{L}(F + G) = \mathrm{L}(F) \cup \mathrm{L}(G)$ $\quad$ $\mathrm{L}(F \cdot G) = \mathrm{L}(F) \cdot \mathrm{L}(G)$ $\quad$ $\mathrm{L}(F^*) = \mathrm{L}(F)^*$

Example

- $E = (a + bc)(d + e)$ with $\mathrm{L}(E) = \{ad, ae, bcd, bce\}$
- $F = (a + bc)^*$ with $\mathrm{L}(F) = \{\varepsilon, a, aa, bc, aaa, abc, bca, \ldots\}$

Automaton

Definition

An *automaton* A is a 5-tuple $(\Sigma, Q, I, F, \delta)$ with:

- Σ a finite alphabet
- Q a finite set of states
- $I \subset Q$ the set of initial states
- $F \subset Q$ the set of final states
- $\delta \subset Q \times \Sigma \times Q$ a set of transitions

Automaton

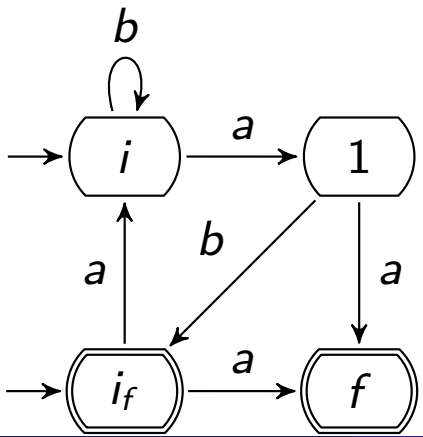
Definition

An *automaton* A is a 5-tuple $(\Sigma, Q, I, F, \delta)$ with:

- Σ a finite alphabet
- Q a finite set of states
- $I \subset Q$ the set of initial states
- $F \subset Q$ the set of final states
- $\delta \subset Q \times \Sigma \times Q$ a set of transitions

Language recognized

The language recognized by an automaton is the set of labels of all paths that lead from an initial state to a final state.



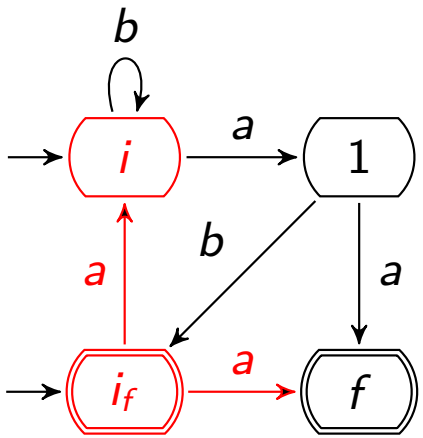
$$L = \{\varepsilon, a, aa, ab, \ldots\}$$

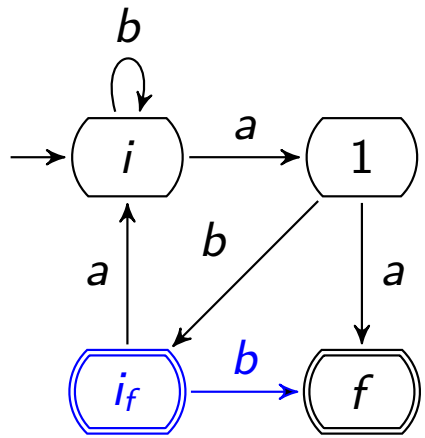
Deterministic automaton (DFA)

Definition

An automaton $A = (\Sigma, Q, I, F, \delta)$ is *deterministic* if:

- $|I| = 1$
- $\forall t_1 = (p, a, q_1), t_2 = (p, b, q_2) \in \delta, (t_1 \neq t_2) \Longrightarrow (a \neq b)$



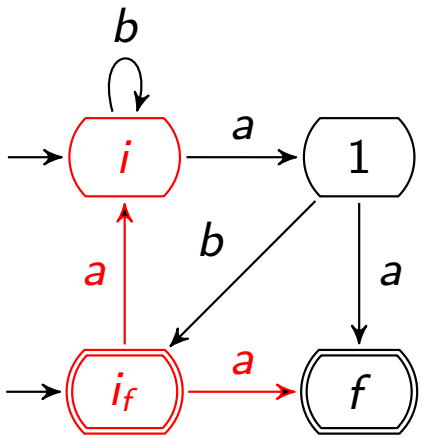


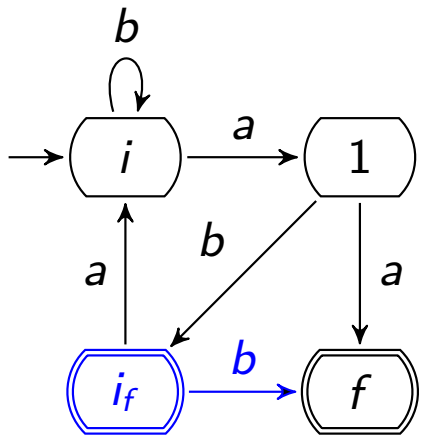
Deterministic automaton (DFA)

Definition

An automaton $A = (\Sigma, Q, I, F, \delta)$ is *deterministic* if:

- $|I| = 1$
- $\forall t_1 = (p, a, q_1), t_2 = (p, b, q_2) \in \delta, (t_1 \neq t_2) \Longrightarrow (a \neq b)$





Minimality

A DFA is minimal if there does not exist another DFA recognizing the same language with less states.

