Computational Experiments with Synchronizing Automata

Mikhail Volkov

Ural Federal University, Ekaterinburg, Russia

Lisbon, July 23rd, 2014

Definitions and Terminology

We consider complete deterministic finite automata (DFA)

$\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.

$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.

$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.
$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.
$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.
$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma \rangle$ where Q stands for the state set, Σ is the input alphabet, and each letter $a \in \Sigma$ acts on the set Q as a total function: $q \mapsto q \,.\, a$.

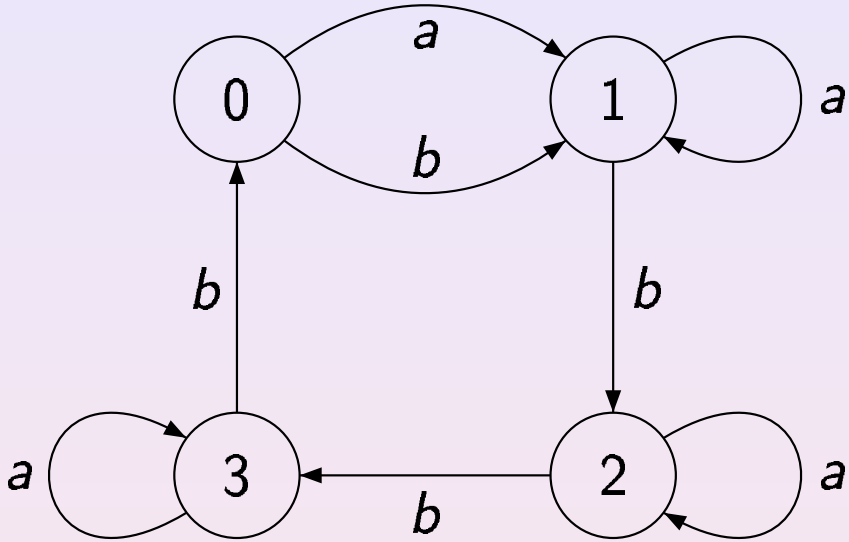
Then each word $w = a_1 \dots a_\ell \in \Sigma^*$ also acts on Q:
$q \,.\, w := (\dots((q \,.\, a_1) \,.\, a_2) \dots) \,.\, a_\ell$.
$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q it started: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.
In short, $|Q \,.\, w| = 1$.

Any w with this property is a reset word for $\mathscr{A}$.

Other names:

- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

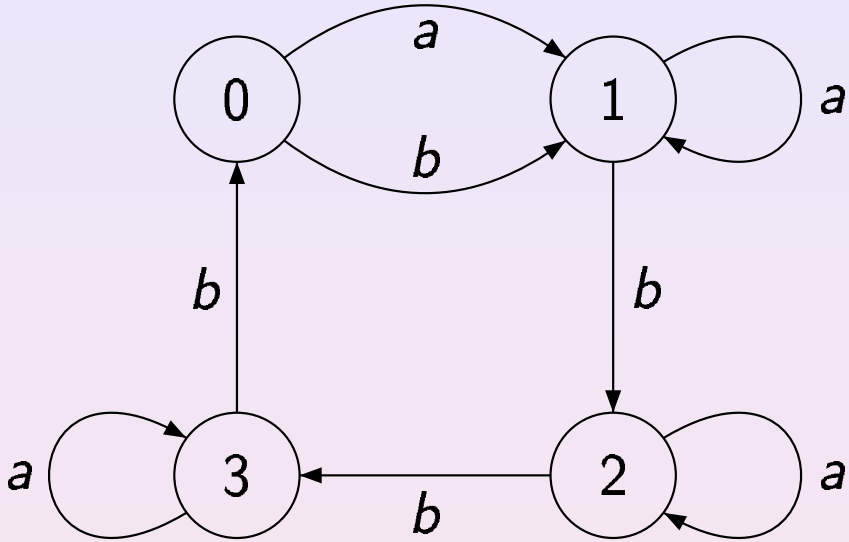
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

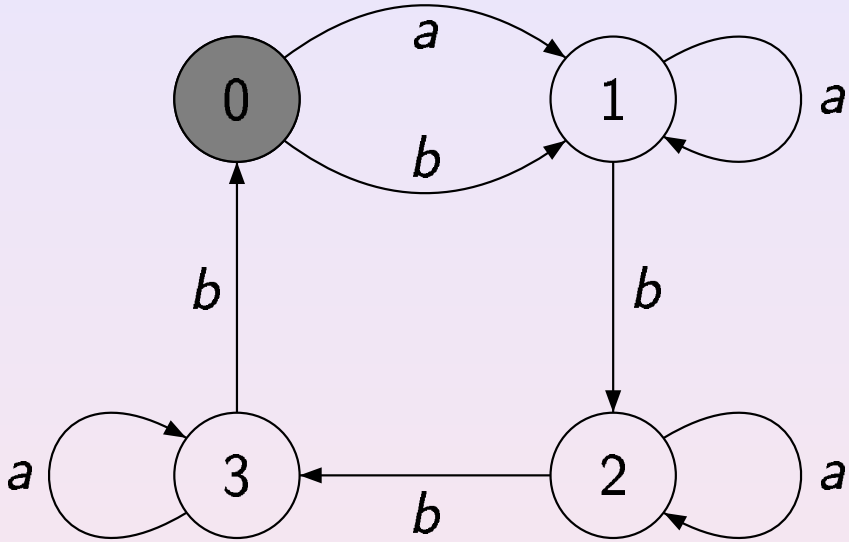
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

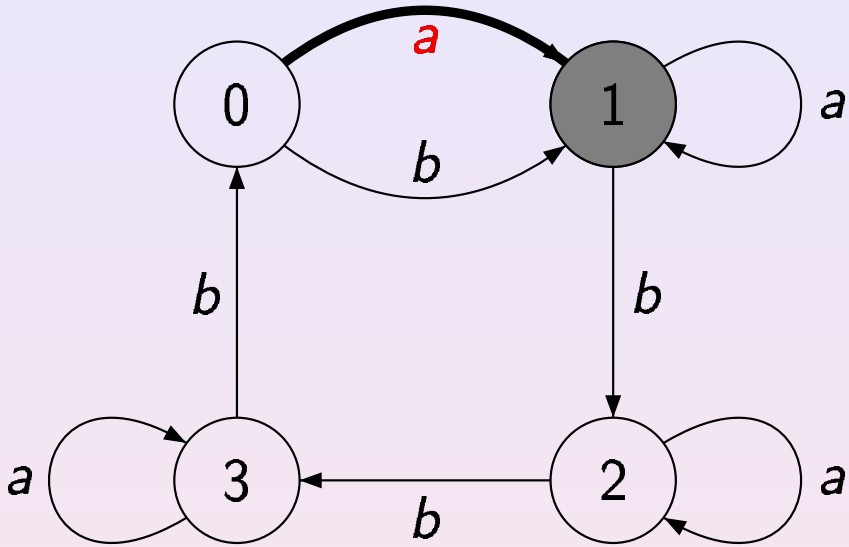
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

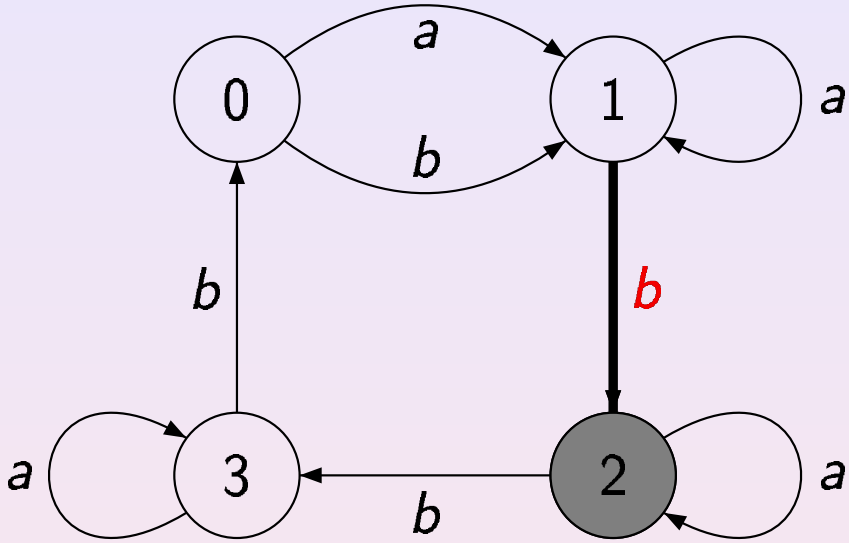
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

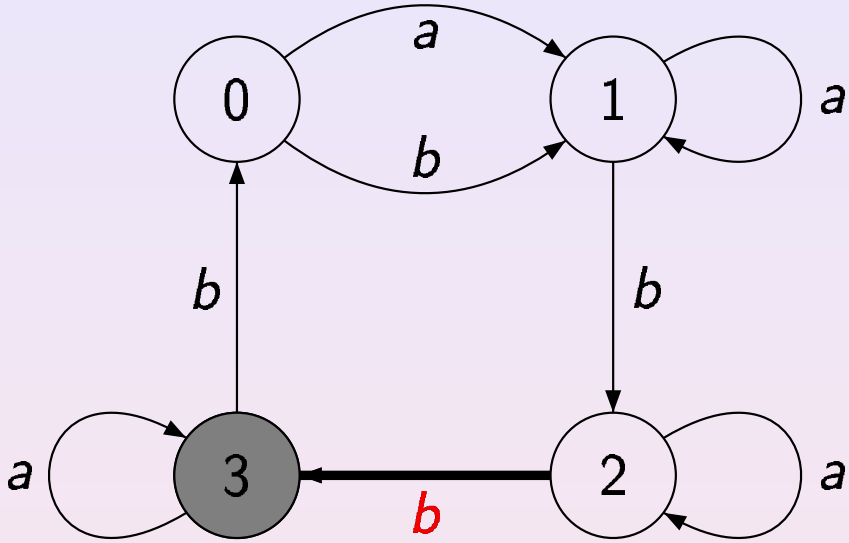
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

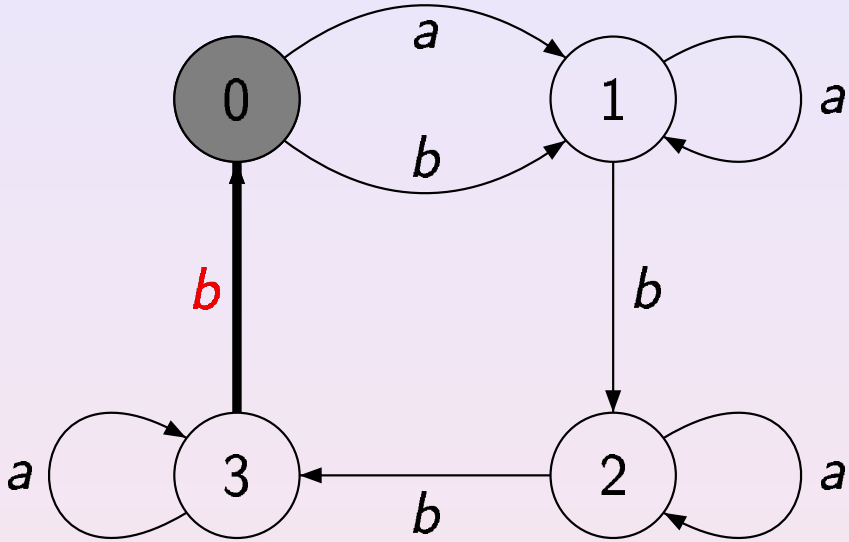
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

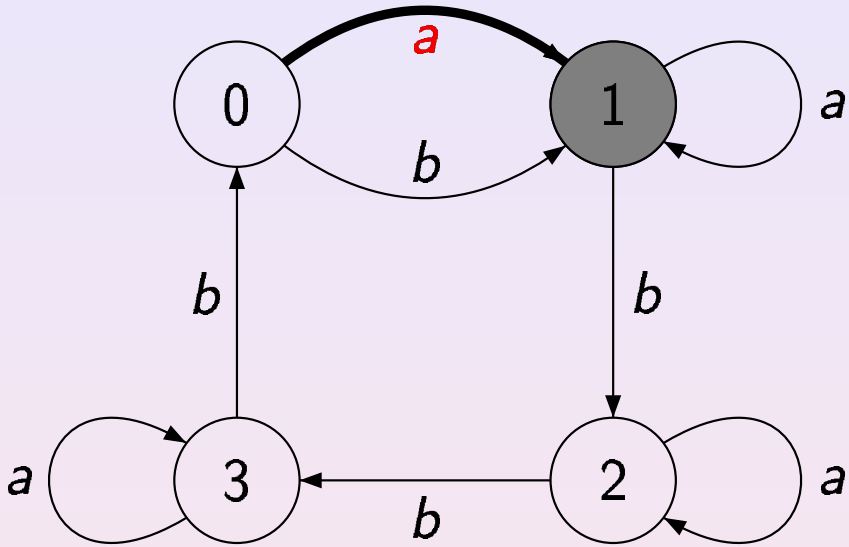
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

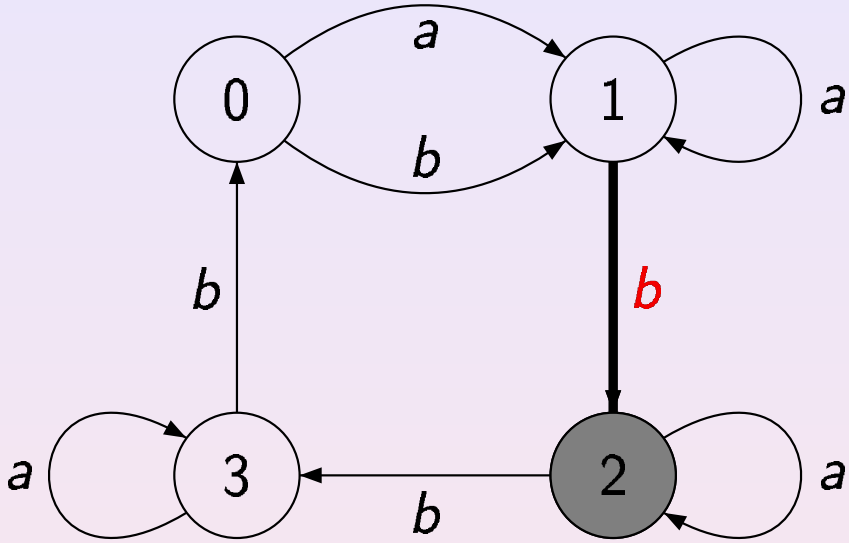
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

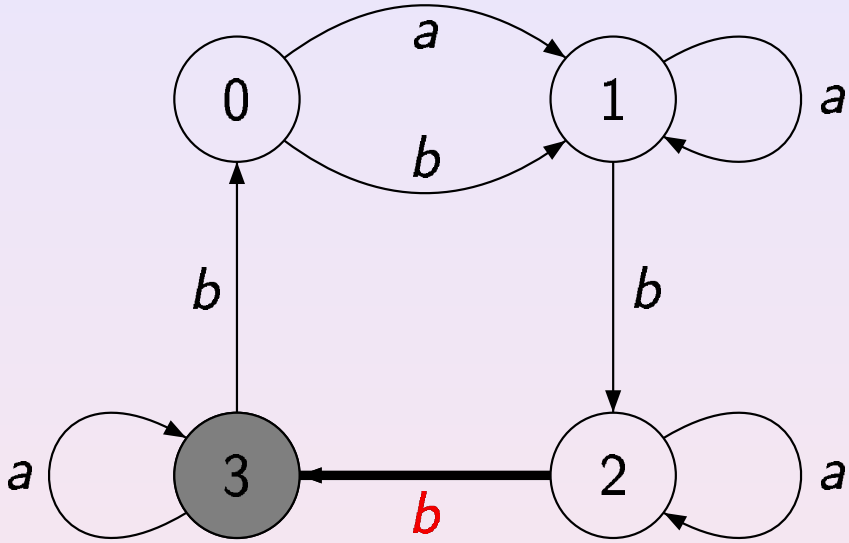
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

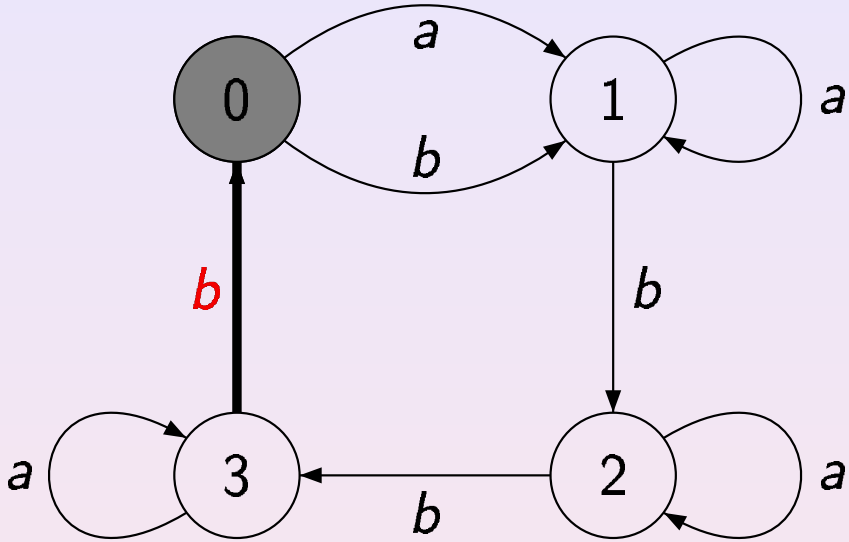
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

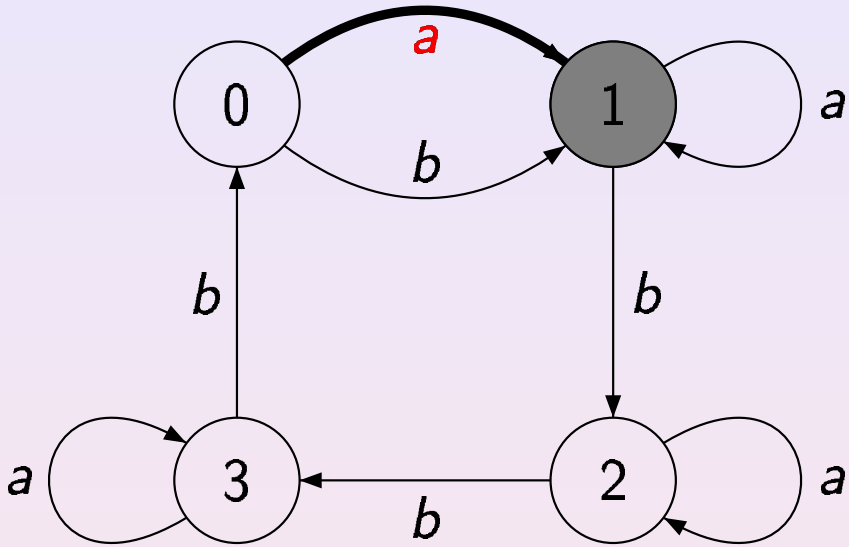
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

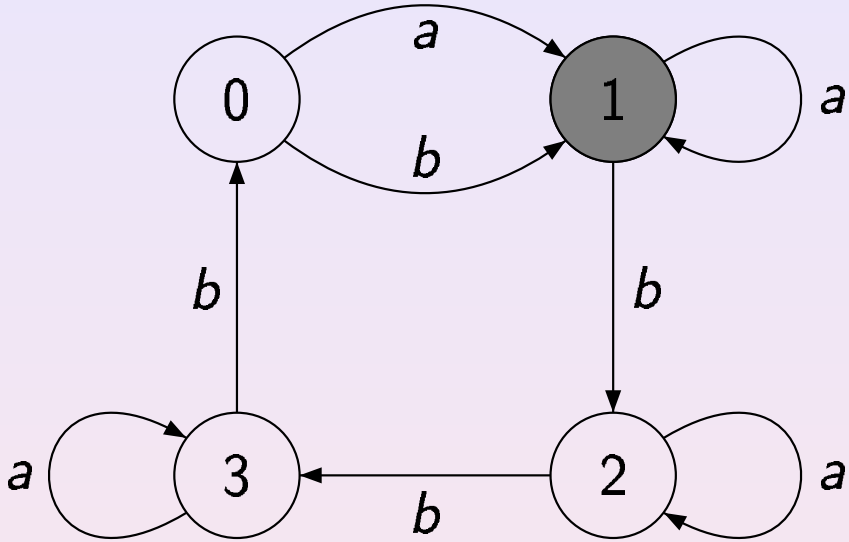
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

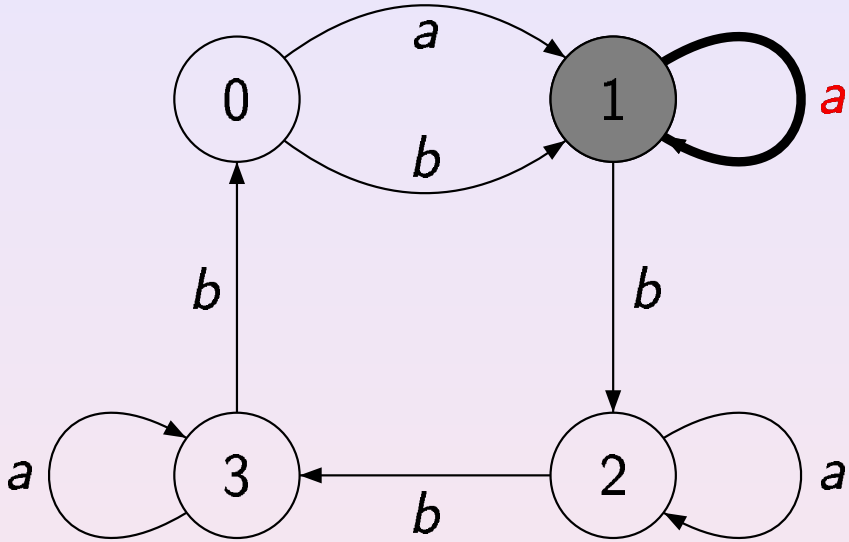
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

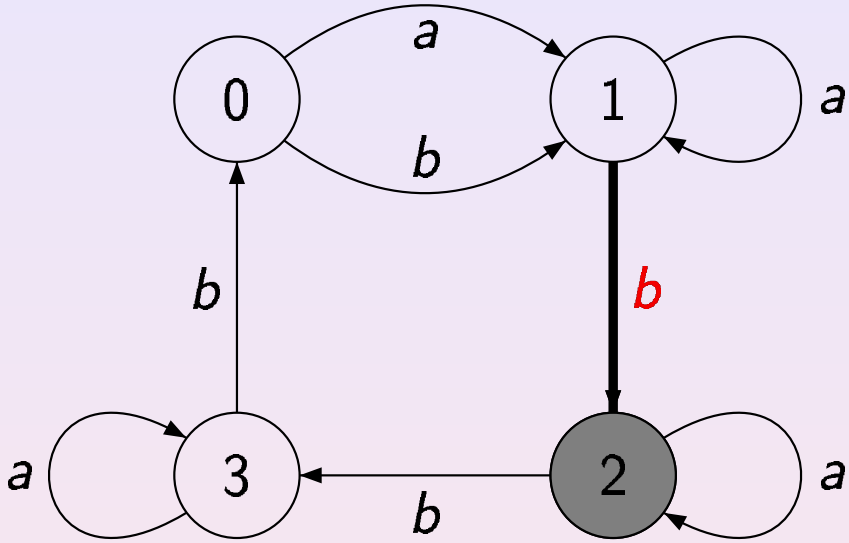
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

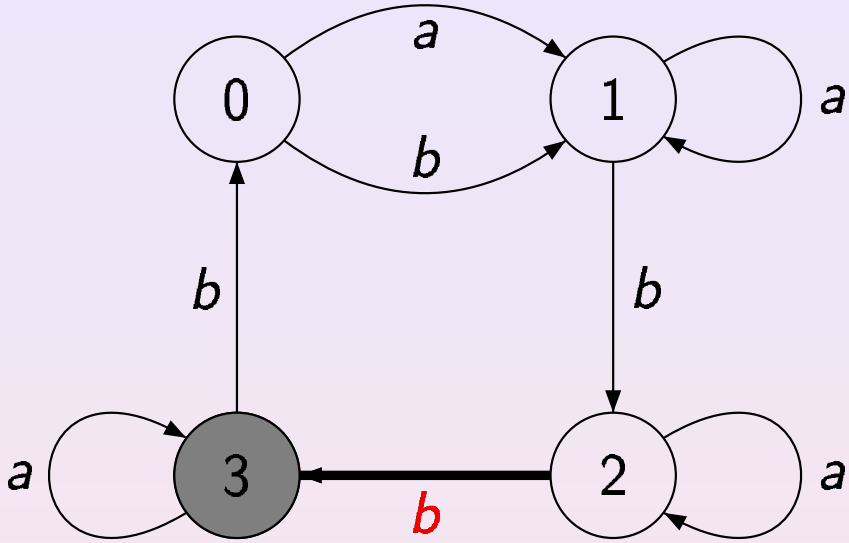
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

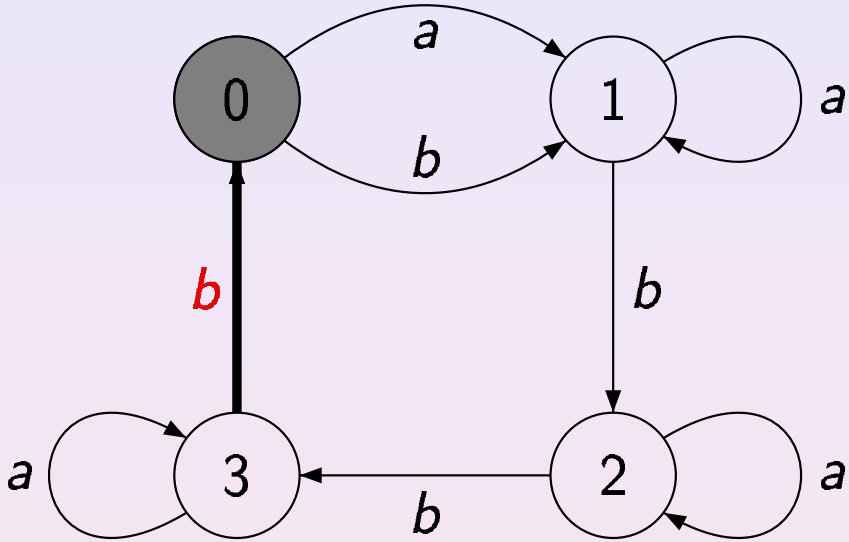
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

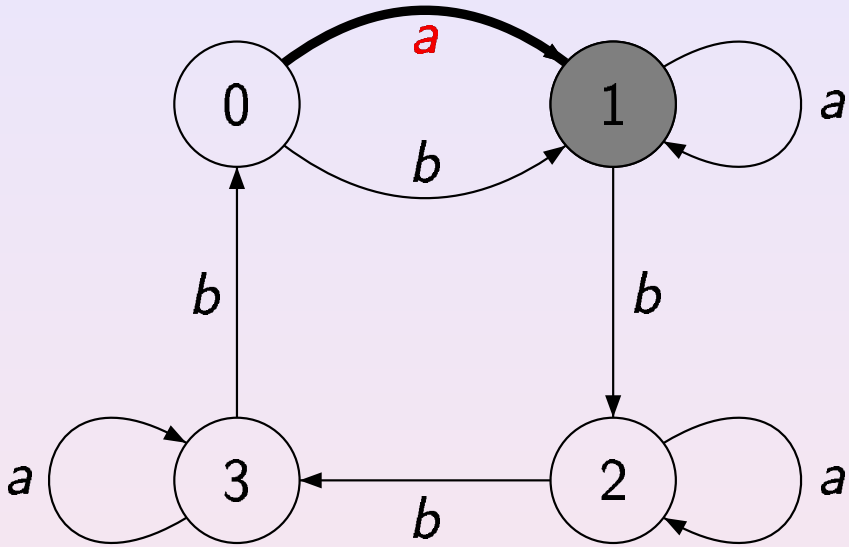
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

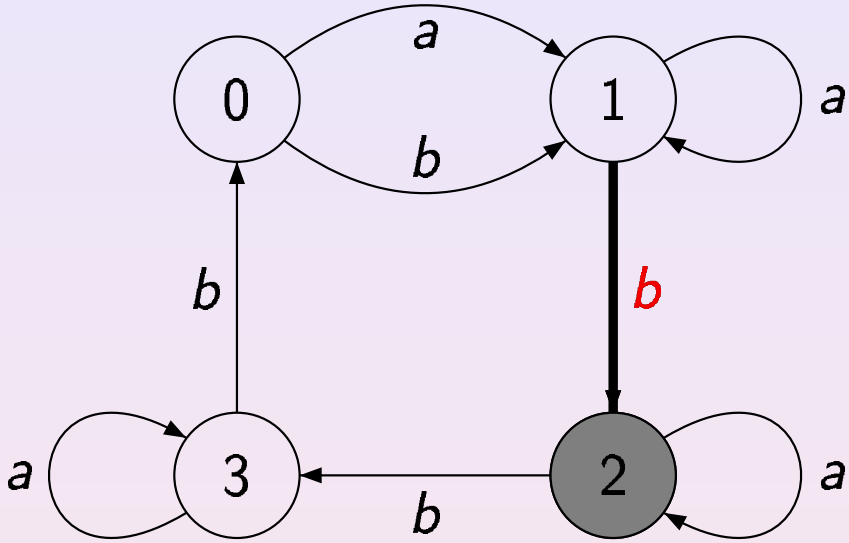
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

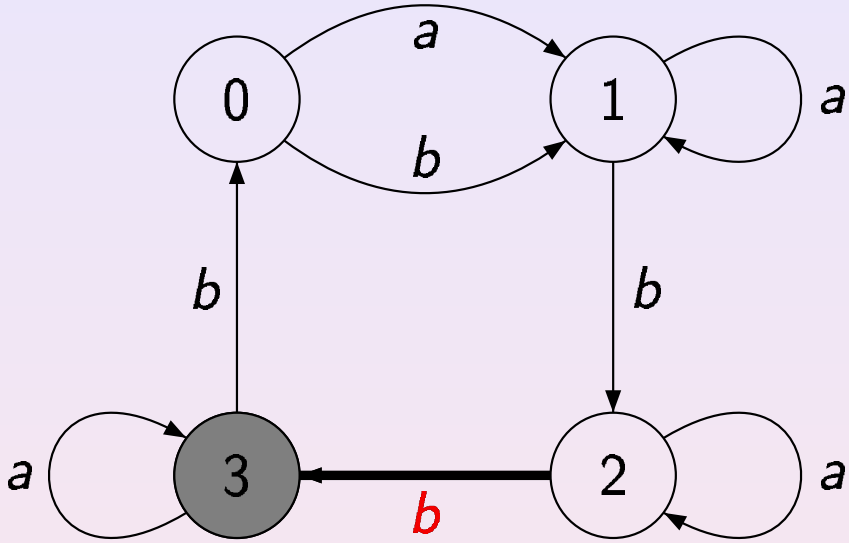
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

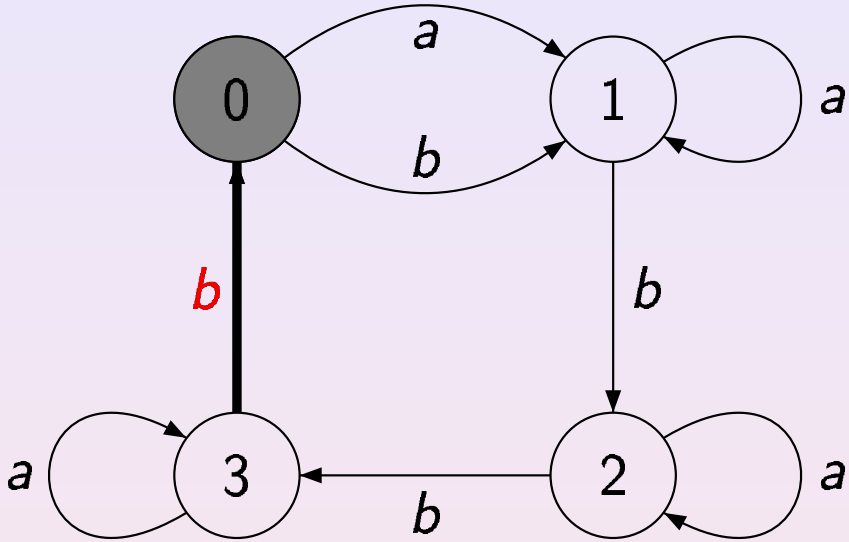
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