From a regular expression to its Glushkov automaton

$$E = (a + (bb)^*)ab^* + \varepsilon$$

Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 b_5^* + \varepsilon$$

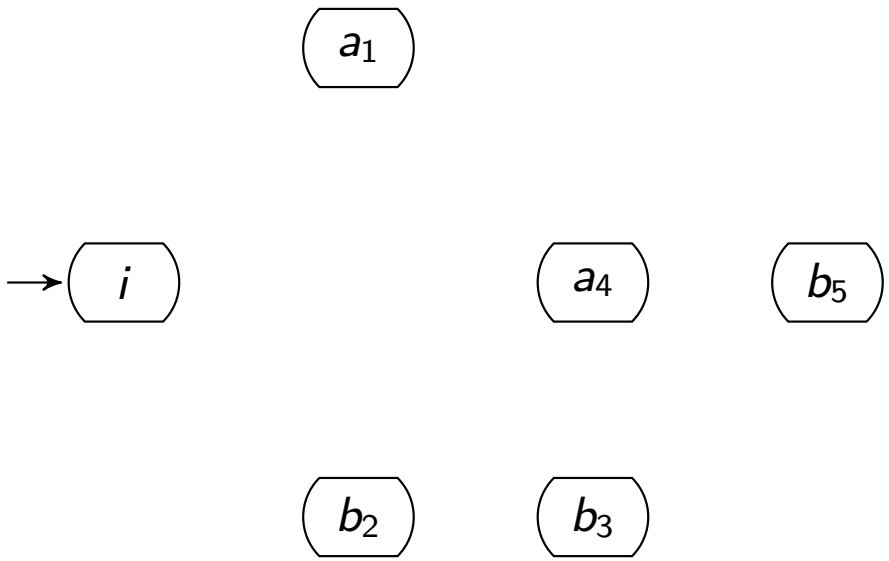


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 b_5^* + \varepsilon$$

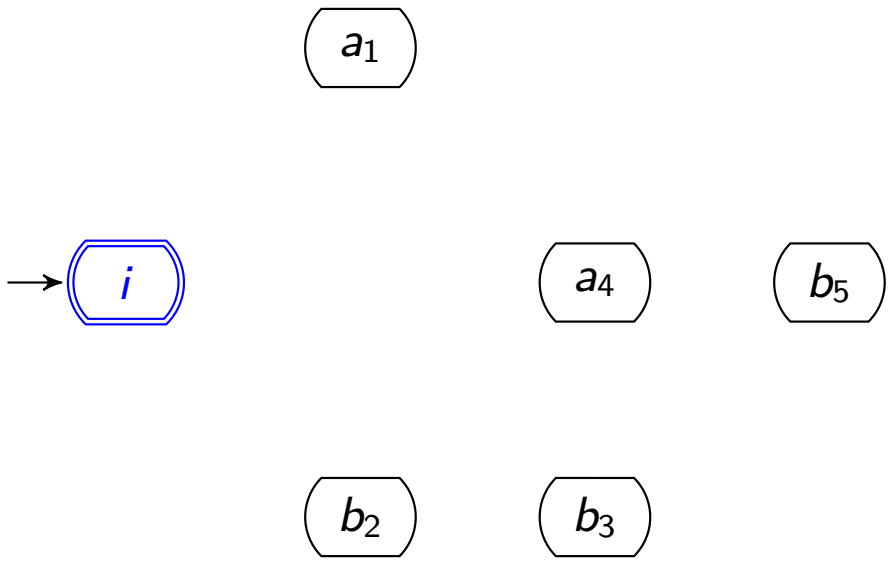


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 {b_5}^* + \varepsilon$$

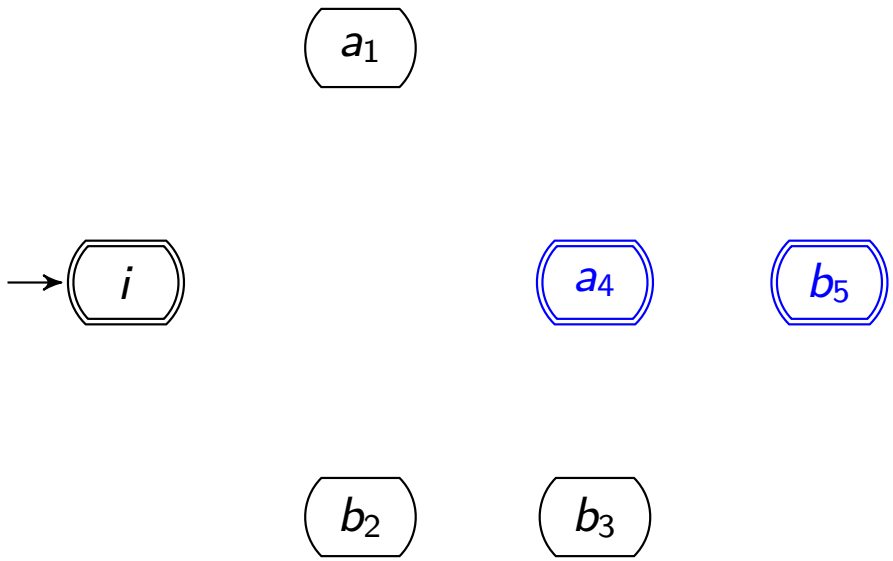


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 b_5^* + \varepsilon$$

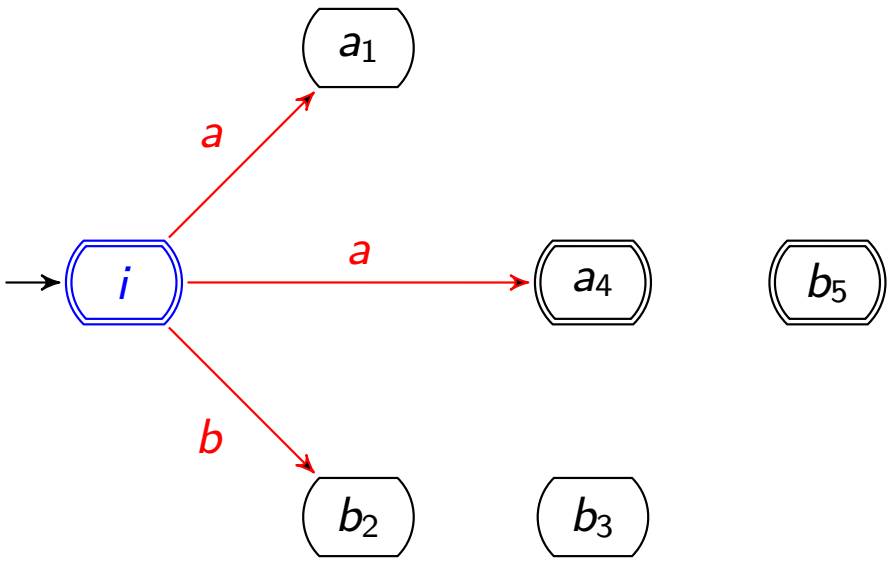


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 b_5^* + \varepsilon$$

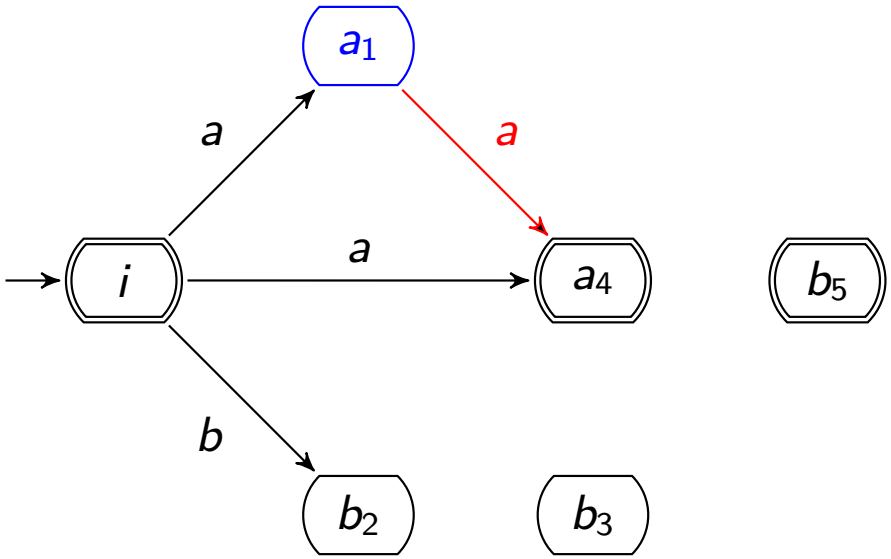


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2\, b_3)^*)\, a_4\, b_5^* + \varepsilon$$

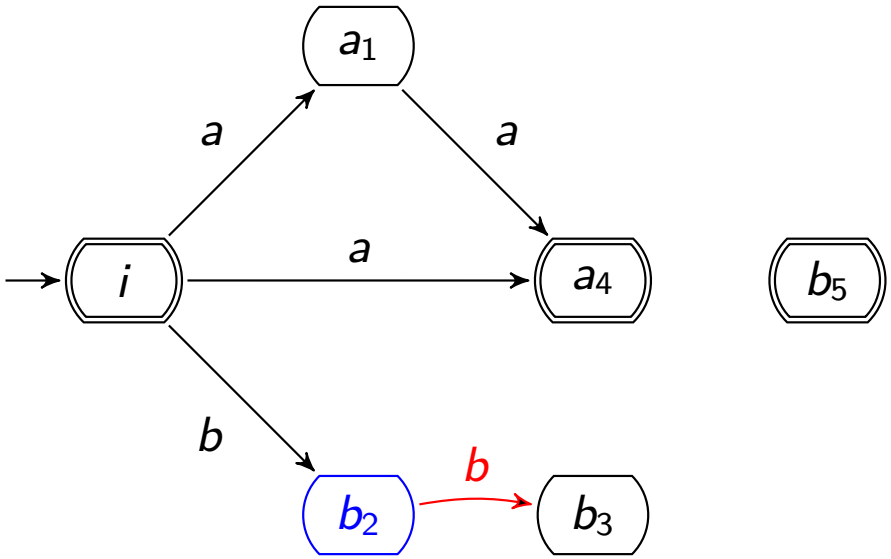


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 b_5^* + \varepsilon$$

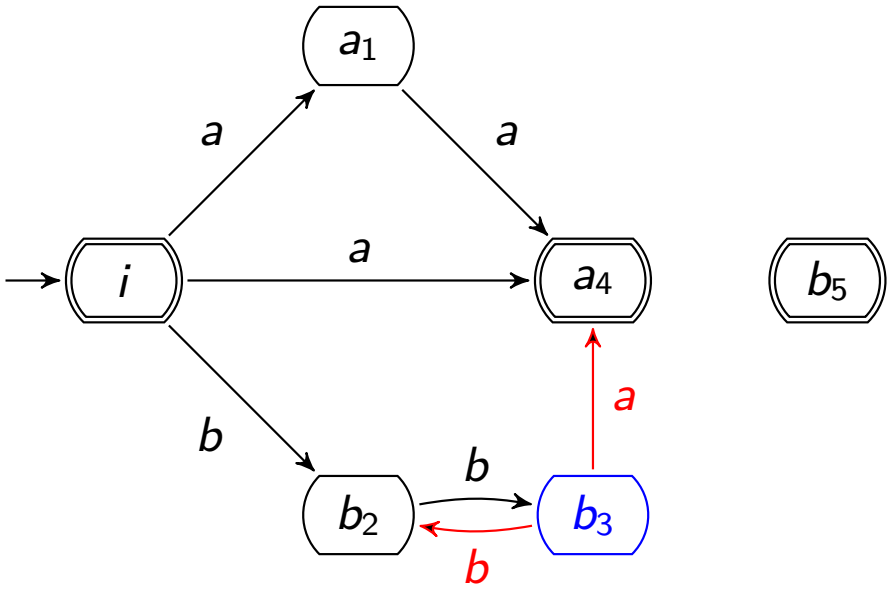


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 {b_5}^* + \varepsilon$$

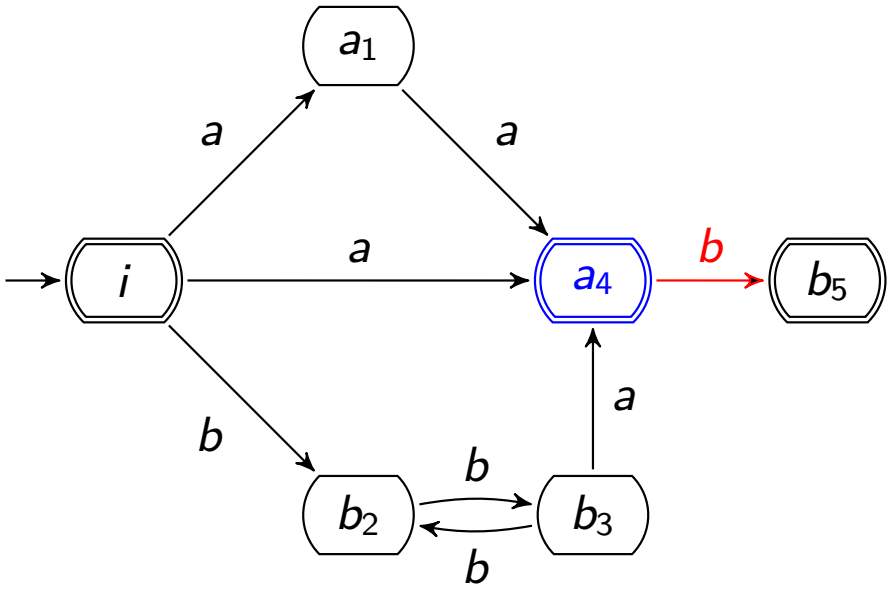


Figure: The Glushkov automaton of E

From a regular expression to its Glushkov automaton

$$E^\sharp = (a_1 + (b_2 b_3)^*) a_4 {b_5}^* + \varepsilon$$

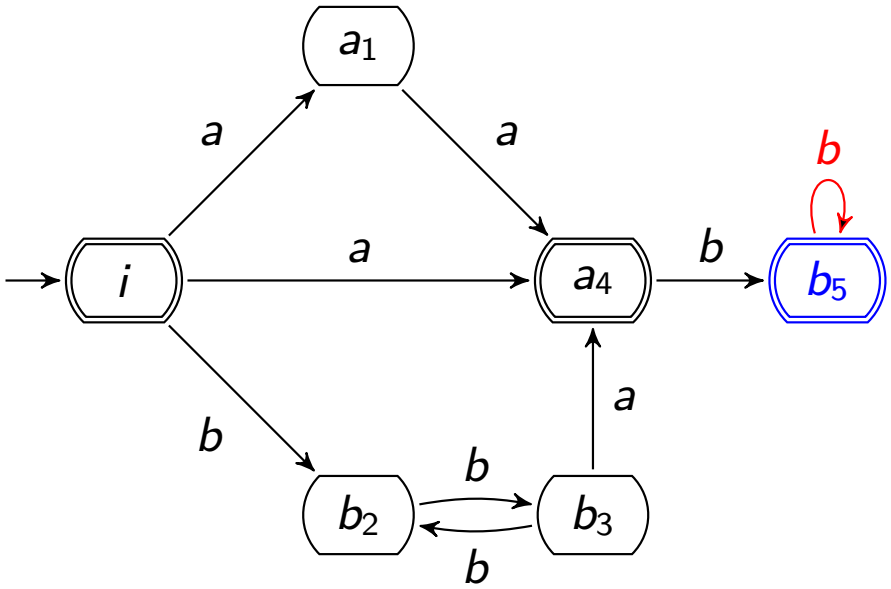


Figure: The Glushkov automaton of E

1-unambiguous regular expression

A regular expression E is 1-*unambiguous* if its Glushkov automaton is deterministic.

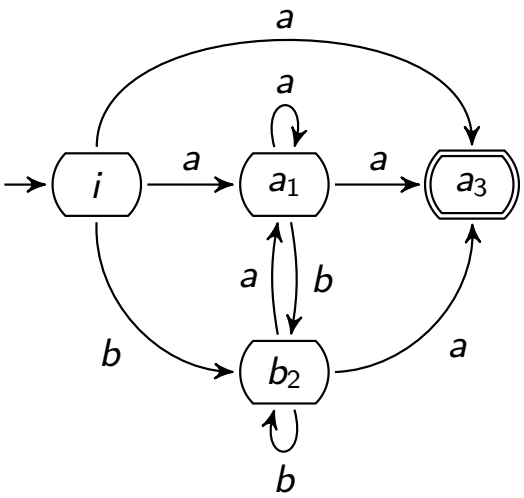


Figure: The Glushkov automaton of $E = (a + b)^* a$

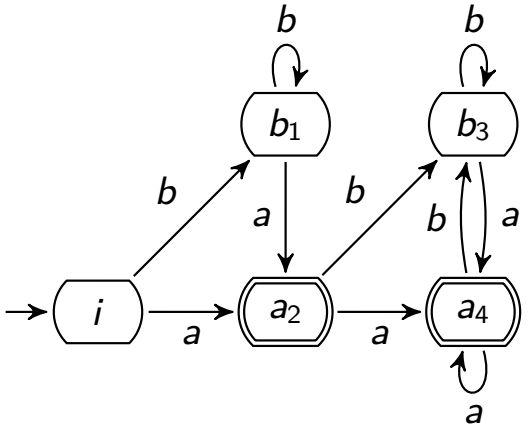


Figure: The Glushkov automaton of $F = b^* a (b^* a)^*$

1-unambiguous regular expression

A regular expression E is 1-*unambiguous* if its Glushkov automaton is deterministic.

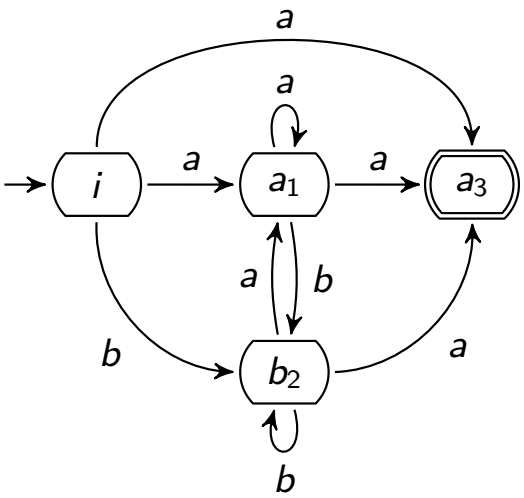


Figure: The Glushkov automaton of $E = (a + b)^* a$

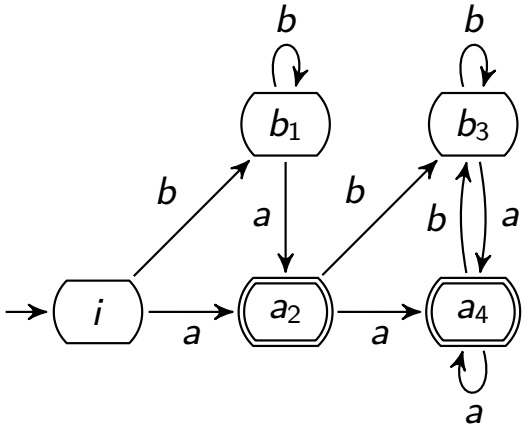
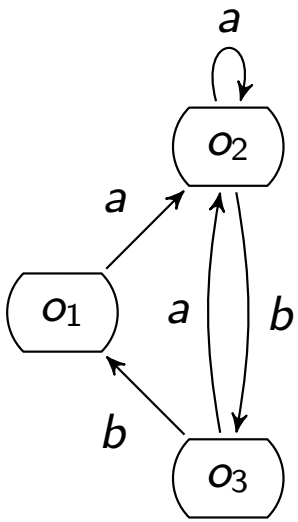


Figure: The Glushkov automaton of $F = b^* a(b^* a)^*$

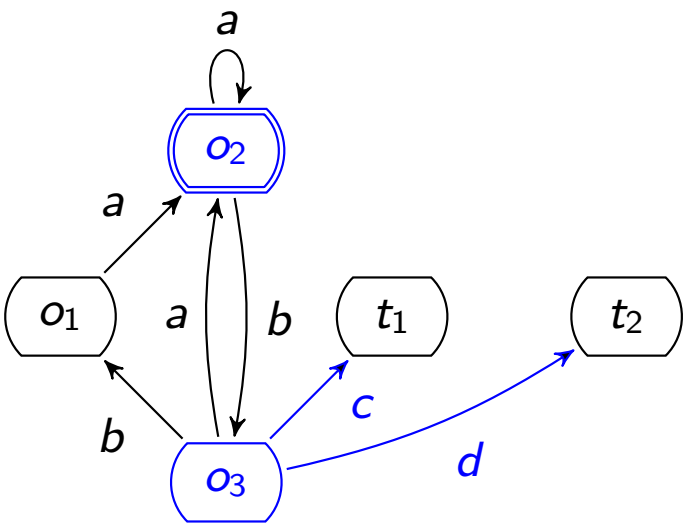
1-unambiguous regular language

A regular language is 1-*unambiguous* if it can be denoted by a 1-unambiguous regular expression.

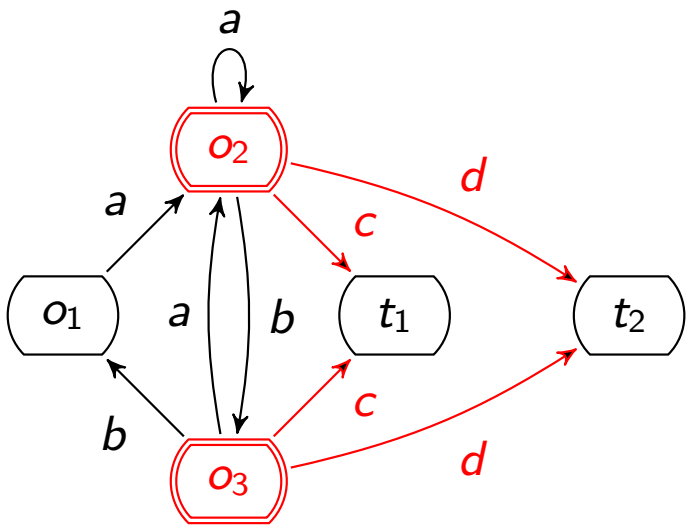
Orbit, gate and orbit property



Orbit, gate and orbit property



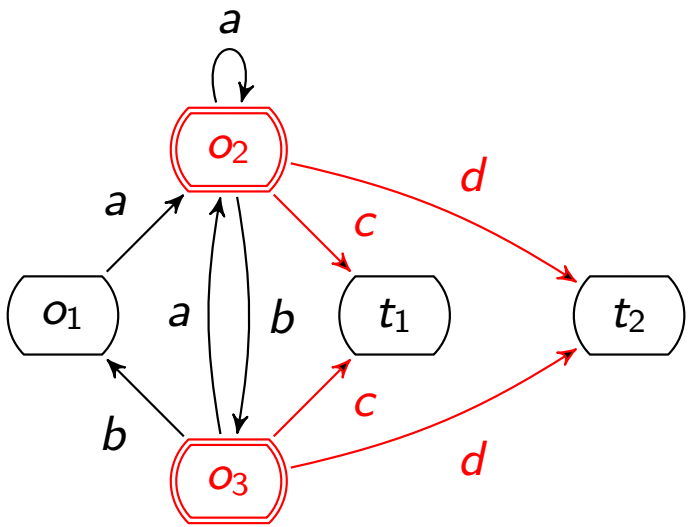
Orbit, gate and orbit property



Outerly transverse orbit

All the gates have the same finality and the same transitions to outside states.

Orbit, gate and orbit property



Outerly transverse orbit

All the gates have the same finality and the same transitions to outside states.

Orbit property

An automaton has the orbit property if all of its orbits are outerly transverse.

Determining whether a language L is 1-unambiguous

A necessary condition

If a language is 1-unambiguous, then its minimal DFA has the orbit property.

Determining whether a language L is 1-unambiguous

A necessary condition

If a language is 1-unambiguous, then its minimal DFA has the orbit property.