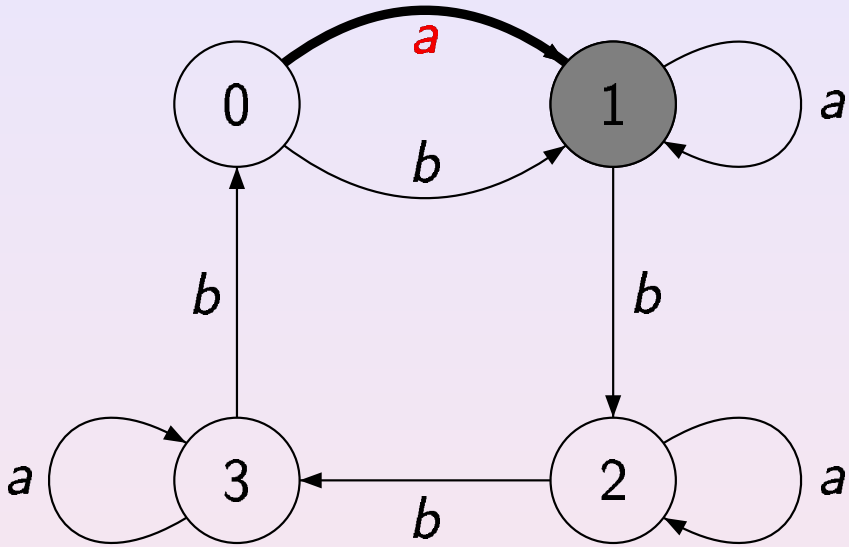
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

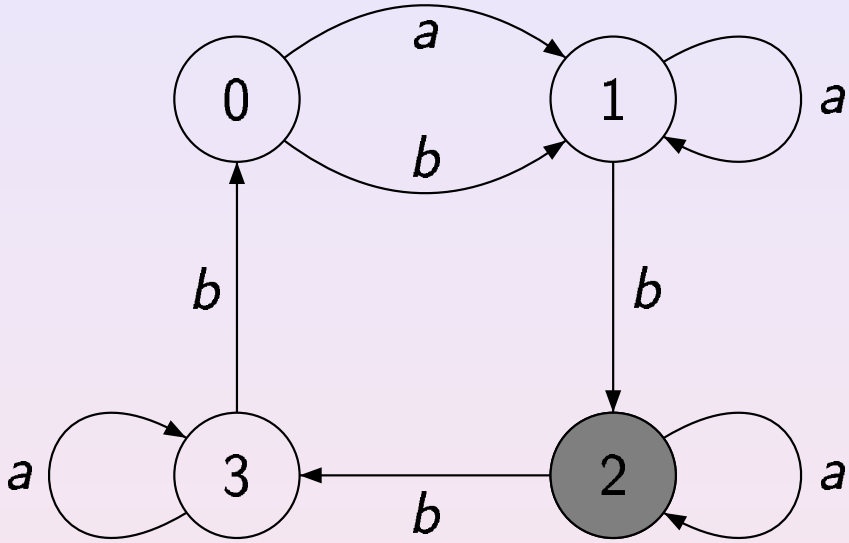
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

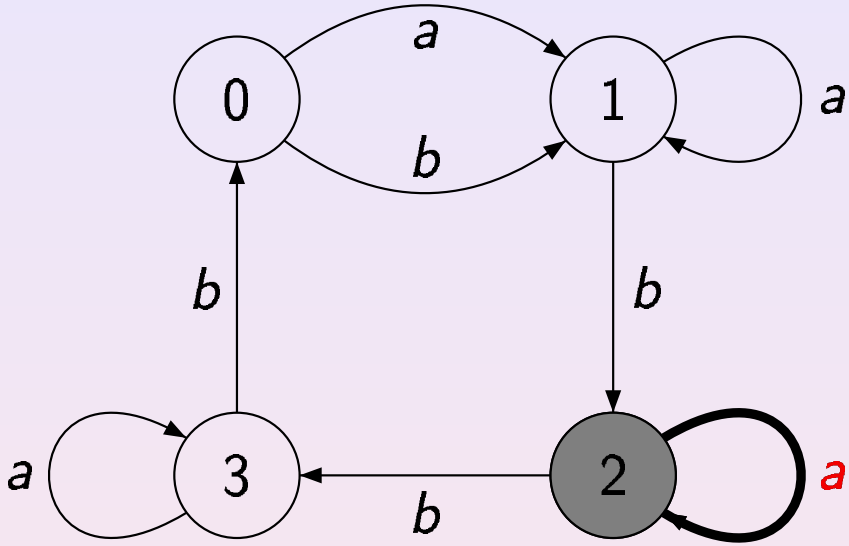
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

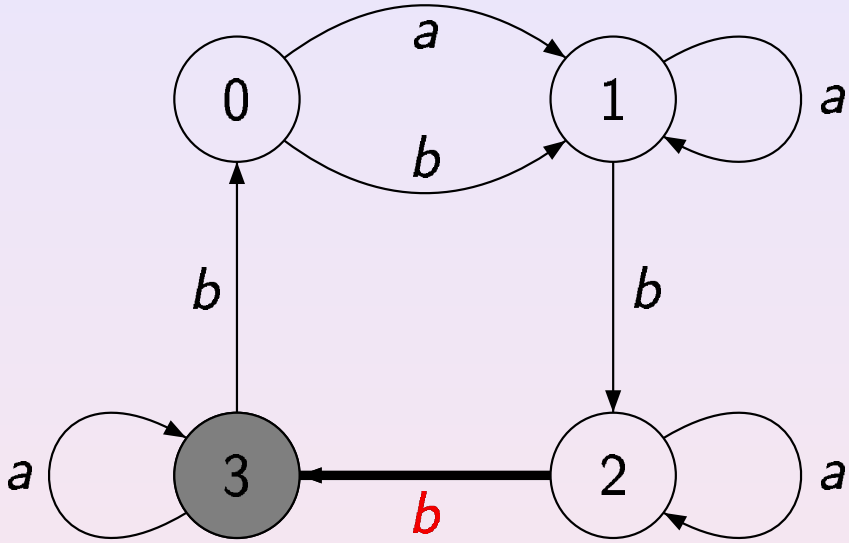
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

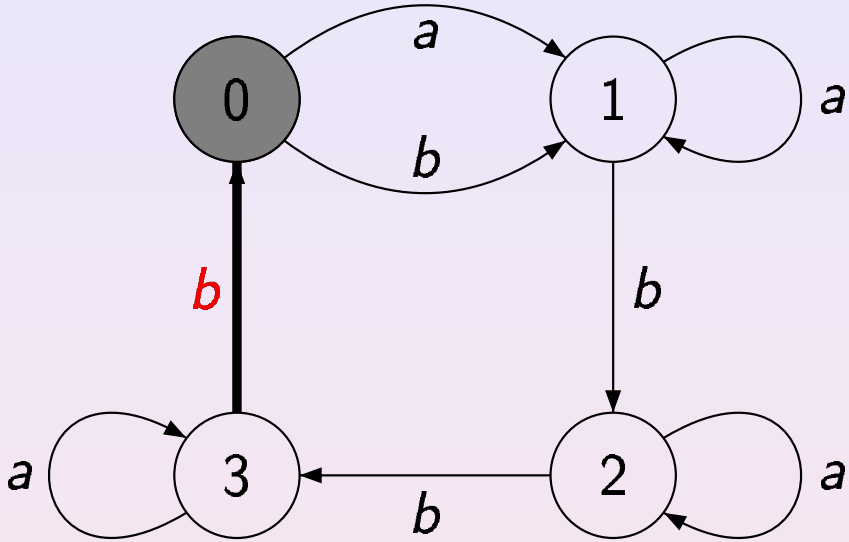
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

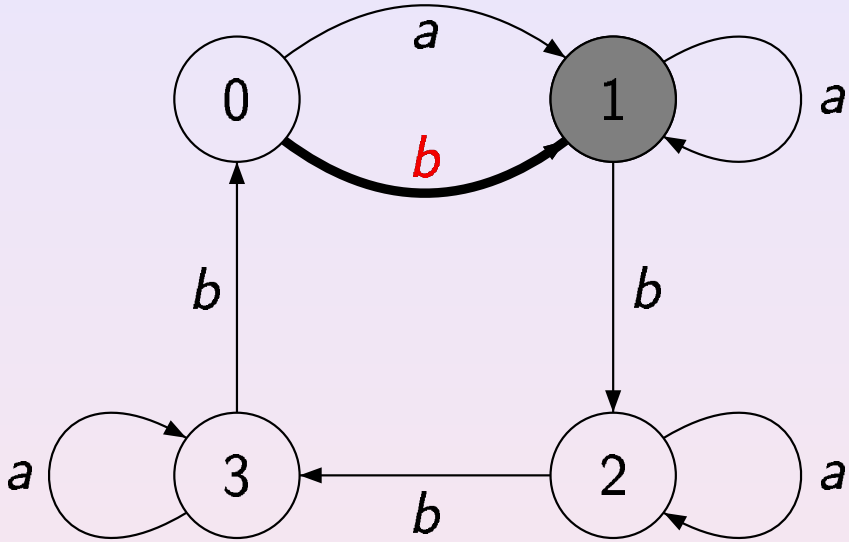
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

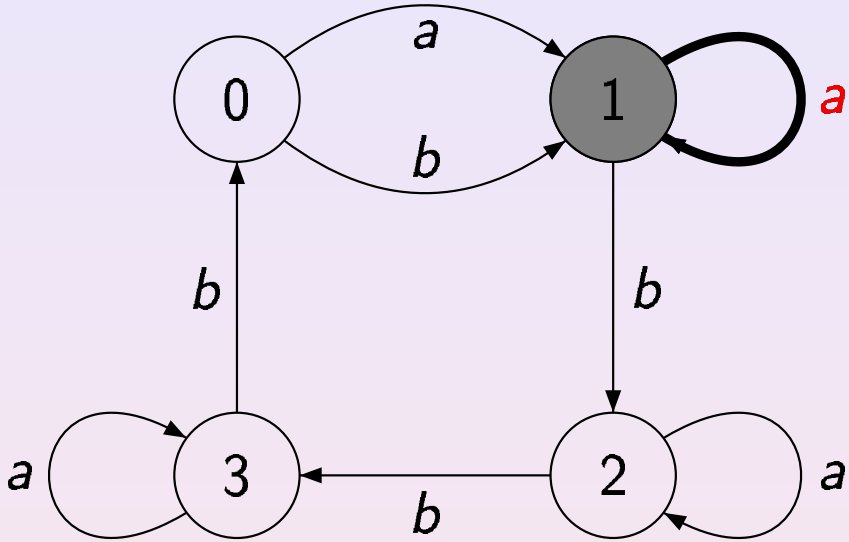
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

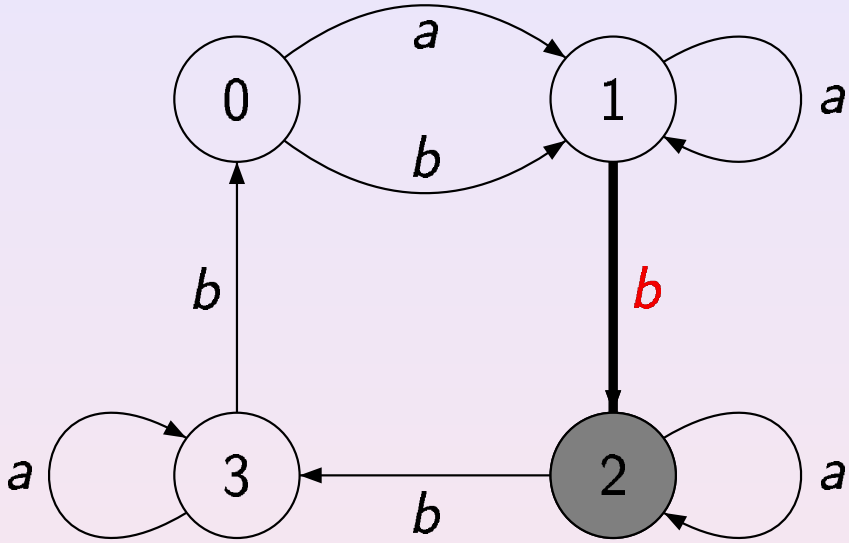
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

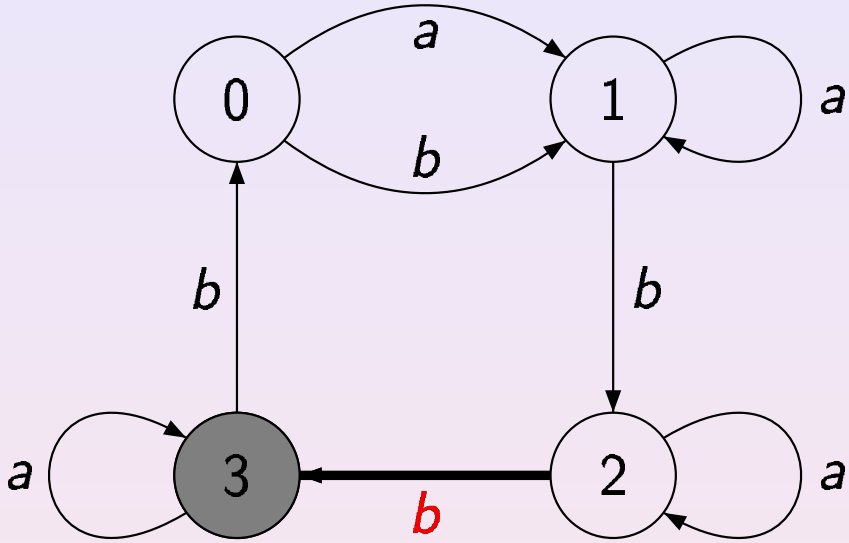
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

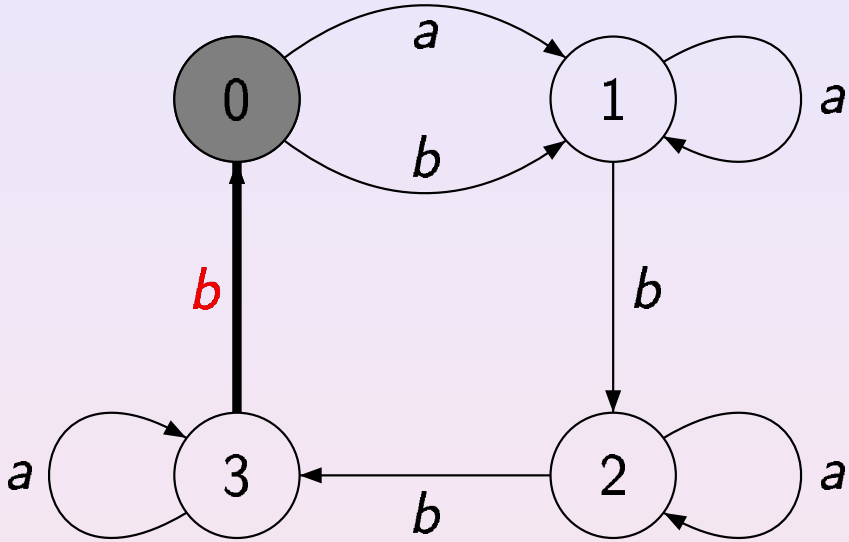
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