Brüggemann-Klein and Wood

There exist regular languages which are not 1-unambiguous

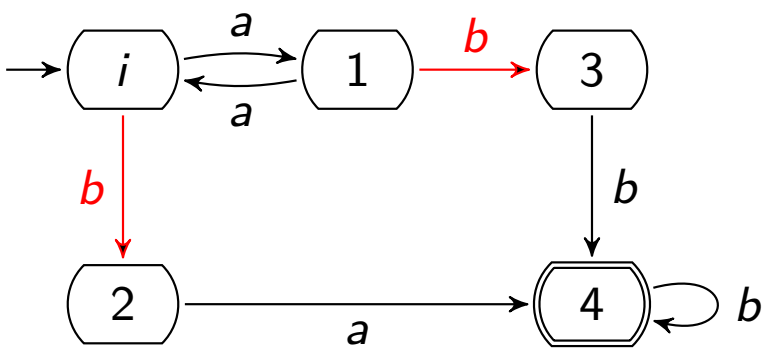


Figure: The minimal DFA recognizing $(aa)^*(abb + ba)b^*$

k-block regular expression and k-block automaton

- a block = a non-empty word
- k-block: when all blocks have a length at most k
- blocks can be considered as symbols in a different alphabet

k-block regular expression and k-block automaton

- a block = a non-empty word
- k-block: when all blocks have a length at most k
- blocks can be considered as symbols in a different alphabet

A 2-block regular expression

$[aa]^*([ab]b + ba)b^*$

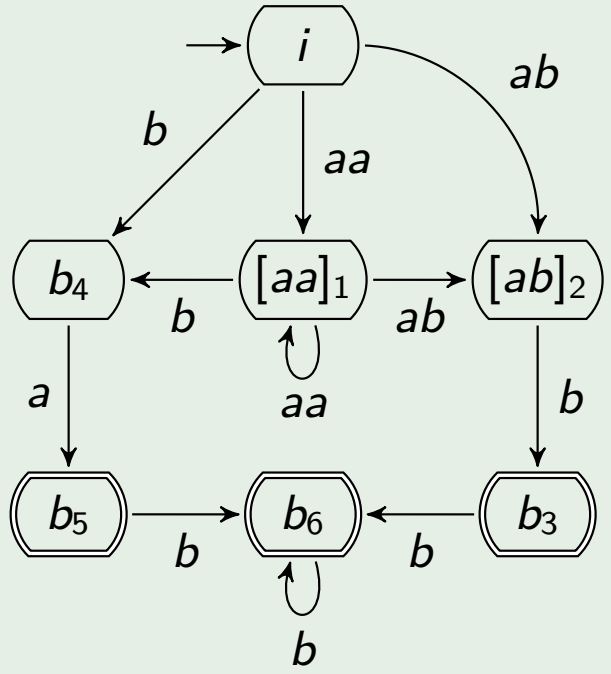
k-block regular expression and k-block automaton

- a block = a non-empty word
- k-block: when all blocks have a length at most k
- blocks can be considered as symbols in a different alphabet

A 2-block regular expression

$[aa]^*([ab]b + ba)b^*$

A 2-block automaton



Block deterministic automaton

Block determinism

A block automaton $B = (\Gamma, Q, I, F, \delta)$ is *block deterministic* if:

- it is deterministic

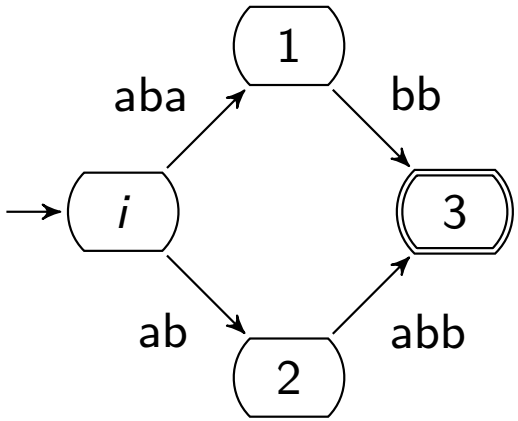


Figure: A non-block deterministic automaton

Block determinism

A block automaton $B = (\Gamma, Q, I, F, \delta)$ is *block deterministic* if:

- it is deterministic
- $\forall t_1 = (p, b_1, q_1), t_2 = (p, b_2, q_2) \in \delta, (t_1 \neq t_2) \Longrightarrow (b_1 \notin \mathrm{Pref}(b_2))$

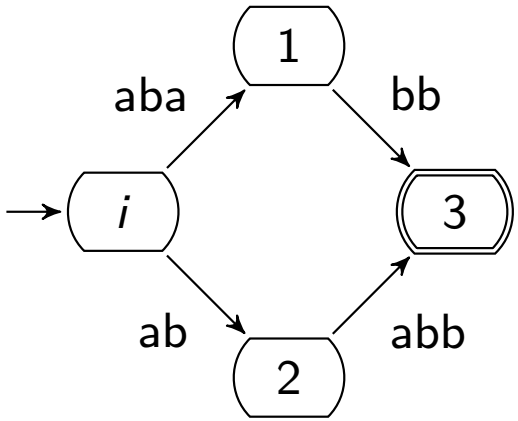


Figure: A non-block deterministic automaton

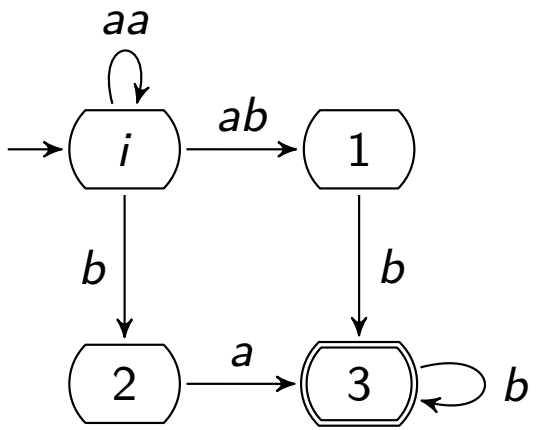


Figure: A 2-block deterministic automaton

k-block deterministic regular languages

Definitions

- A k-block regular expression E is k-block deterministic if its Glushkov automaton is k-block deterministic.
- A regular language is k-block deterministic if it can be denoted by a k-block deterministic regular expression.

Definitions

- A k-block regular expression E is k-block deterministic if its Glushkov automaton is k-block deterministic.
- A regular language is k-block deterministic if it can be denoted by a k-block deterministic regular expression.

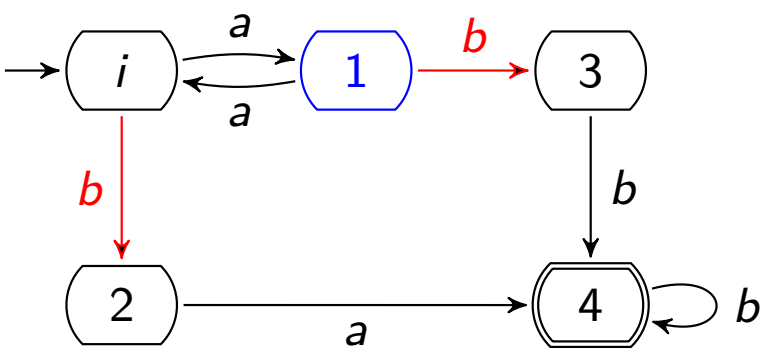


Figure: The minimal DFA recognizing $(aa)^*(abb + ba)b^*$

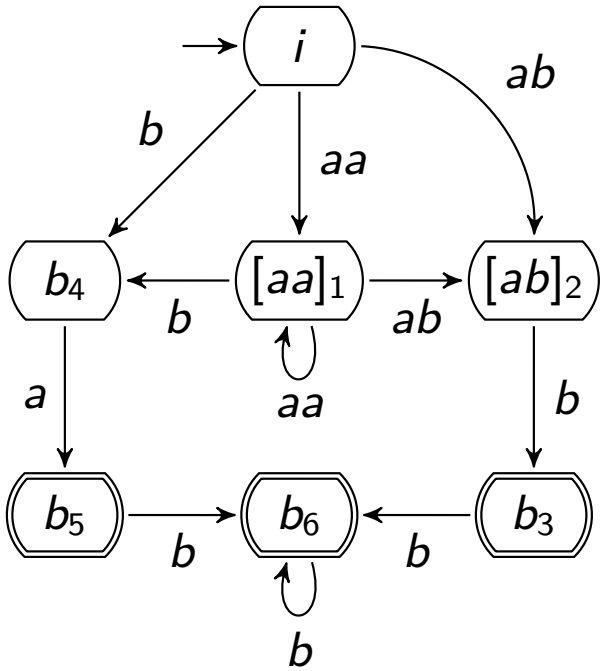


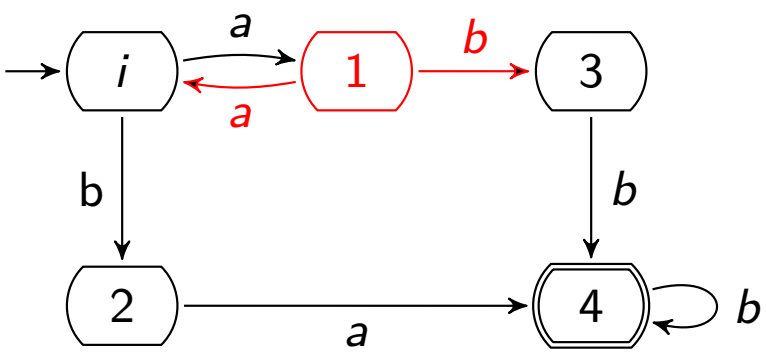
Figure: The Glushkov automaton of $E = [aa]^*([ab]b + ba)b^*$

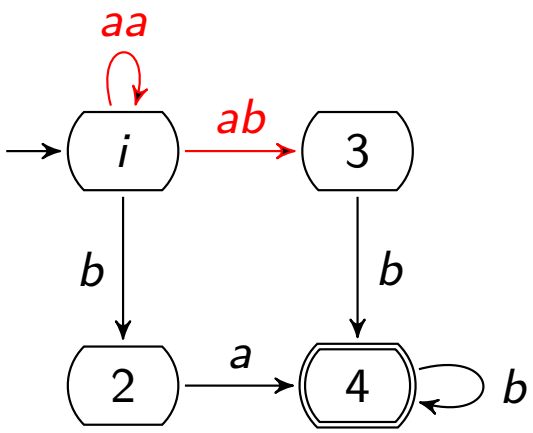
1. Preliminaries
   - Basics
   - 1-unambiguous regular languages
   - Block deterministic regular languages

2. Previous results on block deterministic regular languages
   - Determining whether a language L is block deterministic
   - The hierarchy of k-block deterministic regular languages

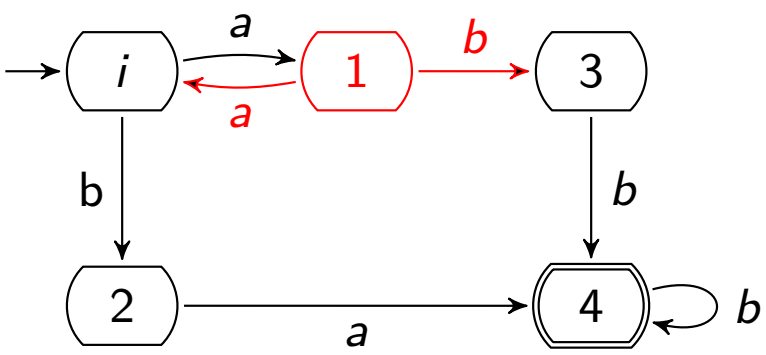
3. New results
   - Flawed results
   - A witness for the infinite hierarchy

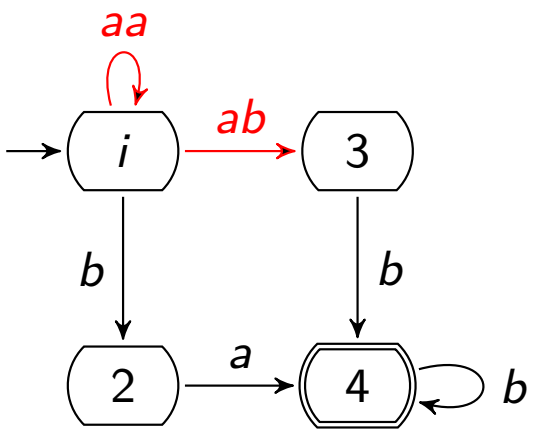
State elimination on block automaton





State elimination on block automaton

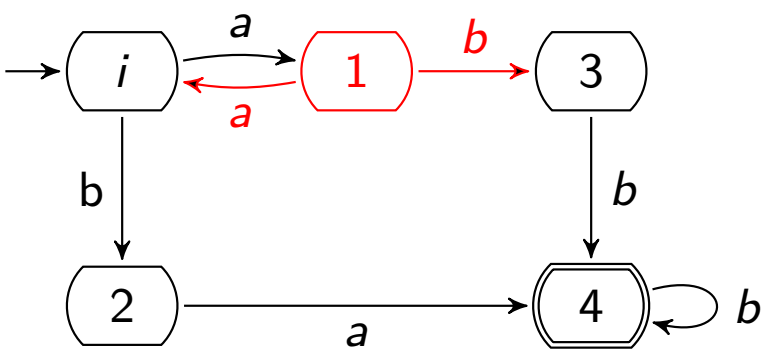


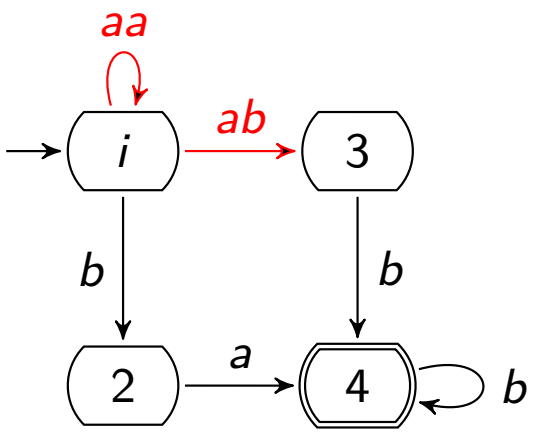


State elimination precondition

A state satisfies the *state elimination precondition* if it is neither initial nor final and it has no self-loop.

State elimination on block automaton





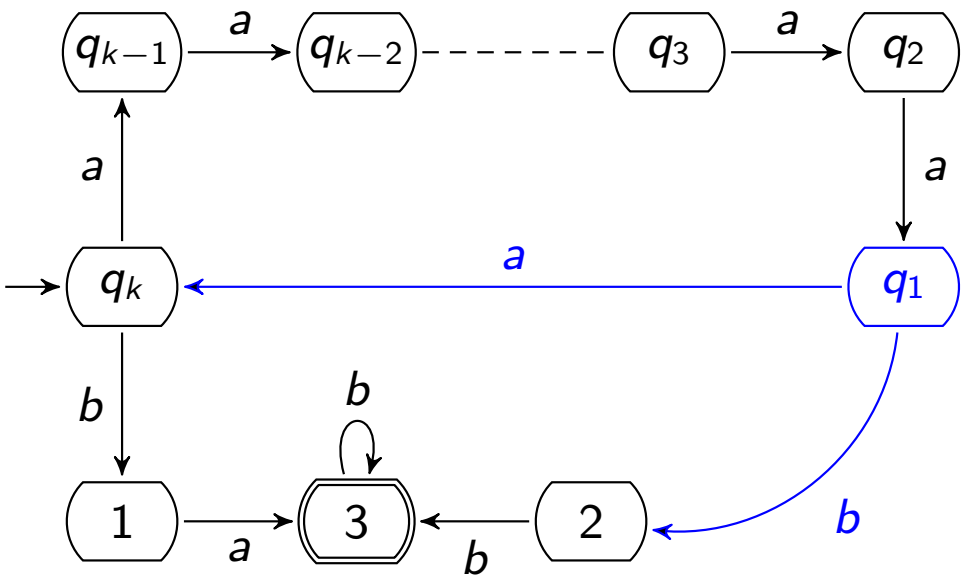
State elimination precondition

A state satisfies the *state elimination precondition* if it is neither initial nor final and it has no self-loop.

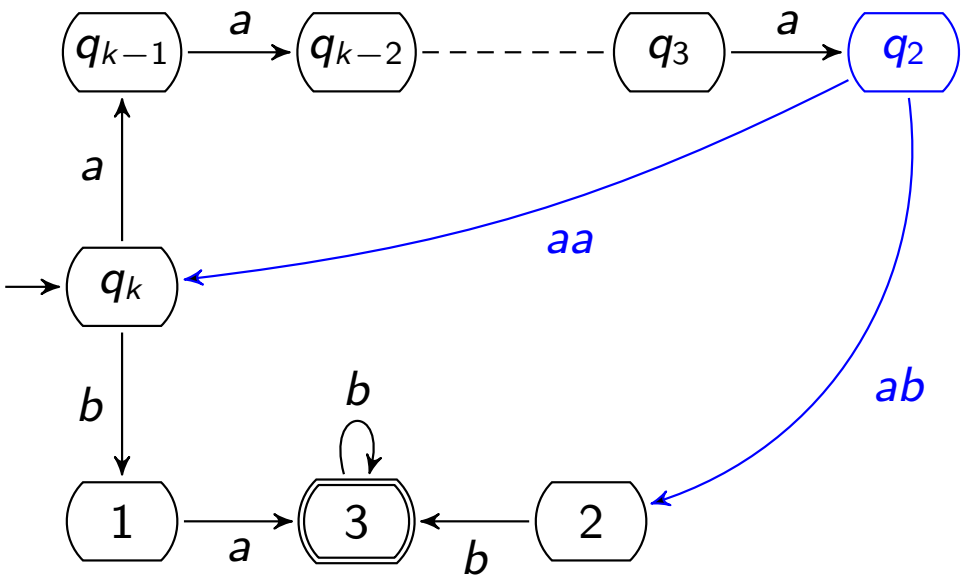
State elimination lemma

If a language is k-block deterministic, then, from the minimal DFA, we can get a k-block deterministic automaton with the orbit property using only state elimination.