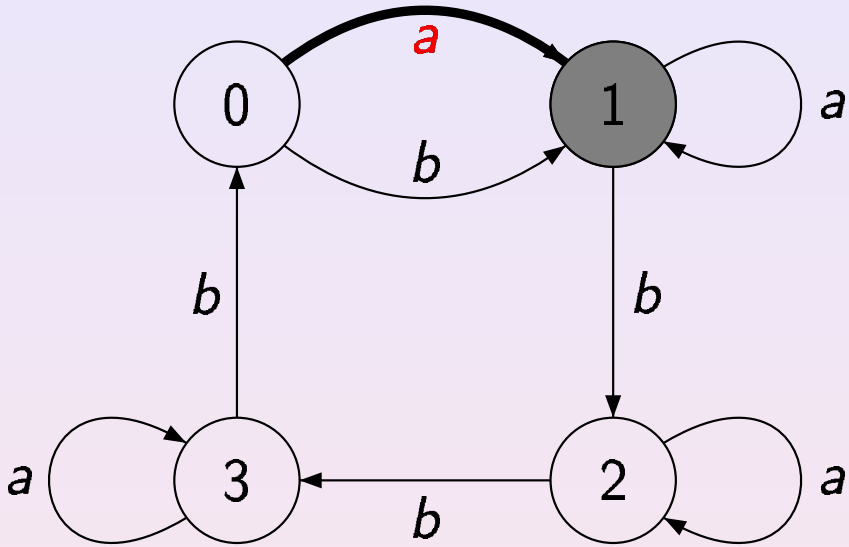
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

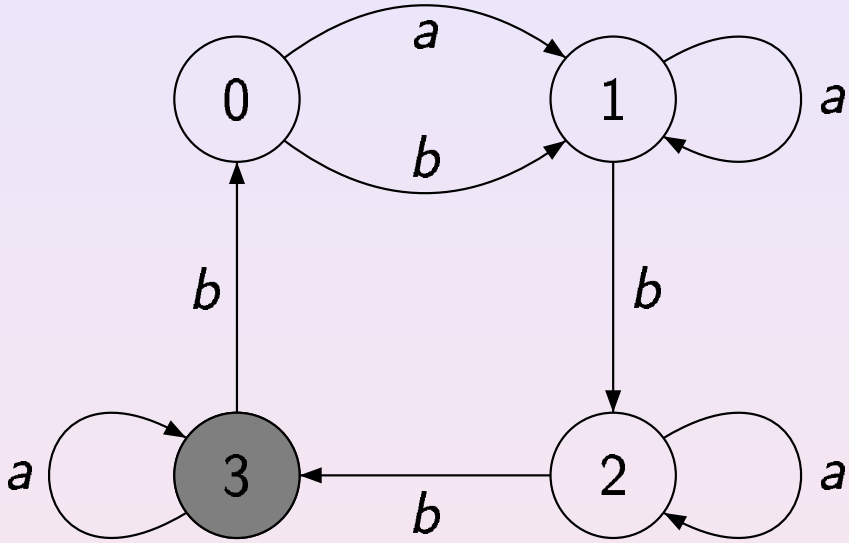
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

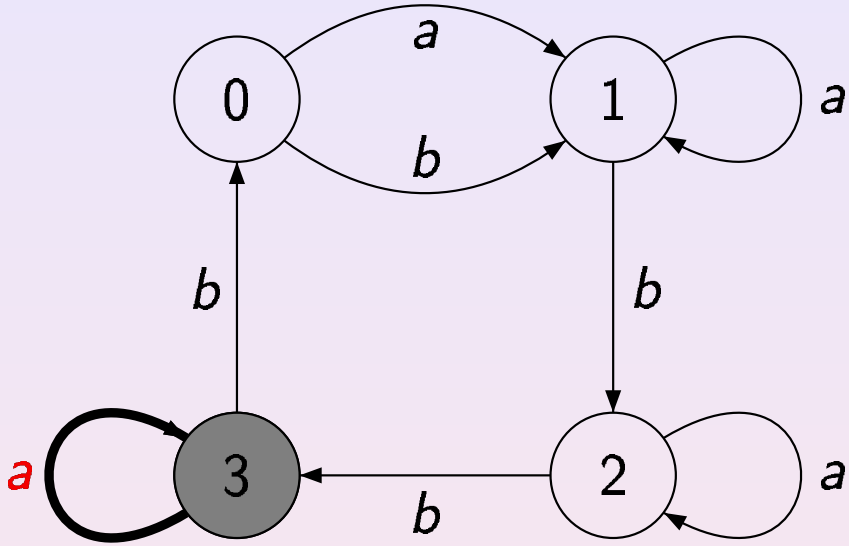
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

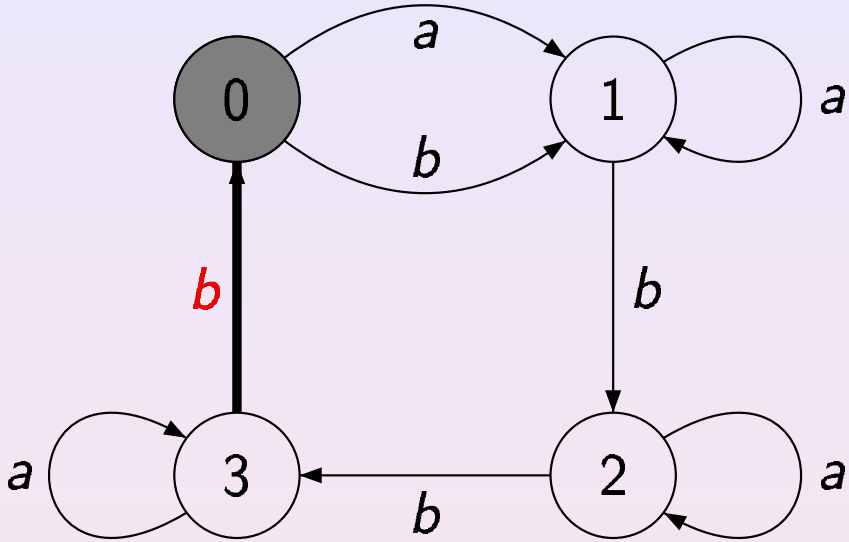
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

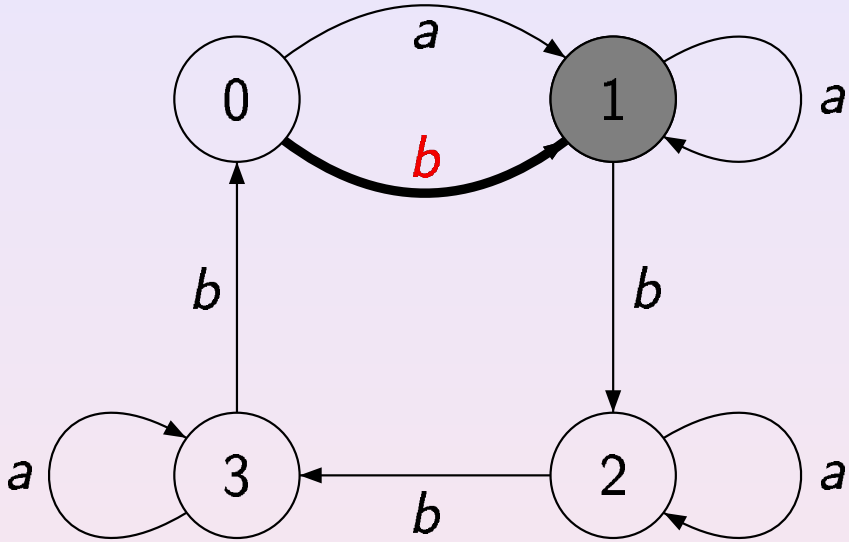
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

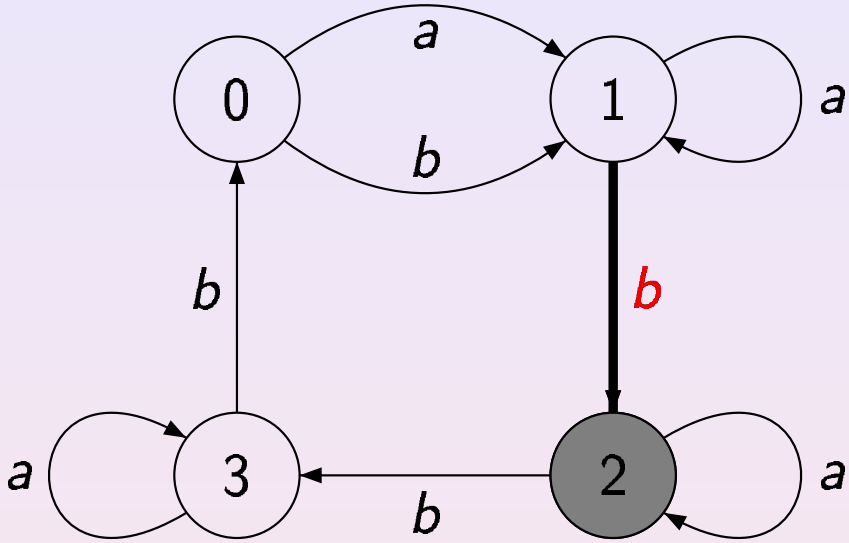
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

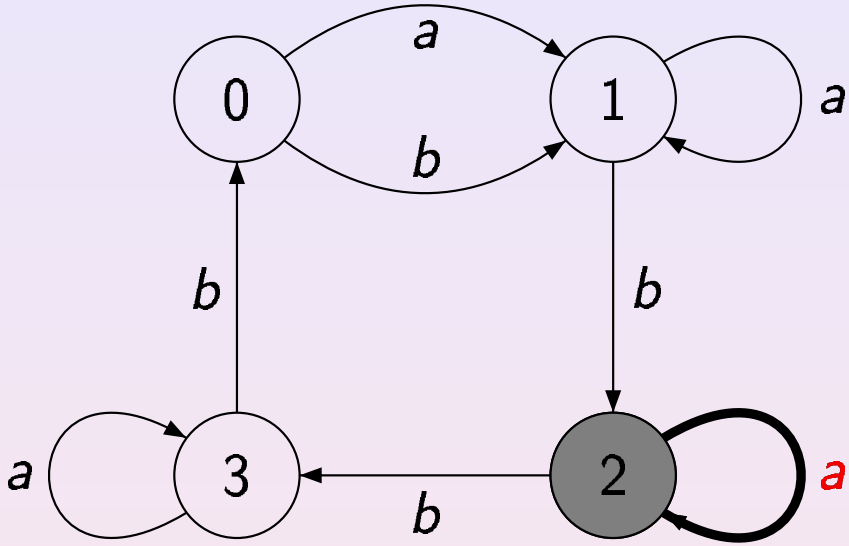
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

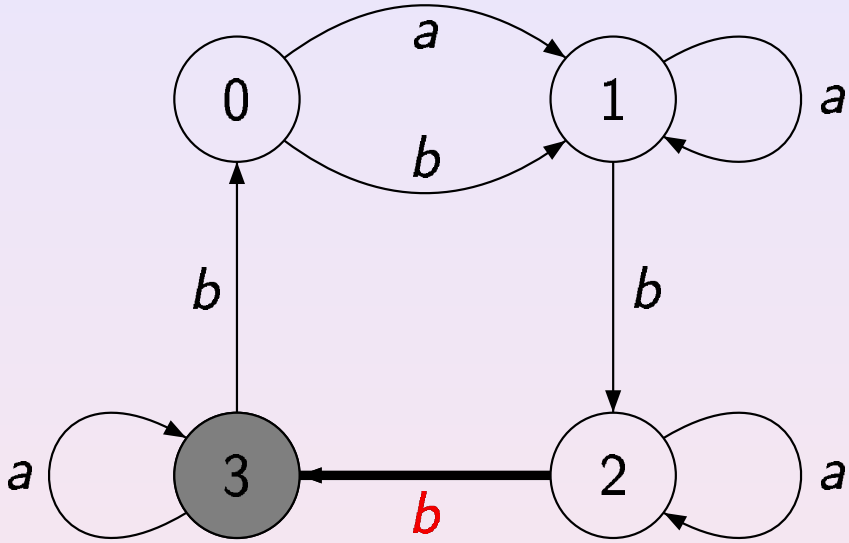
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

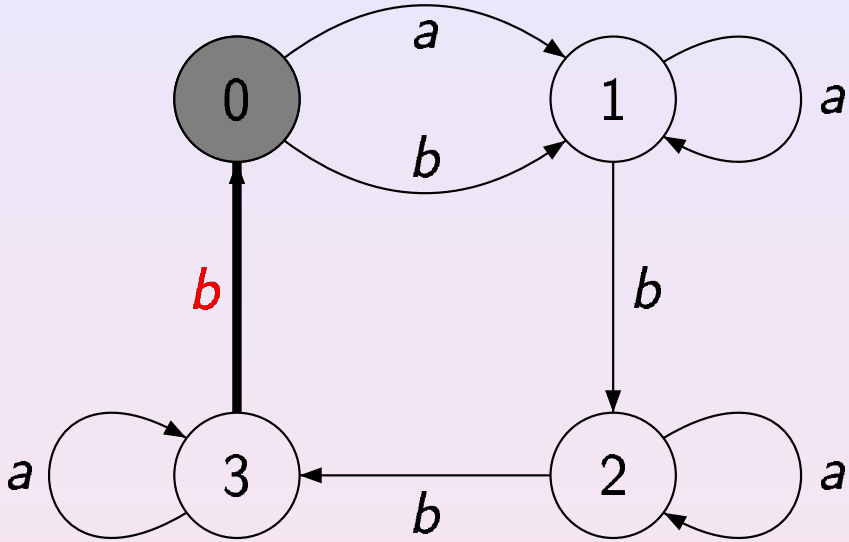
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

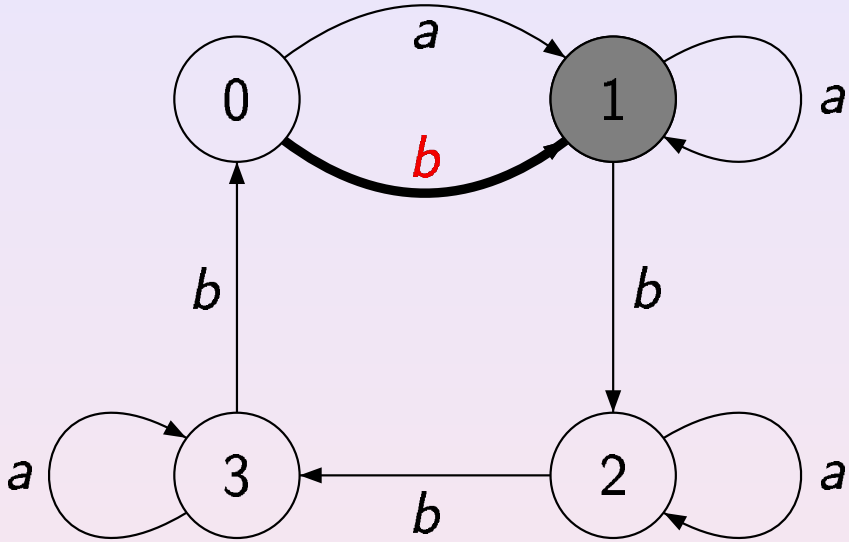
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

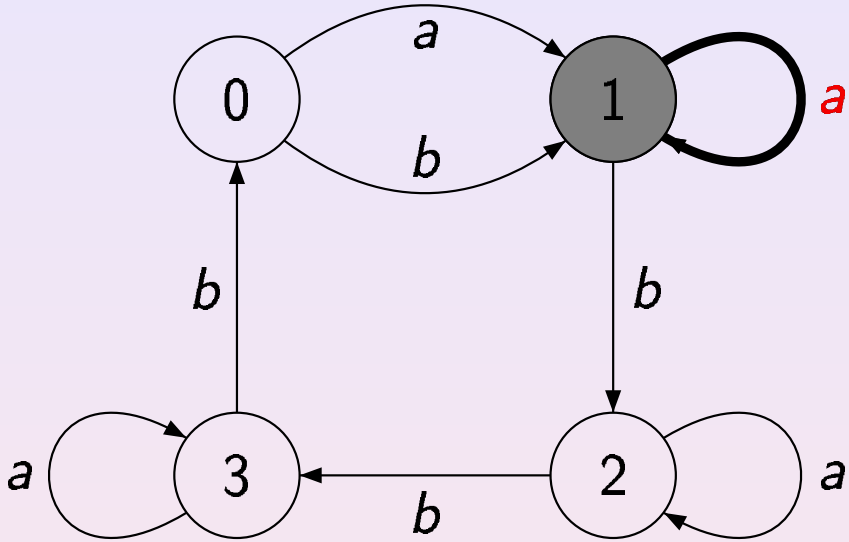
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

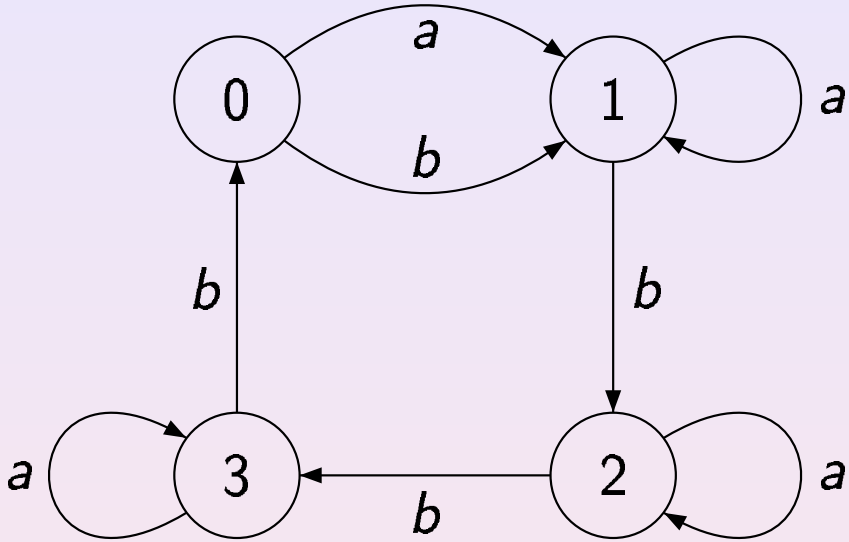
An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

An Example



A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

Černý's Paper

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

Černý's Paper

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

Černý's Paper

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

Černý's Paper

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

Černý's Paper

The notion was formalized in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied 14, no.3 (1964) 208–216 [in Slovak]) though implicitly it had been around since at least 1956.