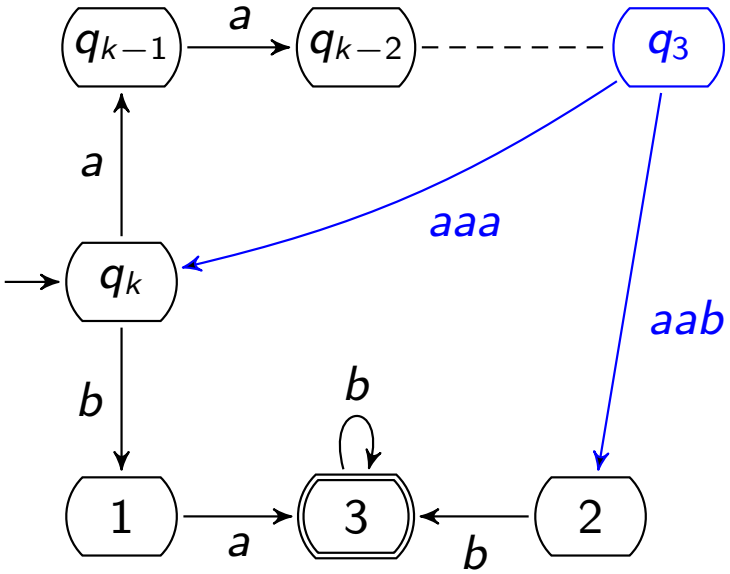
Han and Wood parameterized family: HW_k



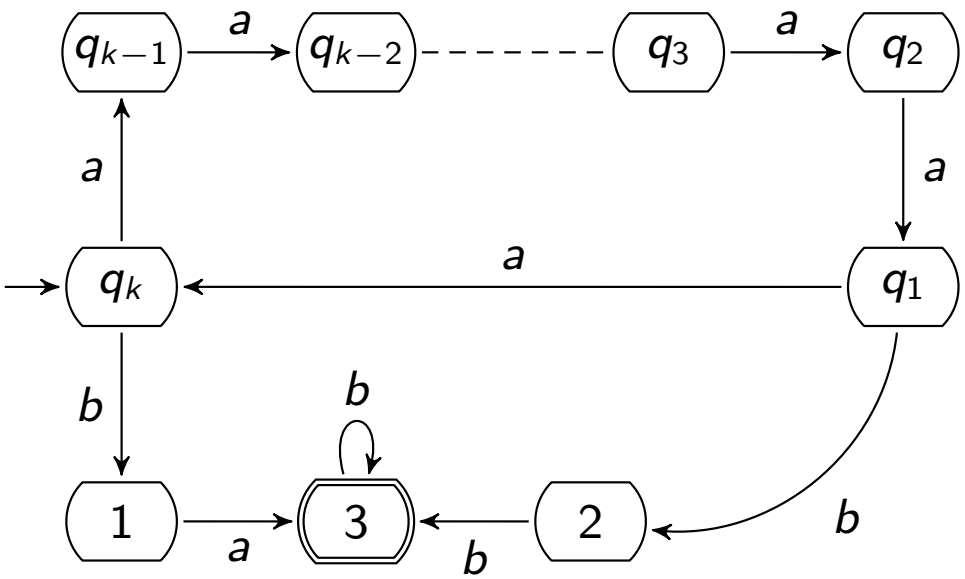
Han and Wood parameterized family: HW_k

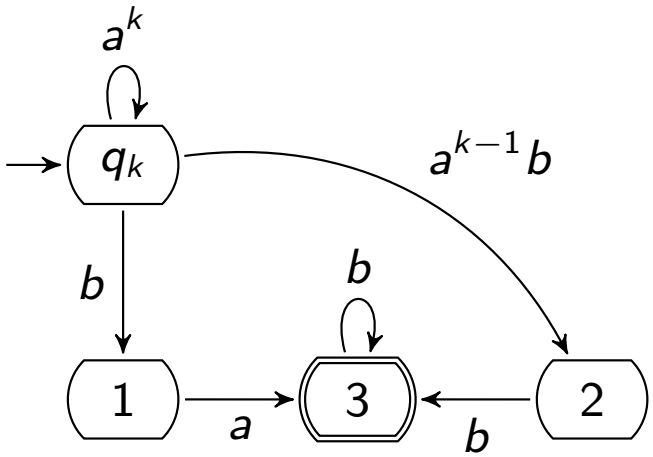


Han and Wood parameterized family: HW_k



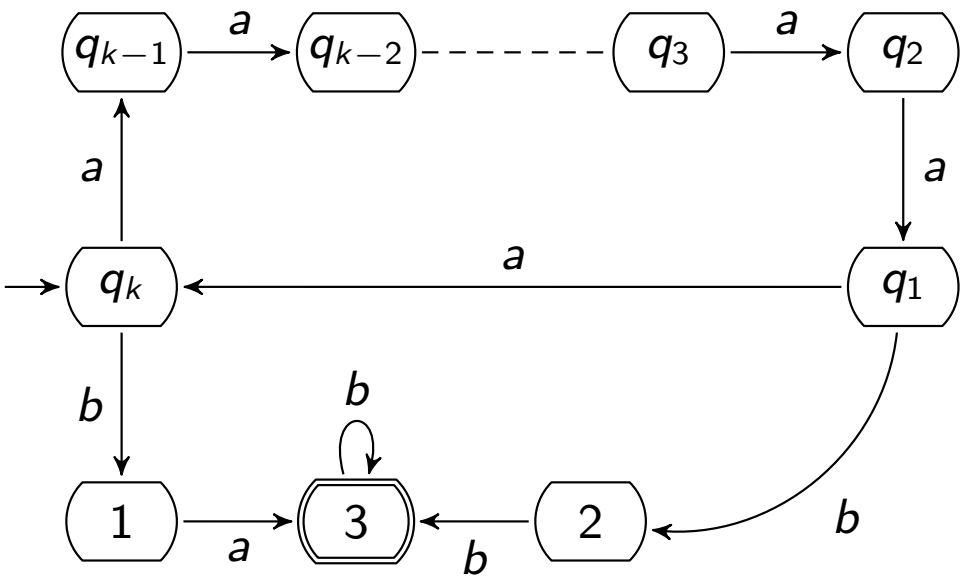
Han and Wood parameterized family: HW_k

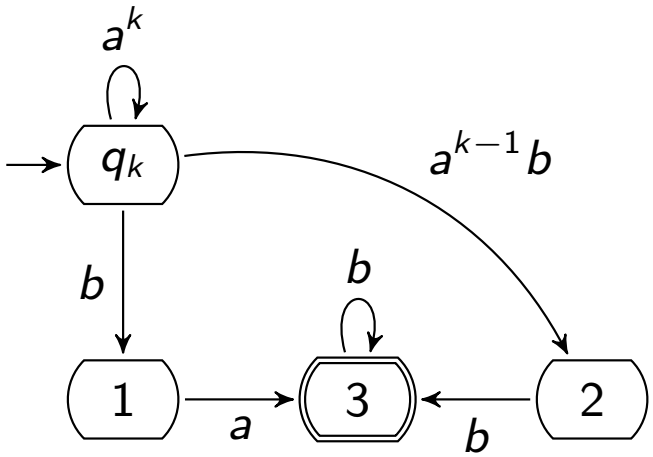




The family of languages HW_k can be denoted by $E_k = [a^k]^*([a^{k-1}b]b + ba)b^*$.

Han and Wood parameterized family: HW_k





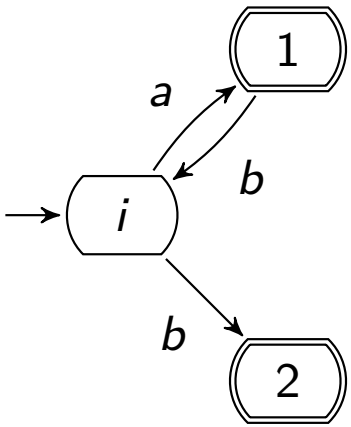
The family of languages HW_k can be denoted by $E_k = [a^k]^*([a^{k-1}b]b + ba)b^*$.

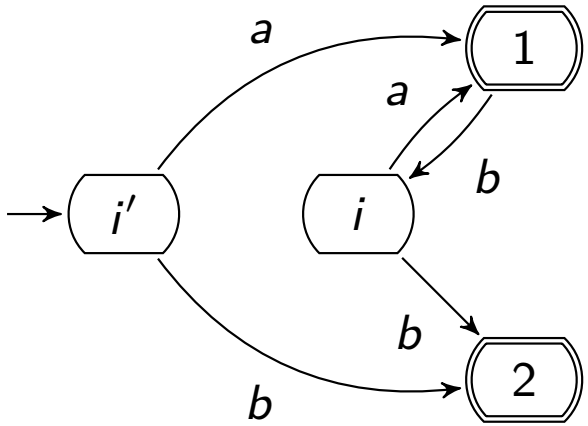
It is not $(k-1)$ since we have to eliminate states q_1 to q_{k-1} to obtain the orbit property.

A counter-example for the lemma

Standard automaton

An automaton $A = (\Sigma, Q, I, F, \delta)$ is *standard* if $|I| = 1$ and $\forall (p, a, q) \in \delta, q \notin I$.

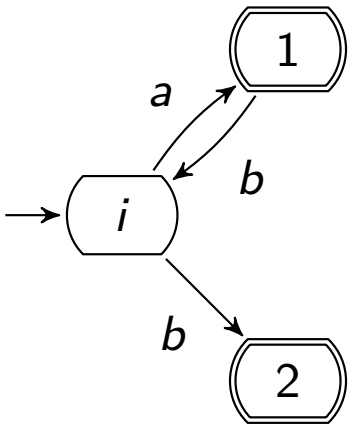


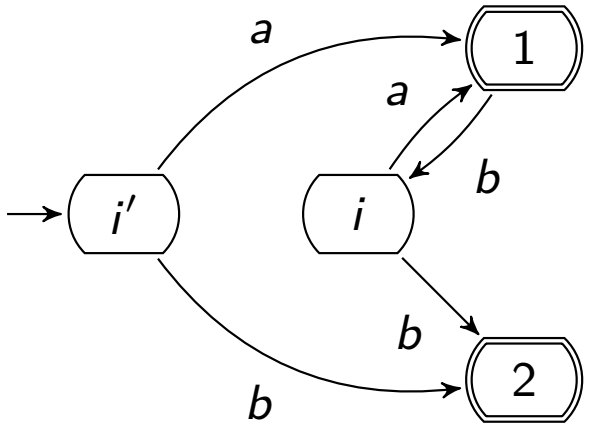


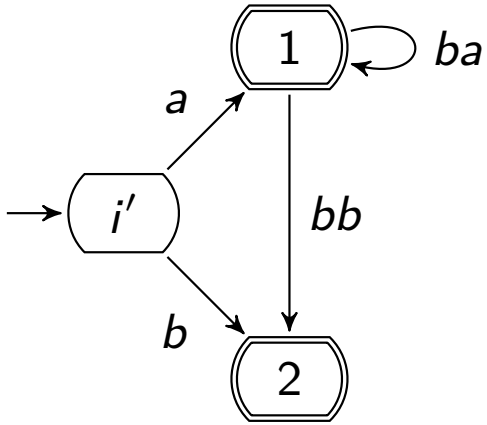
A counter-example for the lemma

Standard automaton

An automaton $A = (\Sigma, Q, I, F, \delta)$ is *standard* if $|I| = 1$ and $\forall (p, a, q) \in \delta, q \notin I$.