The idea of synchronization is pretty natural and of obvious importance: we aim to restore control over a device whose current state is not known.

Think of a satellite which loops around the Moon and cannot be controlled from the Earth while "behind" the Moon (Černý's original motivation).

Independently, the same notion was discovered in coding theory by Shimon Even (Test for synchronizability of finite automata and variable length codes, IEEE Trans. Inform. Theory 10 (1964) 185–189). The name synchronizing seems to have originated from Even's paper.

An Algebraic Viewpoint

In algebraic terms, a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is merely a bunch Σ of transformations from $T(Q)$, the full transformation monoid on Q.

$\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing iff the subsemigroup of $T(Q)$ generated by Σ contains a rank 1 transformation.

An Algebraic Viewpoint

In algebraic terms, a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is merely a bunch Σ of transformations from $T(Q)$, the full transformation monoid on Q. $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing iff the subsemigroup of $T(Q)$ generated by Σ contains a rank 1 transformation.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$.

The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

$$\text{(Černý, 1964)} \quad (n-1)^2 \leq C(n)$$

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

$$\text{(Černý, 1964)} \quad (n-1)^2 \leq C(n) \leq \frac{n^3-n}{6} \quad \text{(Pin–Frankl, 1983).}$$

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Černý Conjecture

The Černý conjecture is the claim that every synchronizing automaton with n states possesses a reset word of length $(n-1)^2$. The validity of the conjecture is main open problem of the area and arguably one of the most long-standing open problems in combinatorial theory of finite automata.

Define the Černý function $C(n)$ as the maximum reset threshold for synchronizing automata with n states. In terms of this function, our current knowledge can be summarized in one line:

$$\text{(Černý, 1964)} \quad (n-1)^2 \leq C(n) \leq \frac{n^3-n}{6} \quad \text{(Pin–Frankl, 1983).}$$

The Černý conjecture thus claims that in fact $C(n) = (n-1)^2$.

Why so Difficult?

Why is the problem so surprisingly difficult?

One of the reasons: "slowly" synchronizing automata turn out to be extremely rare. Only one infinite series of n-state synchronizing automata with reset threshold $(n-1)^2$ is known (due to Černý, 1964), with a few (actually, 8) sporadic examples for $n \leq 6$.

Why so Difficult?

Why is the problem so surprisingly difficult?

One of the reasons: "slowly" synchronizing automata turn out to be extremely rare. Only one infinite series of n-state synchronizing automata with reset threshold $(n-1)^2$ is known (due to Černý, 1964), with a few (actually, 8) sporadic examples for $n \leq 6$.

Why so Difficult?

Why is the problem so surprisingly difficult?

One of the reasons: "slowly" synchronizing automata turn out to be extremely rare. Only one infinite series of n-state synchronizing automata with reset threshold $(n-1)^2$ is known (due to Černý, 1964), with a few (actually, 8) sporadic examples for $n \leq 6$.

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata.

The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata. The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata. The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

8 states:

Reset threshold	49	48	47	46	45	44	43	42	41	40
# of automata	1	0	0	0	0	1	1	3	1	5

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata. The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

8 states:

Reset threshold	49	48	47	46	45	44	43	42	41	40
# of automata	1	0	0	0	0	1	1	3	1	5

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata. The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

8 states:

Reset threshold	49	48	47	46	45	44	43	42	41	40
# of automata	1	0	0	0	0	1	1	3	1	5

9 states:

Reset threshold	64	63	62	61	60	59	58	57	56	55
# of automata	1	0	0	0	0	0	1	2	3	0

Reset threshold	54	53	52	51
# of automata	0	0	4	4

Lisbon, July 23rd, 2014

Episode IV: A New Hope

In 2009/10, Vladimir Gusev, at that time a PhD student of mine, has performed a massive series of experiments searching exhaustively through automata with a modest number of states in order to find new examples of "slowly" synchronizing automata. The next tables present the distribution of non-isomorphic synchronizing automata with 8 and 9 states and 2 letters with respect to their reset thresholds.

8 states:

Reset threshold	49	48	47	46	45	44	43	42	41	40
# of automata	1	0	0	0	0	1	1	3	1	5

9 states:

Reset threshold	64	63	62	61	60	59	58	57	56	55
# of automata	1	0	0	0	0	0	1	2	3	0

Reset threshold	54	53	52	51
# of automata	0	0	4	4

Lisbon, July 23rd, 2014

Advantage of Being Old

Thus, the pattern is:

$$(n-1)^2 \quad \text{the first gap} \quad \text{the "island"} \quad \text{the second gap}$$

The second gap first appears at 9 states and grows rather regularly with the number of states. The size of the island depends only on the parity of the number of states.

The very same pattern appears in the distribution of exponents of non-negative matrices.

Advantage of Being Old

Thus, the pattern is:

$$(n-1)^2 \quad \text{the first gap} \quad \text{the "island"} \quad \text{the second gap}$$

The second gap first appears at 9 states and grows rather regularly with the number of states. The size of the island depends only on the parity of the number of states.

The very same pattern appears in the distribution of exponents of non-negative matrices.

Advantage of Being Old

Thus, the pattern is:

$$(n-1)^2 \quad \text{the first gap} \quad \text{the "island"} \quad \text{the second gap}$$

The second gap first appears at 9 states and grows rather regularly with the number of states. The size of the island depends only on the parity of the number of states.
The very same pattern appears in the distribution of exponents of non-negative matrices.

Exponents of Non-negative Matrices

A non-negative matrix A is said to be primitive if some power A^k is positive. The minimum k with this property is called the exponent of A, denoted $\exp A$.

Helmut Wielandt proved in 1950 that for any primitive $n \times n$-matrix A, one has $\exp A \leq n^2 - 2n + 2 = (n-1)^2 + 1$, and this bound is tight. Possible exponents of $n \times n$-matrices were intensively studied in the 1960s, and it was discovered that two extreme values are each attained by a unique matrix, then there is a gap followed by an island followed by another gap. The sizes of the gaps and of the island perfectly match the sizes of the gaps and of the islands in possible reset thresholds of synchronizing automata with n states — basically one has the same picture shifted by 1. Clearly, this cannot be a mere coincidence.

Exponents of Non-negative Matrices

A non-negative matrix A is said to be primitive if some power A^k is positive. The minimum k with this property is called the exponent of A, denoted $\exp A$.

Helmut Wielandt proved in 1950 that for any primitive $n \times n$-matrix A, one has $\exp A \leq n^2 - 2n + 2 = (n-1)^2 + 1$, and this bound is tight. Possible exponents of $n \times n$-matrices were intensively studied in the 1960s, and it was discovered that two extreme values are each attained by a unique matrix, then there is a gap followed by an island followed by another gap. The sizes of the gaps and of the island perfectly match the sizes of the gaps and of the islands in possible reset thresholds of synchronizing automata with n states — basically one has the same picture shifted by 1. Clearly, this cannot be a mere coincidence.

Exponents of Non-negative Matrices

A non-negative matrix A is said to be primitive if some power A^k is positive. The minimum k with this property is called the exponent of A, denoted $\exp A$.

Helmut Wielandt proved in 1950 that for any primitive $n \times n$-matrix A, one has $\exp A \leq n^2 - 2n + 2 = (n-1)^2 + 1$, and this bound is tight. Possible exponents of $n \times n$-matrices were intensively studied in the 1960s, and it was discovered that two extreme values are each attained by a unique matrix, then there is a gap followed by an island followed by another gap. The sizes of the gaps and of the island perfectly match the sizes of the gaps and of the islands in possible reset thresholds of synchronizing automata with n states — basically one has the same picture shifted by 1. Clearly, this cannot be a mere coincidence.

Exponents of Non-negative Matrices

A non-negative matrix A is said to be primitive if some power A^k is positive. The minimum k with this property is called the exponent of A, denoted $\exp A$.

Helmut Wielandt proved in 1950 that for any primitive $n \times n$-matrix A, one has $\exp A \leq n^2 - 2n + 2 = (n-1)^2 + 1$, and this bound is tight. Possible exponents of $n \times n$-matrices were intensively studied in the 1960s, and it was discovered that two extreme values are each attained by a unique matrix, then there is a gap followed by an island followed by another gap. The sizes of the gaps and of the island perfectly match the sizes of the gaps and of the islands in possible reset thresholds of synchronizing automata with n states — basically one has the same picture shifted by 1. Clearly, this cannot be a mere coincidence.

Exponents of Non-negative Matrices

A non-negative matrix A is said to be primitive if some power A^k is positive. The minimum k with this property is called the exponent of A, denoted $\exp A$.

Helmut Wielandt proved in 1950 that for any primitive $n \times n$-matrix A, one has $\exp A \leq n^2 - 2n + 2 = (n-1)^2 + 1$, and this bound is tight. Possible exponents of $n \times n$-matrices were intensively studied in the 1960s, and it was discovered that two extreme values are each attained by a unique matrix, then there is a gap followed by an island followed by another gap. The sizes of the gaps and of the island perfectly match the sizes of the gaps and of the islands in possible reset thresholds of synchronizing automata with n states — basically one has the same picture shifted by 1. Clearly, this cannot be a mere coincidence.

Digraphs and Matrices

A directed graph (digraph) is a pair $D = \langle V, E \rangle$.

- V set of vertices
- $E \subseteq V \times V$ set of edges

This definition allows loops but excludes multiple edges.

The matrix of a digraph $D = \langle V, E \rangle$ is just the incidence matrix of the edge relation, that is, a $V \times V$-matrix whose entry in the row v and the column v' is 1 if $(v, v') \in E$ and 0 otherwise.

Digraphs and Matrices

A directed graph (digraph) is a pair $D = \langle V, E \rangle$.

- V set of vertices
- $E \subseteq V \times V$ set of edges

This definition allows loops but excludes multiple edges.

The matrix of a digraph $D = \langle V, E \rangle$ is just the incidence matrix of the edge relation, that is, a $V \times V$-matrix whose entry in the row v and the column v' is 1 if $(v, v') \in E$ and 0 otherwise.

Digraphs and Matrices

A directed graph (digraph) is a pair $D = \langle V, E \rangle$.

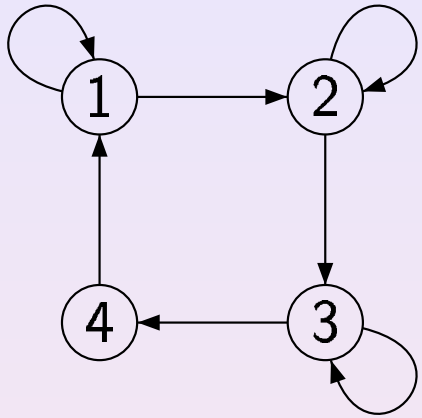
- V set of vertices
- $E \subseteq V \times V$ set of edges

This definition allows loops but excludes multiple edges.

The matrix of a digraph $D = \langle V, E \rangle$ is just the incidence matrix of the edge relation, that is, a $V \times V$-matrix whose entry in the row v and the column v' is 1 if $(v, v') \in E$ and 0 otherwise.

Digraphs and Matrices

For instance, the matrix of the digraph



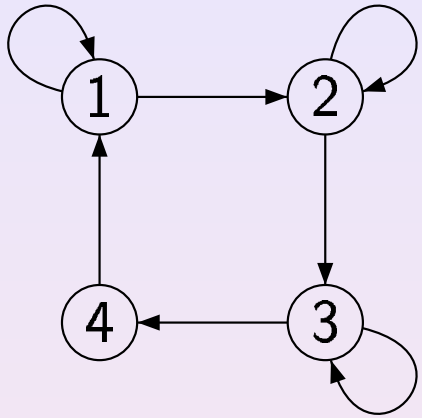
(with respect to the chosen numbering of its vertices) is $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$.

Conversely, given an $n \times n$-matrix $P = (p_{ij})$ with non-negative real entries, we assign to it a digraph $D(P)$ on the set $\{1, 2, \ldots, n\}$ as follows: (i, j) is an edge of $D(P)$ if and only if $p_{ij} > 0$.

This 'two-way' correspondence allows us to reformulate in terms of digraphs several important notions and results which originated in the classical Perron–Frobenius theory of non-negative matrices.

Digraphs and Matrices

For instance, the matrix of the digraph

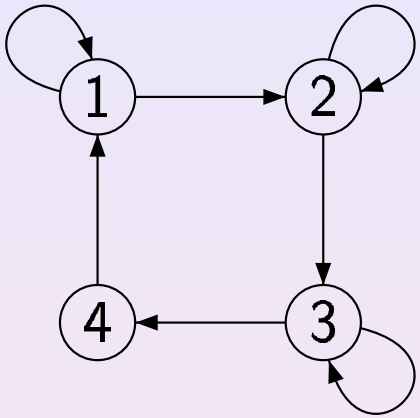


(with respect to the chosen numbering of its vertices) is $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$.

Conversely, given an $n \times n$-matrix $P = (p_{ij})$ with non-negative real entries, we assign to it a digraph $D(P)$ on the set $\{1, 2, \ldots, n\}$ as follows: (i, j) is an edge of $D(P)$ if and only if $p_{ij} > 0$.

This 'two-way' correspondence allows us to reformulate in terms of digraphs several important notions and results which originated in the classical Perron–Frobenius theory of non-negative matrices.

For instance, the matrix of the digraph



(with respect to the chosen numbering of its vertices) is $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$.

Conversely, given an $n \times n$-matrix $P = (p_{ij})$ with non-negative real entries, we assign to it a digraph $D(P)$ on the set $\{1, 2, \ldots, n\}$ as follows: (i, j) is an edge of $D(P)$ if and only if $p_{ij} > 0$.