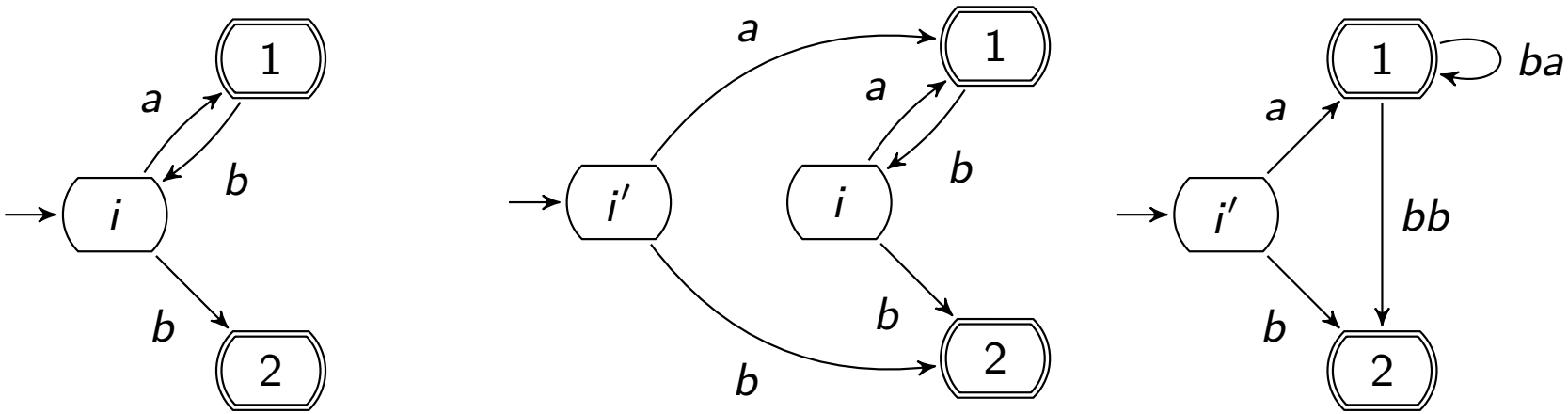




A counter-example for the lemma

Standard automaton

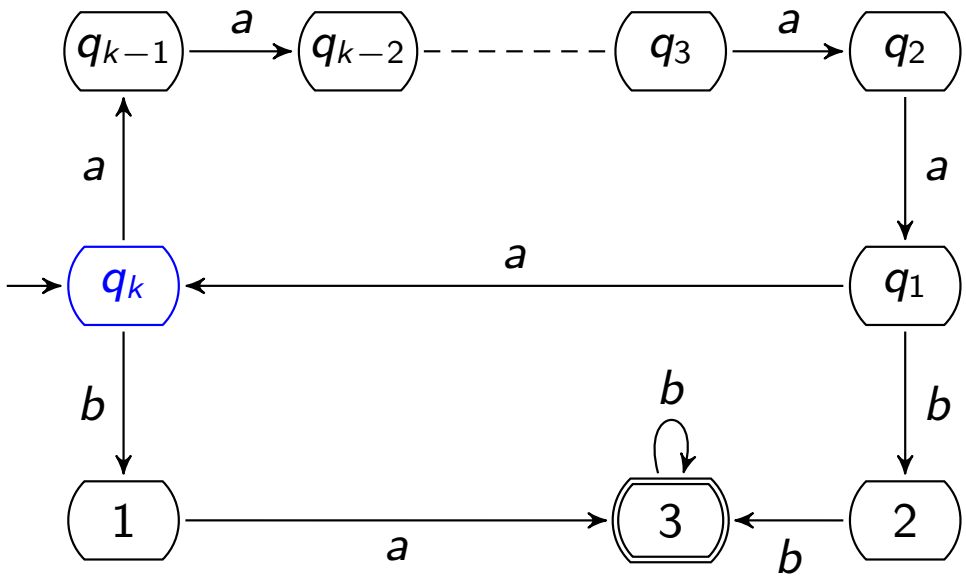
An automaton $A = (\Sigma, Q, I, F, \delta)$ is *standard* if $|I| = 1$ and $\forall (p, a, q) \in \delta, q \notin I$.



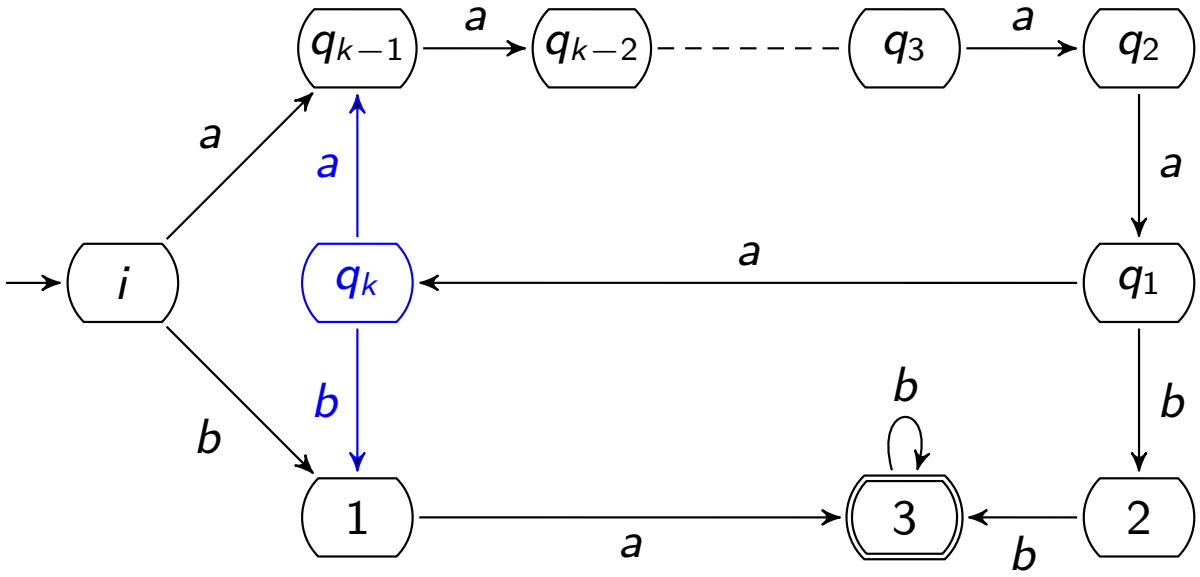
State elimination lemma is erroneous

State elimination alone from the minimal DFA is not sufficient to determine if a regular language is block deterministic.

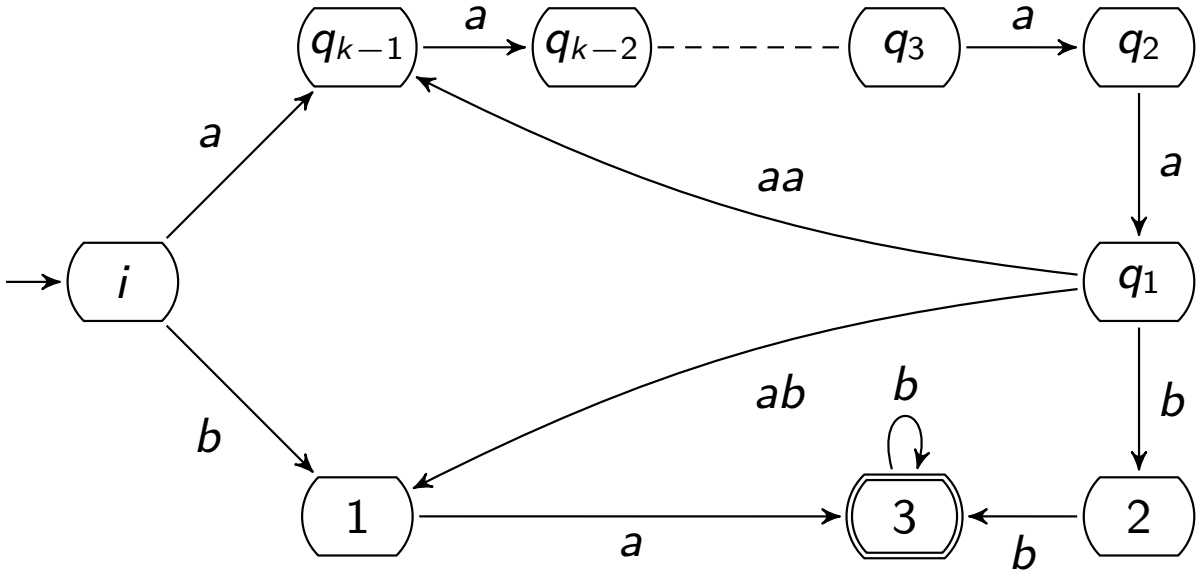
HW_k is 2-block deterministic for all $k \geq 2$



HW_k is 2-block deterministic for all $k \geq 2$

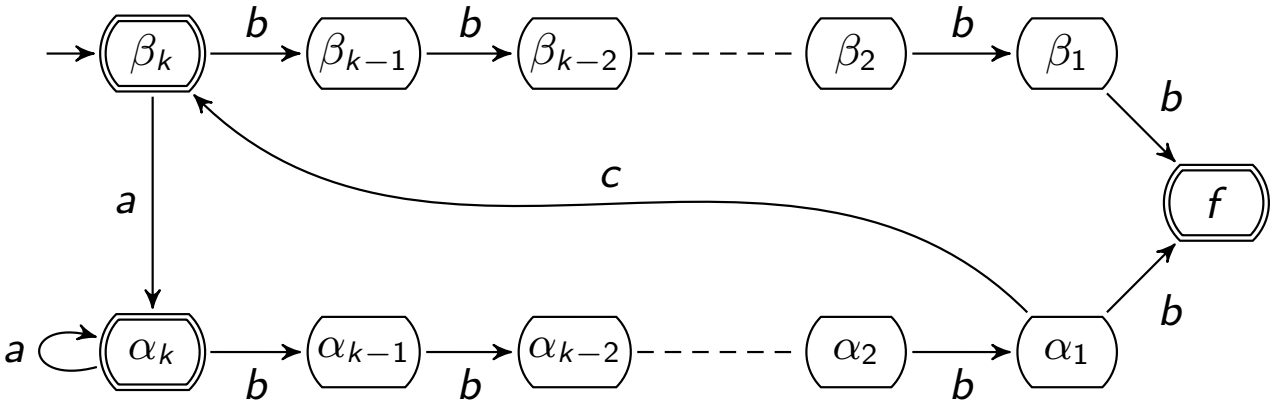


HW_k is 2-block deterministic for all $k \geq 2$

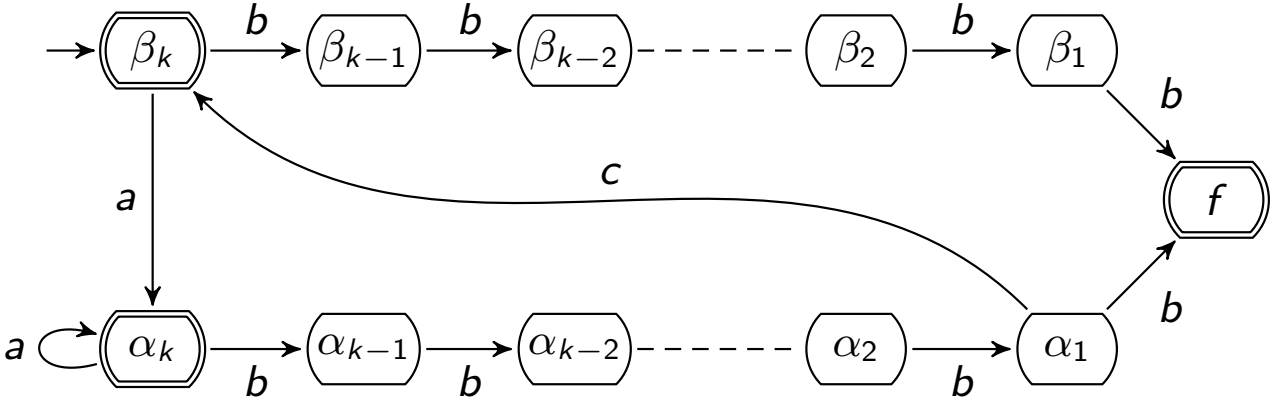


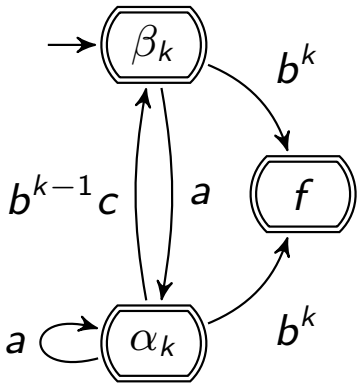
The family of languages HW_k can be denoted by
$F_k = (a^{k-1}([aa]a^{k-2})^*([ab]a + bb) + ba)b^*$

Our parameterized family: CMM_k



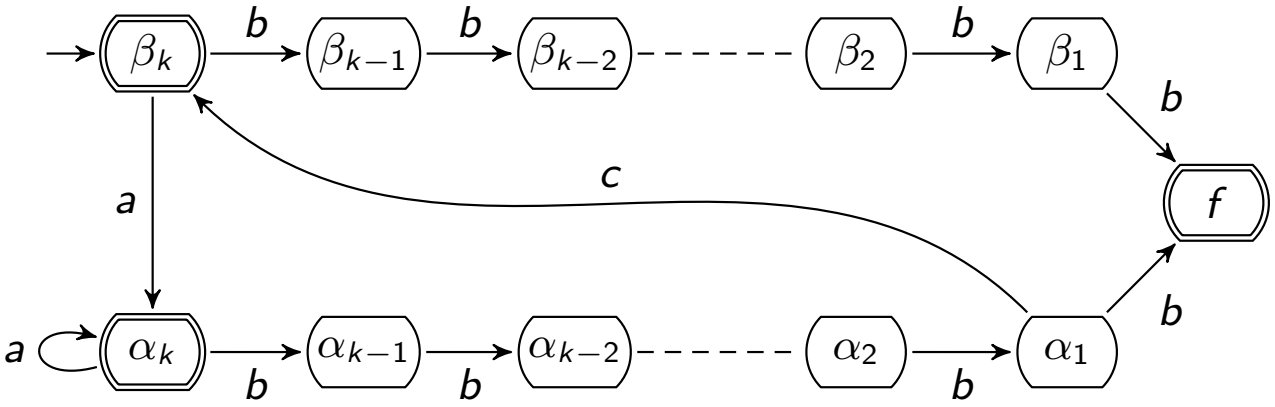
Our parameterized family: CMM_k

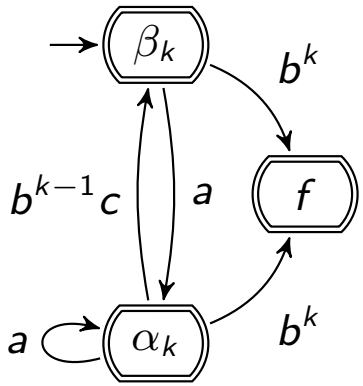




The family of languages CMM_k denoted by $E_k = (a(\varepsilon + [b^{k-1}c]))^*(\varepsilon + [b^k])$

Our parameterized family: CMM_k



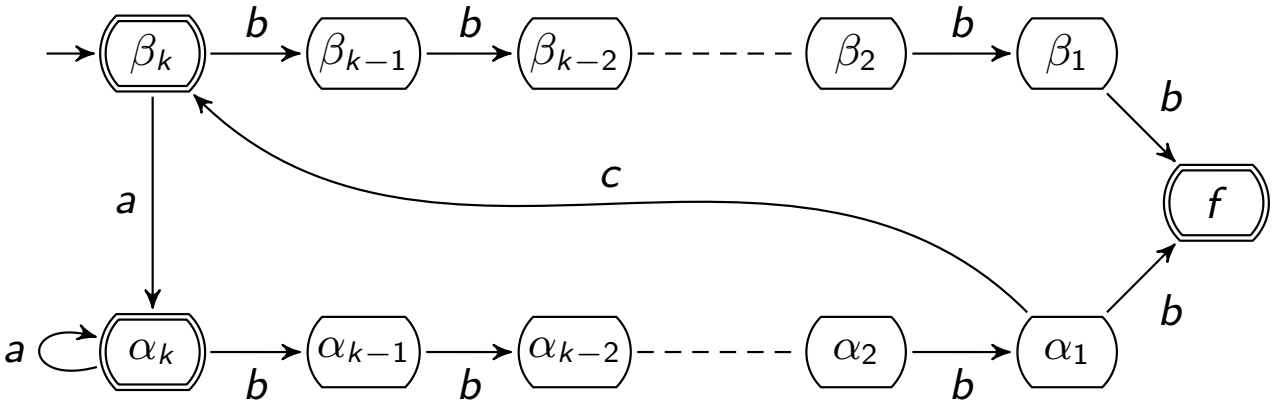


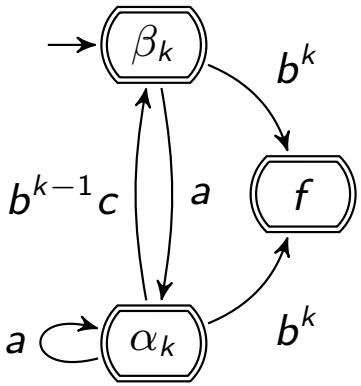
The family of languages CMM_k denoted by $E_k = (a(\varepsilon + [b^{k-1}c]))^*(\varepsilon + [b^k])$

Why is CMM_k not $(k-1)$-block deterministic: some ideas of the proof

- If CMM_k is $(k-1)$-block deterministic, then it is recognized by a $(k-1)$-block deterministic automaton with the orbit property.

Our parameterized family: CMM_k



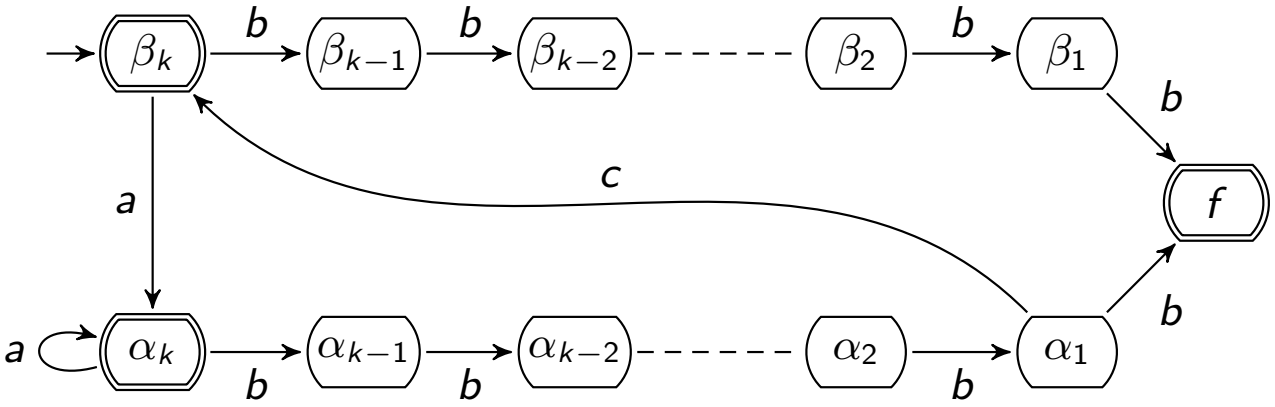


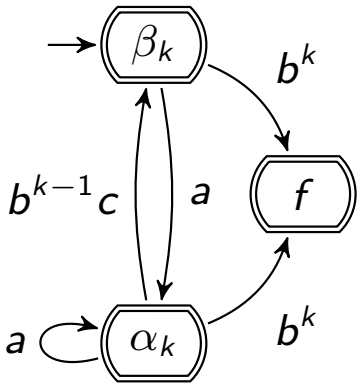
The family of languages CMM_k denoted by $E_k = (a(\varepsilon + [b^{k-1}c]))^*(\varepsilon + [b^k])$

Why is CMM_k not $(k-1)$-block deterministic: some ideas of the proof

- If CMM_k is $(k-1)$-block deterministic, then it is recognized by a $(k-1)$-block deterministic automaton with the orbit property.
- Since α_k and β_k are final, this $(k-1)$-block deterministic automaton must have an orbit with two states equivalent to α_k and β_k.

Our parameterized family: CMM_k





The family of languages CMM_k denoted by $E_k = (a(\varepsilon + [b^{k-1}c]))^*(\varepsilon + [b^k])$

Why is CMM_k not $(k-1)$-block deterministic: some ideas of the proof

- If CMM_k is $(k-1)$-block deterministic, then it is recognized by a $(k-1)$-block deterministic automaton with the orbit property.
- Since α_k and β_k are final, this $(k-1)$-block deterministic automaton must have an orbit with two states equivalent to α_k and β_k.
- In this orbit, the orbit property cannot be obtained with blocks of length $k-1$.

Thank you for your attention

Do you have any question?