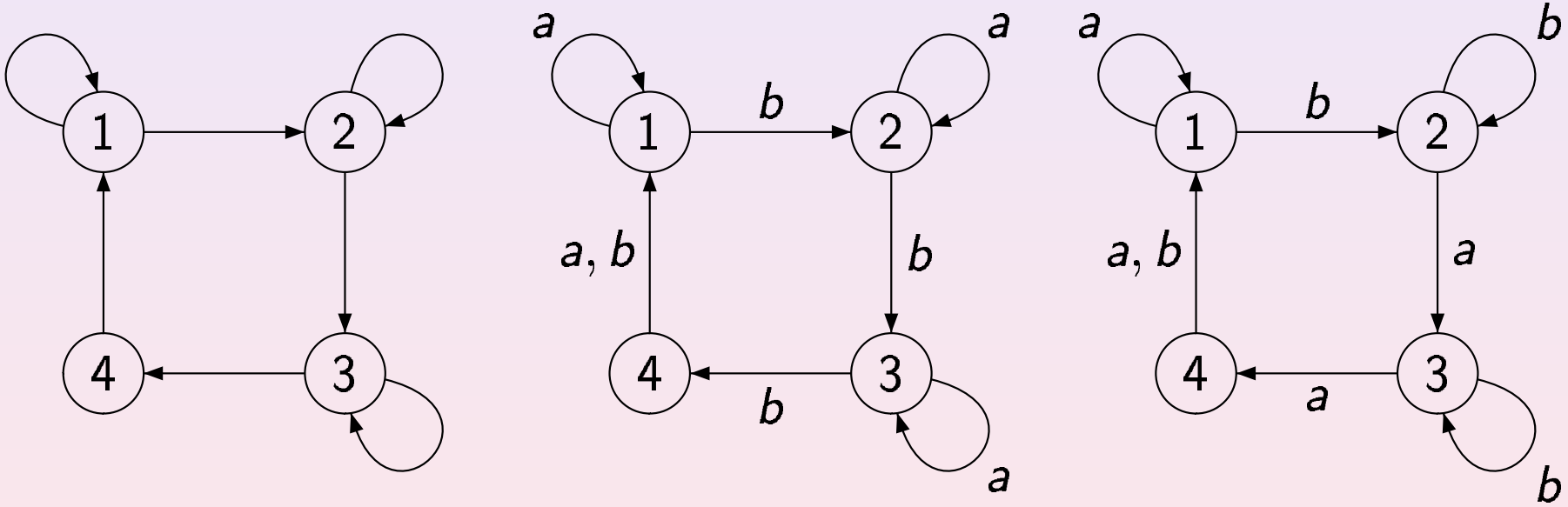
This 'two-way' correspondence allows us to reformulate in terms of digraphs several important notions and results which originated in the classical Perron–Frobenius theory of non-negative matrices.

Digraphs and Colorings

By a coloring of a digraph we mean assigning labels from an alphabet Σ to edges such that the digraph labeled this way becomes a DFA.

Digraphs and Colorings

By a coloring of a digraph we mean assigning labels from an alphabet Σ to edges such that the digraph labeled this way becomes a DFA.



Primitive Digraphs

A digraph D is primitive if D is strongly connected and the greatest common divisor of the lengths of all cycles in D is equal to 1.

Adler, Goodwyn, Weiss (Equivalence of topological Markov shifts, Israel J. Math. 27 (1977) 49–63):
Underlying digraphs of strongly connected synchronizing automata are primitive.

The Road Coloring Conjecture: Every primitive digraph admits a synchronizing coloring.

This was confirmed by Avraham Trahtman (The Road Coloring Problem, Israel J. Math. 172 (2009) 51–60).

Primitive Digraphs

A digraph D is primitive if D is strongly connected and the greatest common divisor of the lengths of all cycles in D is equal to 1.

Adler, Goodwyn, Weiss (Equivalence of topological Markov shifts, Israel J. Math. 27 (1977) 49–63):
Underlying digraphs of strongly connected synchronizing automata are primitive.

The Road Coloring Conjecture: Every primitive digraph admits a synchronizing coloring.

This was confirmed by Avraham Trahtman (The Road Coloring Problem, Israel J. Math. 172 (2009) 51–60).

Primitive Digraphs

A digraph D is primitive if D is strongly connected and the greatest common divisor of the lengths of all cycles in D is equal to 1.

Adler, Goodwyn, Weiss (Equivalence of topological Markov shifts, Israel J. Math. 27 (1977) 49–63):
Underlying digraphs of strongly connected synchronizing automata are primitive.

The Road Coloring Conjecture: Every primitive digraph admits a synchronizing coloring.

This was confirmed by Avraham Trahtman (The Road Coloring Problem, Israel J. Math. 172 (2009) 51–60).

Primitive Digraphs

A digraph D is primitive if D is strongly connected and the greatest common divisor of the lengths of all cycles in D is equal to 1.

Adler, Goodwyn, Weiss (Equivalence of topological Markov shifts, Israel J. Math. 27 (1977) 49–63):
Underlying digraphs of strongly connected synchronizing automata are primitive.

The Road Coloring Conjecture: Every primitive digraph admits a synchronizing coloring.

This was confirmed by Avraham Trahtman (The Road Coloring Problem, Israel J. Math. 172 (2009) 51–60).

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.
If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.
If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.
If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.
If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.
If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.
If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.
If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.
If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.

If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.

If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents

A digraph D is primitive iff there exists t such that for each pair of vertices there exists a path between them of length exactly t. (This is equivalent to saying that the t-th power of the matrix of D is positive.) The least t with this property is called the exponent of the digraph D and is denoted by $\exp(D)$.

1950, Wielandt: The exponent of every primitive digraph on n vertices is not greater than $(n-1)^2+1$ and this bound is tight.

1964, Dulmage–Mendelsohn: There is exactly one primitive digraph on n vertices with exponent $(n-1)^2+1$ and exactly one primitive digraph on n vertices with exponent $(n-1)^2$.
If $n>4$ is even, then there is no primitive digraph D on n vertices such that $n^2-4n+6<\exp(D)<(n-1)^2$.
If $n>3$ is odd, then there is no primitive digraph D on n vertices such that $n^2-3n+4<\exp(D)<(n-1)^2$, or $n^2-4n+6<\exp(D)<n^2-3n+2$.

Exponents vs Reset Lengths

Exponents of primitive digraphs with 9 vertices vs reset thresholds of 2-letter strongly connected synchronizing automata with 9 states

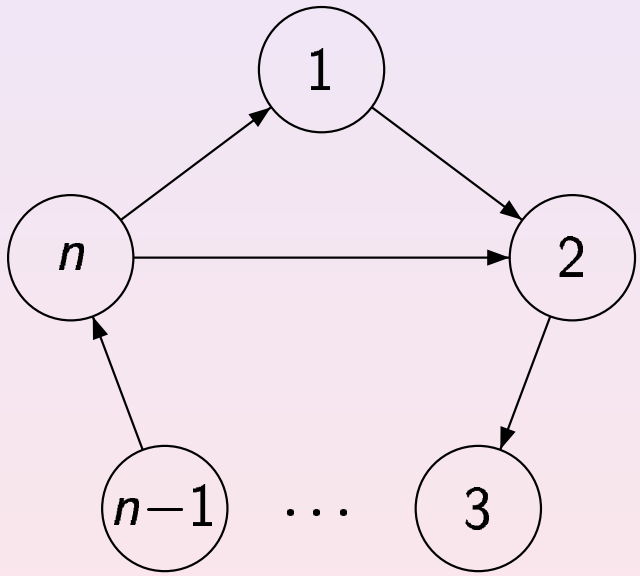
N	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51
# of primitive digraphs with exponent N	1	1	0	0	0	0	0	1	1	2	0	0	0	0	3
# of 2-letter synchronizing automata with reset threshold N	0	1	0	0	0	0	0	1	2	3	0	0	0	4	4

The Wielandt Automaton

The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\exp(W_n) = (n-1)^2 + 1$. The Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.

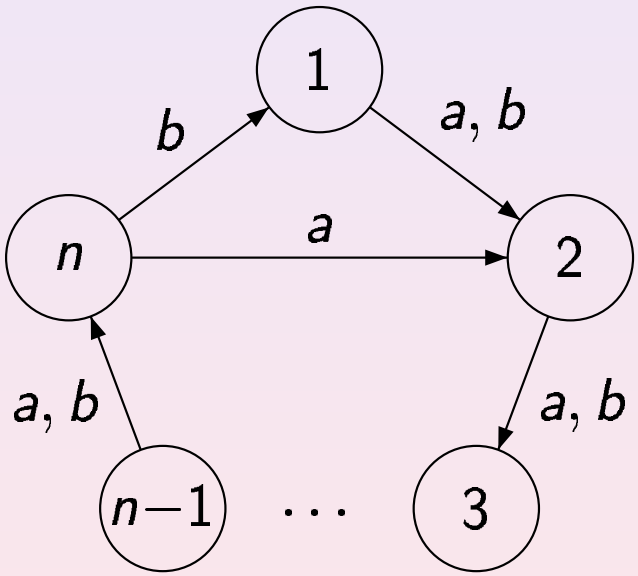
The Wielandt Automaton

The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\exp(W_n) = (n-1)^2 + 1$. The Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



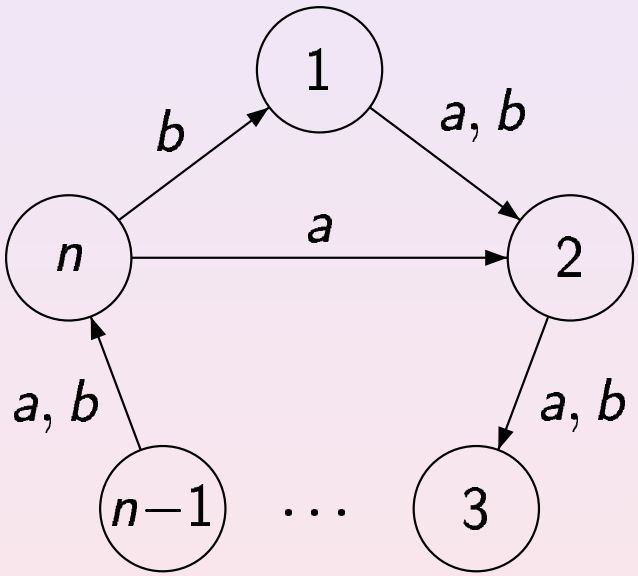
The Wielandt Automaton

The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\exp(W_n) = (n-1)^2 + 1$. The Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



The Wielandt Automaton

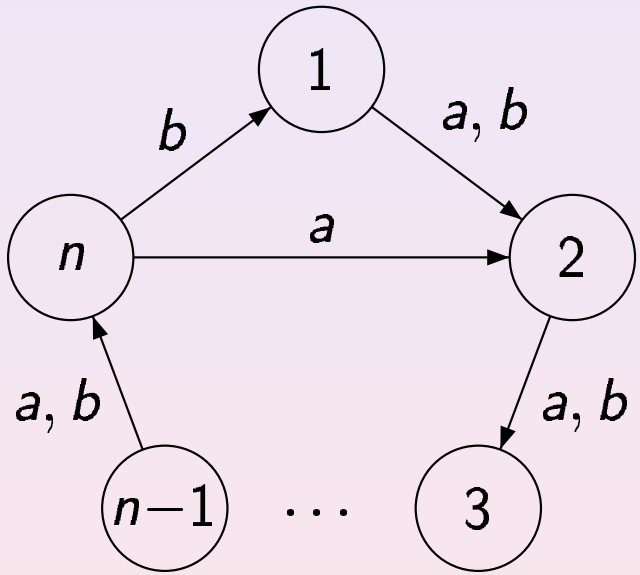
The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\exp(W_n) = (n-1)^2 + 1$. The Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



It is easy to show that the reset threshold of $\mathscr{W}_n$ is $n^2 - 3n + 3$.

The Wielandt Automaton

The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\exp(W_n) = (n-1)^2 + 1$. The Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



It is easy to show that the reset threshold of $\mathscr{W}_n$ is $n^2 - 3n + 3$.

In a similar way, each digraph with large exponent generates slowly synchronizing automata.

Lisbon, July 23rd, 2014

Colorings of Digraphs with Large Exponents

Observation

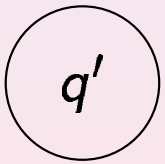
Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$

Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

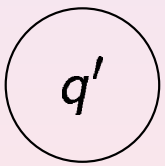
$$\exp(D) \leq t + n - 1.$$

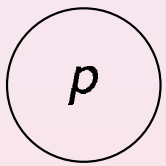


Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$



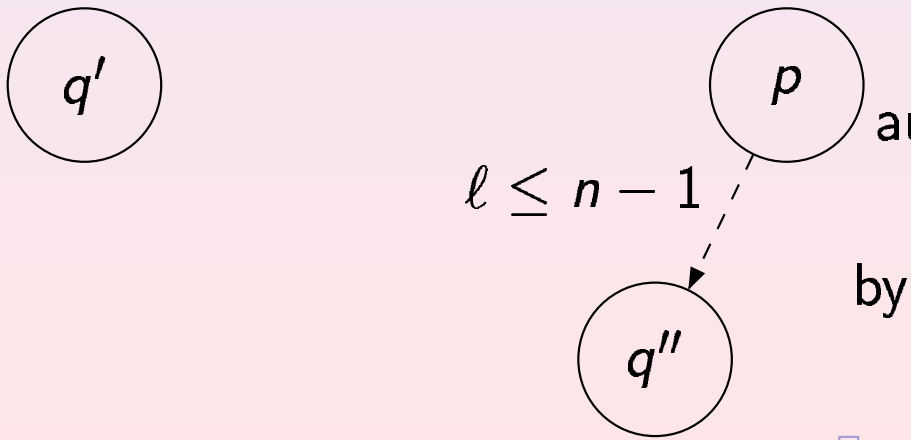


the state to which our automaton is reset by a word of length t

Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$



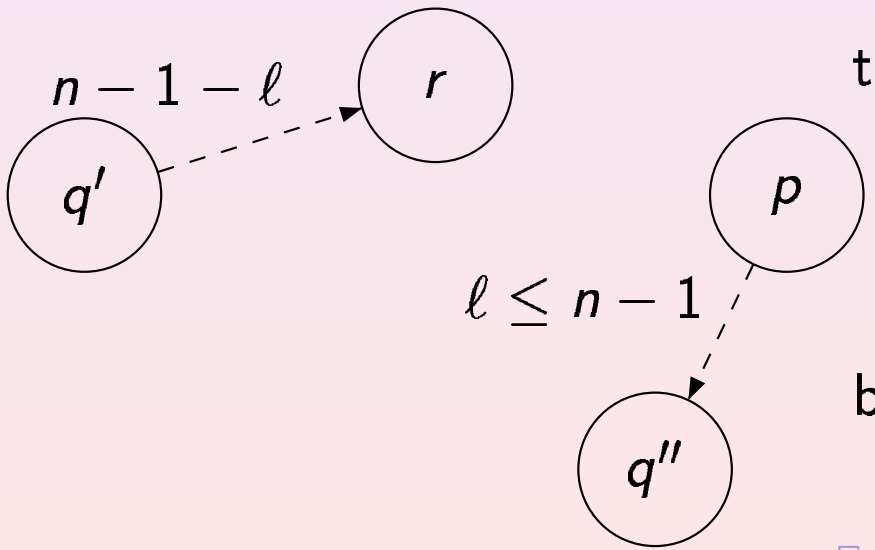
the state to which our automaton is reset by a word of length t

Lisbon, July 23rd, 2014

Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$



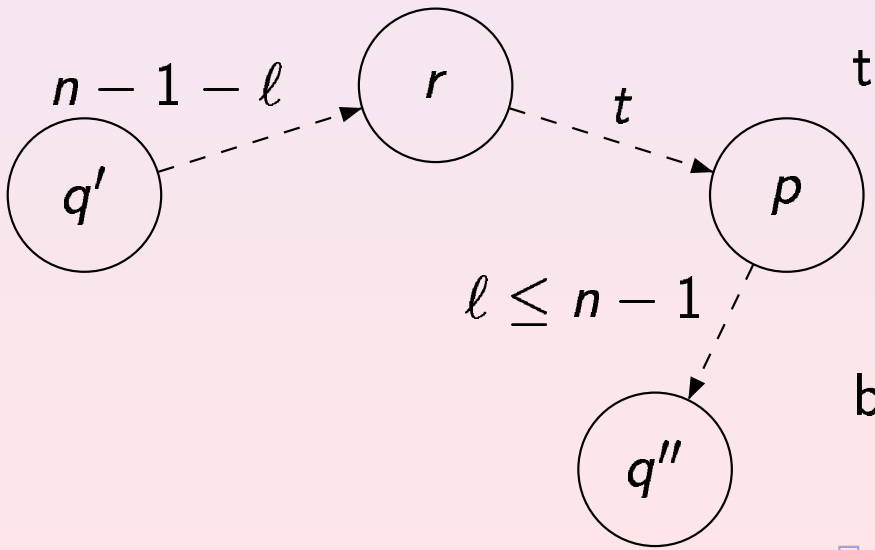
the state to which our automaton is reset by a word of length t

Lisbon, July 23rd, 2014

Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$



the state to which our automaton is reset by a word of length t

Lisbon, July 23rd, 2014

Observation

Let a strongly connected synchronizing automaton with n states and reset threshold t be a coloring of a digraph D. Then

$$\exp(D) \leq t + n - 1.$$

For instance, the reset threshold t of the Wielandt automaton $\mathscr{W}_n$ must satisfy

$$t \geq \exp(W_n) - n + 1 = (n-1)^2 + 1 - n + 1 = n^2 - 3n + 3,$$

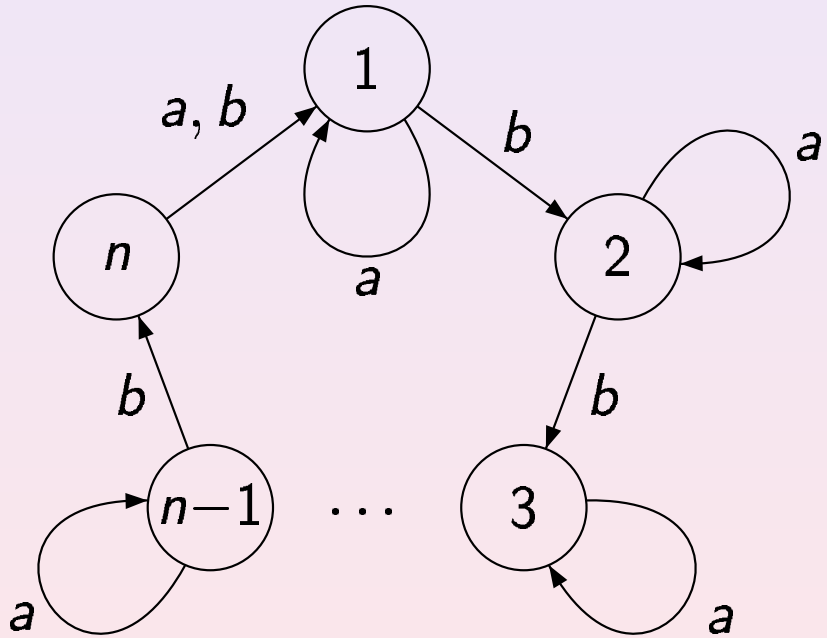
and it is easy to find a reset word of length $n^2 - 3n + 3$.

The Černý Automaton

There are slowly synchronizing automata that cannot be obtained as colorings of a digraph with large exponent. For instance, the Černý automaton $\mathscr{C}_n$ has reset threshold $(n-1)^2$ while its underlying digraph has exponent $n-1$.

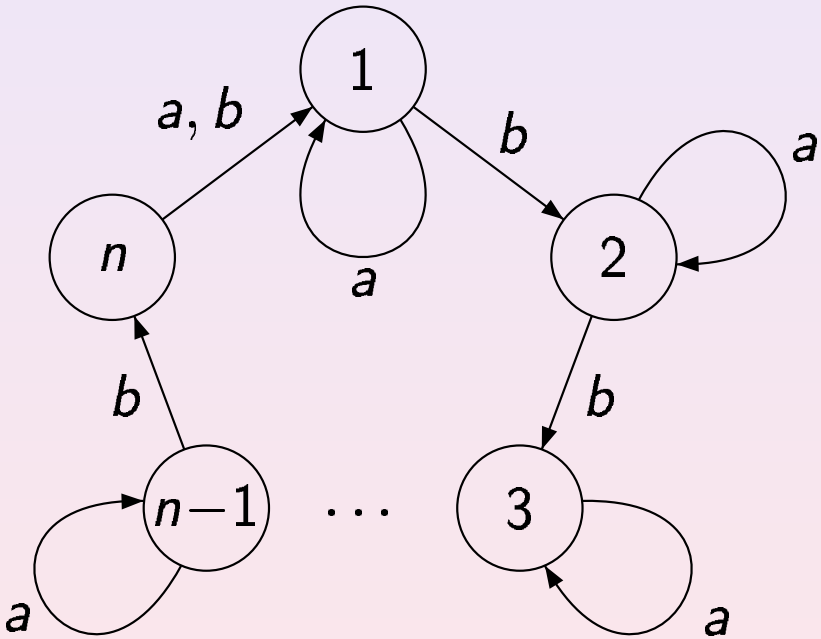
The Černý Automaton

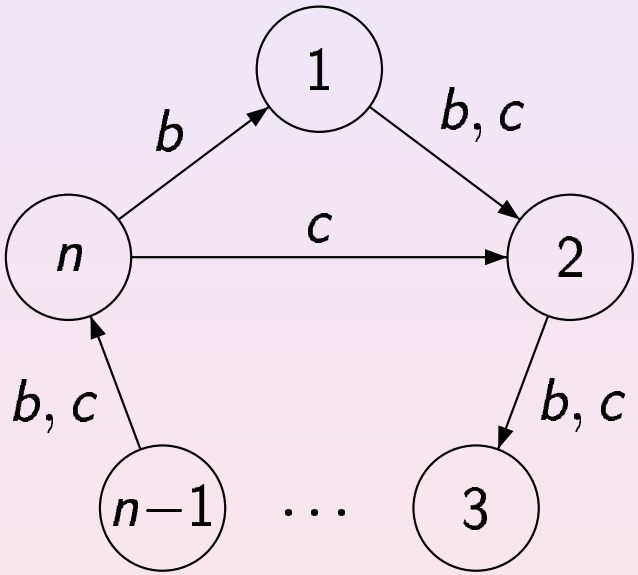
There are slowly synchronizing automata that cannot be obtained as colorings of a digraph with large exponent. For instance, the Černý automaton $\mathscr{C}_n$ has reset threshold $(n-1)^2$ while its underlying digraph has exponent $n-1$.



The Černý Automaton

There are slowly synchronizing automata that cannot be obtained as colorings of a digraph with large exponent. For instance, the Černý automaton $\mathscr{C}_n$ has reset threshold $(n-1)^2$ while its underlying digraph has exponent $n-1$.





However, $\mathscr{C}_n$ becomes $\mathscr{W}_n$ under the action of b and $c = ab$.

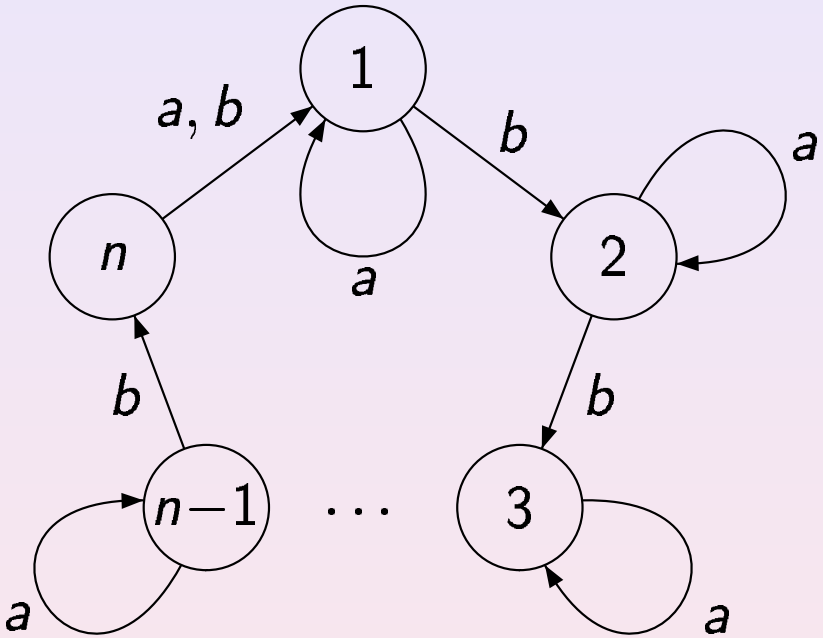
Lisbon, July 23rd, 2014

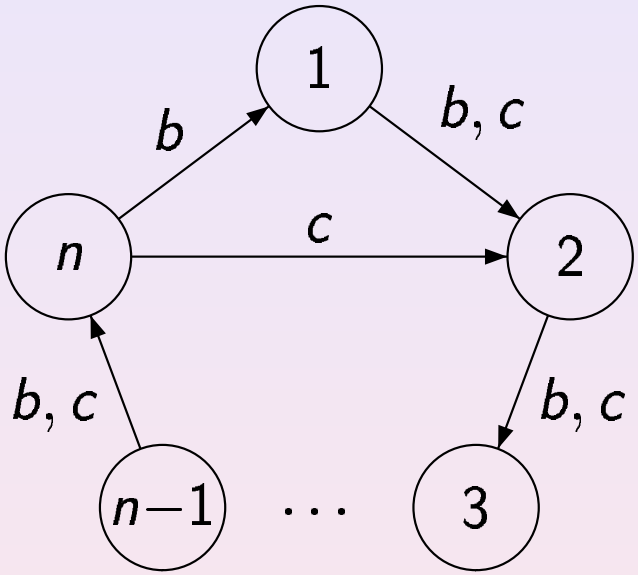
The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b. Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.

The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b.

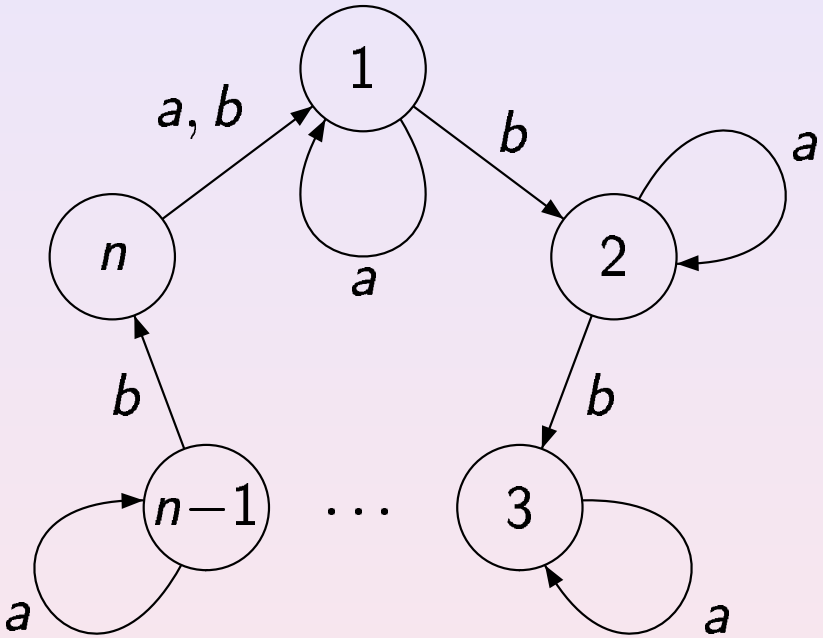


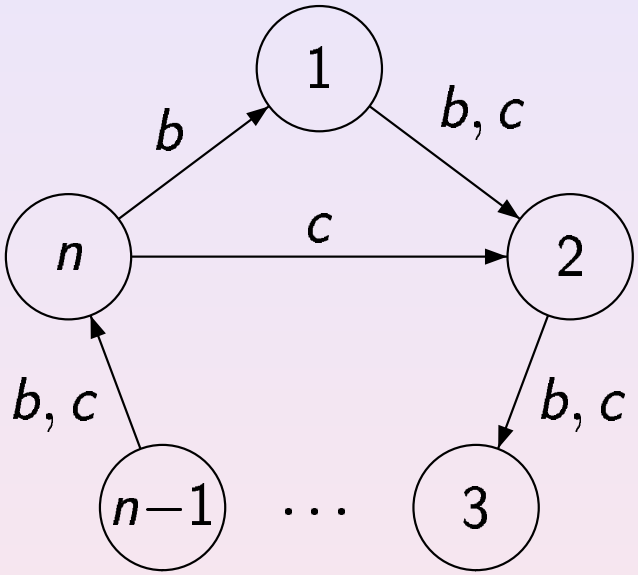


Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.

The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b.



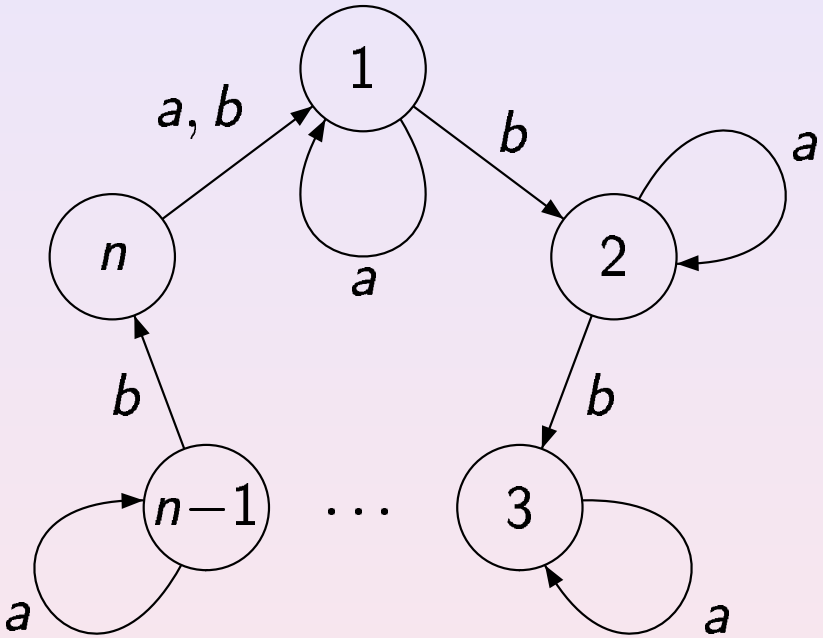


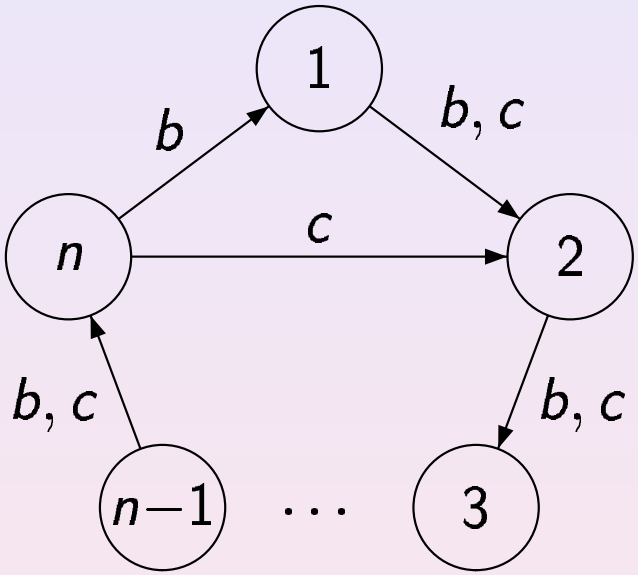
Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.

Lisbon, July 23rd, 2014

The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b.



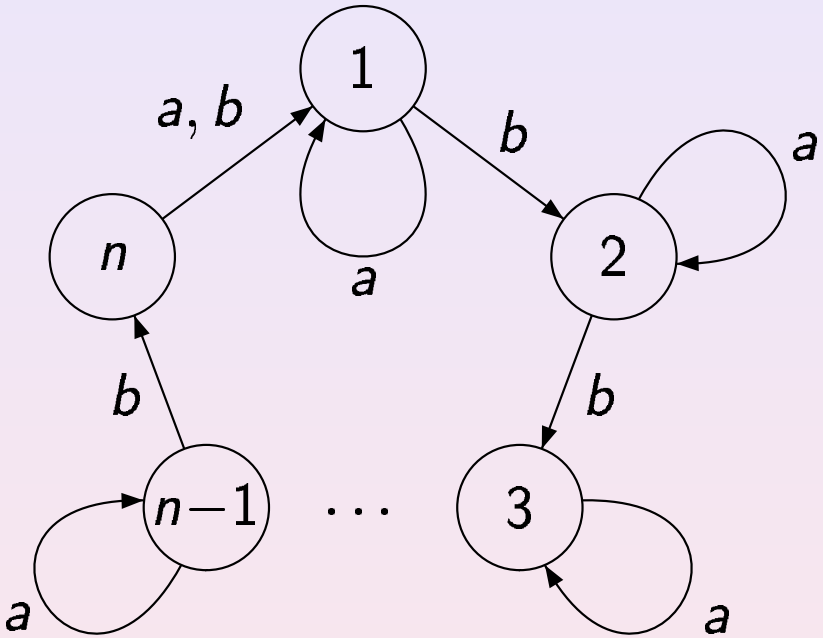


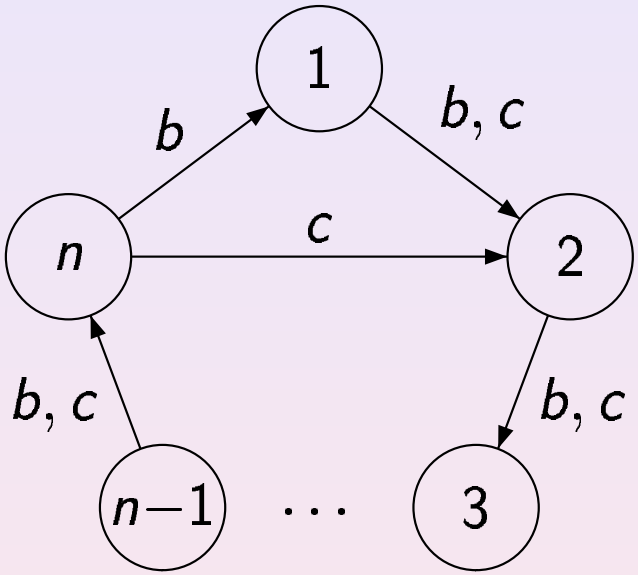
Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.

Lisbon, July 23rd, 2014

The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b.





Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.

Lisbon, July 23rd, 2014

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b. Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.
Further, v contains at least $n - 2$ occurrences of c. Since each occurrence of c in v corresponds to an occurrence of ab in w', we conclude that $|w'| \geq n^2 - 3n + 2 + n - 2 = n^2 - 2n$.

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b. Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.
Further, v contains at least $n - 2$ occurrences of c. Since each occurrence of c in v corresponds to an occurrence of ab in w', we conclude that $|w'| \geq n^2 - 3n + 2 + n - 2 = n^2 - 2n$. Thus, $|w| = |w'a| \geq n^2 - 2n + 1 = (n-1)^2$.

The Černý Automaton

Let w be a shortest reset word for $\mathscr{C}_n$. It must end with a and every other occurrence of a in w is followed by an occurrence of b. Thus, $w = w'a$ where w' can be rewritten into a word v over the alphabet $\{b, c\}$. Since w' and v act in the same way, the word vc is a reset word for $\mathscr{W}_n$. Hence $|v| \geq n^2 - 3n + 2$.
Further, v contains at least $n - 2$ occurrences of c. Since each occurrence of c in v corresponds to an occurrence of ab in w', we conclude that $|w'| \geq n^2 - 3n + 2 + n - 2 = n^2 - 2n$. Thus, $|w| = |w'a| \geq n^2 - 2n + 1 = (n-1)^2$.

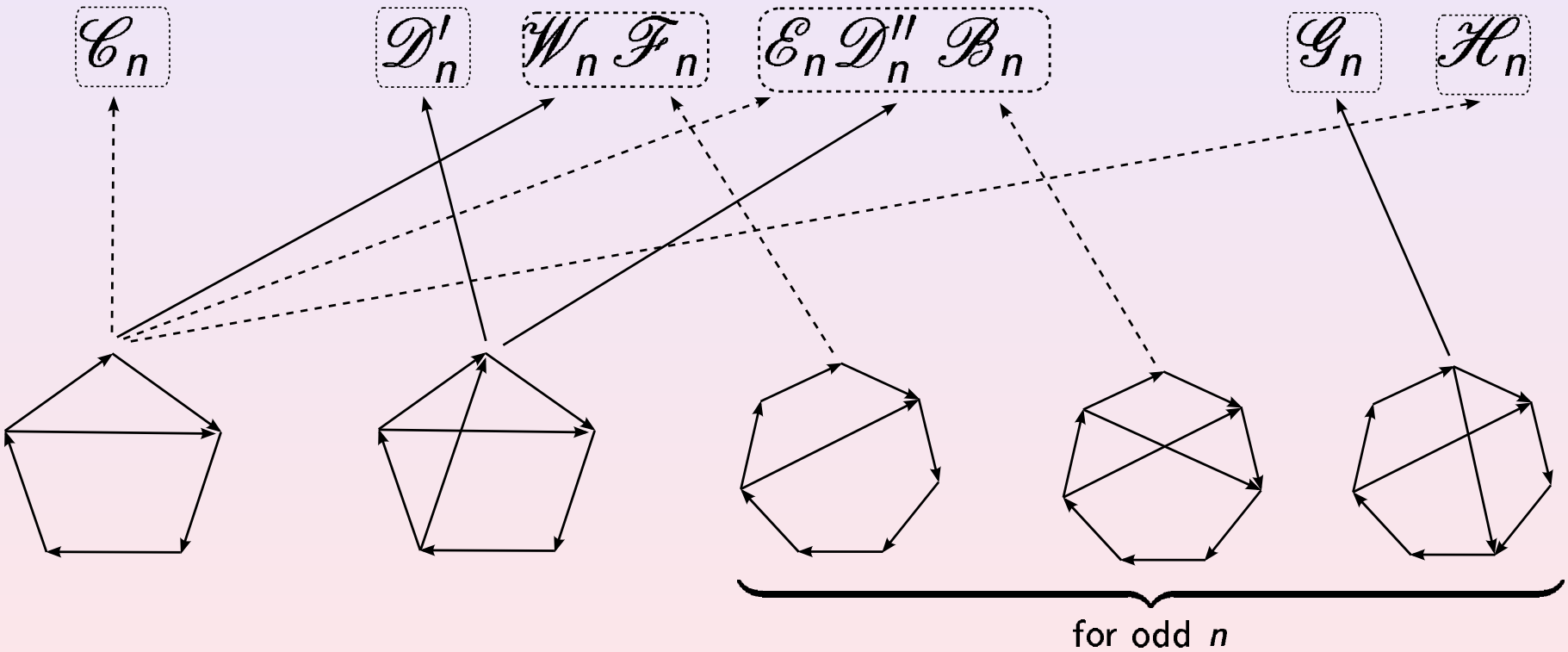
Thus, it is the Wielandt digraph that stays behind the Černý automaton!

Digraphs vs Automata

In a similar manner it is easy to recover every known slowly synchronizing automaton from a suitable digraph with large exponent.

Digraphs vs Automata

In a similar manner it is easy to recover every known slowly synchronizing automaton from a suitable digraph with large exponent.



More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \cdot w$.

More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \cdot w$.

More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \,.\, w$.

More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \,.\, w$.

More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \,.\, w$.

More on Experiments

Specifying a 9-automaton with two input letters is equivalent to specifying a pair of functions on a 9-element set. There are $9^{18} \approx 1.50 \times 10^{17}$ such pairs, and if one spends one nanosecond for calculating the reset threshold of each automaton defined this way, the exhaustive search takes around five years.

Clearly, if an n-automaton with k input letters is specified by a k-tuple of functions on an n-element set, each particular automaton is generated $n!k!$ times. However it is not possible to speed up the search by screening out isomorphic automata—even for $n = 9$ and $k = 2$ neither time nor memory would suffice to check whether the current automaton is isomorphic to one of the previously generated automata.

We used a string representation of initially-connected automata suggested by Almeida, Moreira, and Reis. A DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ is initially-connected if it has a state q_0 from which one can reach any state: for every $q \in Q$ there is $w \in \Sigma^*$ with $q = q_0 \,.\, w$.

String Representation

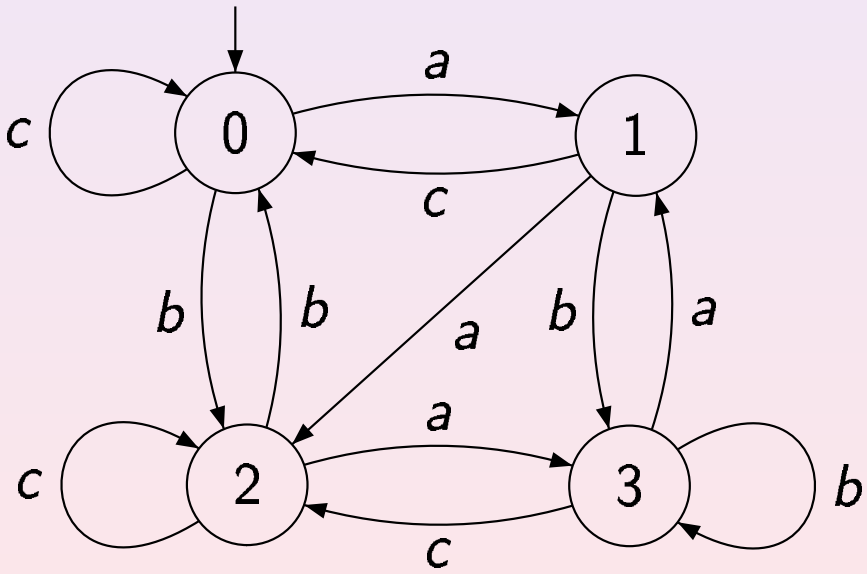
In general, a synchronizing automaton may fail to be strongly connected or initially-connected. However one can restrict to strongly connected automata when dealing with issues related to the Černý conjecture.

String Representation

In general, a synchronizing automaton may fail to be strongly connected or initially-connected. However one can restrict to strongly connected automata when dealing with issues related to the Černý conjecture.

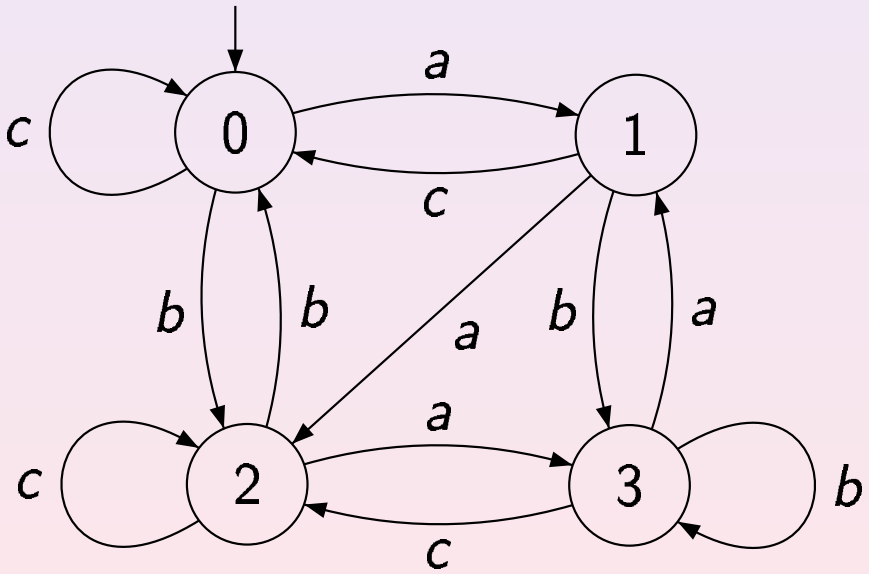
String Representation

In general, a synchronizing automaton may fail to be strongly connected or initially-connected. However one can restrict to strongly connected automata when dealing with issues related to the Černý conjecture.
Suppose that the input letters of the DFA in the picture are ordered $a < b < c$ and the state 0 is chosen as q_0.



String Representation

In general, a synchronizing automaton may fail to be strongly connected or initially-connected. However one can restrict to strongly connected automata when dealing with issues related to the Černý conjecture.
Suppose that the input letters of the DFA in the picture are ordered $a < b < c$ and the state 0 is chosen as q_0.



Then the canonical string is $[1, 2, 0, 2, 3, 0, 3, 0, 2, 1, 3, 2]$.

Lisbon, July 23rd, 2014

Experiments

It is easy to characterize strings $[s_0, \ldots, s_{nk-1}]$ of numbers from the set $\{0, 1, \ldots, n-1\}$ that may appear as canonical strings of initially-connected n-automata with k input letters.

We used a 128-core grid of AMD Opteron 2.6 GHz processors; it runs under Linux, has 256 Gb of memory and the peak performance of 665.6 GFLOPS. One node of the gird generated relatively small portions of canonical strings and sent them to other nodes that worked on their portions of automata in parallel. The management program was written in C with MPI.

Presenting automata via their canonical strings drastically reduces the exhaustive search. (It is easy to see, for instance, that every n-automaton with 2 input letters is generated at most $2n$ times if presented this way.) For $n = 9$ the number of automata to be analyzed was 705 068 085 303. However, thanks to parallelization the computation for $n = 9$ took less than 24 hours.

Experiments

It is easy to characterize strings $[s_0, \ldots, s_{nk-1}]$ of numbers from the set $\{0, 1, \ldots, n-1\}$ that may appear as canonical strings of initially-connected n-automata with k input letters.
We used a 128-core grid of AMD Opteron 2.6 GHz processors; it runs under Linux, has 256 Gb of memory and the peak performance of 665.6 GFLOPS. One node of the gird generated relatively small portions of canonical strings and sent them to other nodes that worked on their portions of automata in parallel. The management program was written in C with MPI.
Presenting automata via their canonical strings drastically reduces the exhaustive search. (It is easy to see, for instance, that every n-automaton with 2 input letters is generated at most $2n$ times if presented this way.) For $n = 9$ the number of automata to be analyzed was 705 068 085 303. However, thanks to parallelization the computation for $n = 9$ took less than 24 hours.

Experiments

It is easy to characterize strings $[s_0, \ldots, s_{nk-1}]$ of numbers from the set $\{0, 1, \ldots, n-1\}$ that may appear as canonical strings of initially-connected n-automata with k input letters.
We used a 128-core grid of AMD Opteron 2.6 GHz processors; it runs under Linux, has 256 Gb of memory and the peak performance of 665.6 GFLOPS. One node of the gird generated relatively small portions of canonical strings and sent them to other nodes that worked on their portions of automata in parallel. The management program was written in C with MPI.
Presenting automata via their canonical strings drastically reduces the exhaustive search. (It is easy to see, for instance, that every n-automaton with 2 input letters is generated at most $2n$ times if presented this way.) For $n = 9$ the number of automata to be analyzed was 705 068 085 303. However, thanks to parallelization the computation for $n = 9$ took less than 24 hours.

Experiments

It is easy to characterize strings $[s_0, \ldots, s_{nk-1}]$ of numbers from the set $\{0, 1, \ldots, n-1\}$ that may appear as canonical strings of initially-connected n-automata with k input letters.
We used a 128-core grid of AMD Opteron 2.6 GHz processors; it runs under Linux, has 256 Gb of memory and the peak performance of 665.6 GFLOPS. One node of the gird generated relatively small portions of canonical strings and sent them to other nodes that worked on their portions of automata in parallel. The management program was written in C with MPI.
Presenting automata via their canonical strings drastically reduces the exhaustive search. (It is easy to see, for instance, that every n-automaton with 2 input letters is generated at most $2n$ times if presented this way.) For $n = 9$ the number of automata to be analyzed was 705 068 085 303. However, thanks to parallelization the computation for $n = 9$ took less than 24 hours.

Experiments

It is easy to characterize strings $[s_0, \ldots, s_{nk-1}]$ of numbers from the set $\{0, 1, \ldots, n-1\}$ that may appear as canonical strings of initially-connected n-automata with k input letters.
We used a 128-core grid of AMD Opteron 2.6 GHz processors; it runs under Linux, has 256 Gb of memory and the peak performance of 665.6 GFLOPS. One node of the gird generated relatively small portions of canonical strings and sent them to other nodes that worked on their portions of automata in parallel. The management program was written in C with MPI.
Presenting automata via their canonical strings drastically reduces the exhaustive search. (It is easy to see, for instance, that every n-automaton with 2 input letters is generated at most $2n$ times if presented this way.) For $n = 9$ the number of automata to be analyzed was 705 068 085 303. However, thanks to parallelization the computation for $n = 9$ took less than 24 hours.

Limits and Prospects

The number of initially-connected DFA with 2 and 3 input letters

state #	7	8	9
2 letters	256 182 290	12 665 445 248	705 068 085 303
3 letters	500 750 172 337 212	572 879 126 392 178 688	835 007 874 759 393 878 655