Exponent Sequences of Labeled Digraphs vs Reset Thresholds of Synchronizing Automata

Mikhail Volkov
(based on a joint work with Dmitry Ananichev and Vladimir Gusev)

Ural Federal University, Ekaterinburg, Russia

RuFiDim, St Petersburg, September 22, 2011

Definitions and Terminology

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.

$\mathscr{A}$ 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) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q\,.\,w$ for $\delta(q, w)$ and $P\,.\,w$ for $\{\delta(q, w) \mid q \in P\}$.

$\mathscr{A}$ 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.

We consider complete deterministic finite automata (DFA) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.

$\mathscr{A}$ 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) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.

$\mathscr{A}$ 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) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.
$\mathscr{A}$ 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) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.

$\mathscr{A}$ 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) $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ where Q stands for the state set, Σ is the input alphabet, and $\delta : Q \times \Sigma \to Q$ is a (total) transition function.

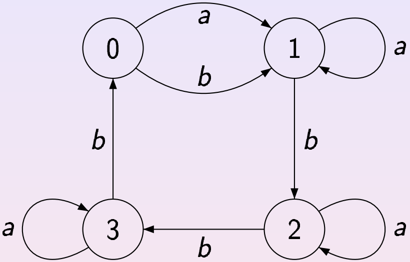
To simplify notation we often write $q \,.\, w$ for $\delta(q, w)$ and $P \,.\, w$ for $\{\delta(q, w) \mid q \in P\}$.
$\mathscr{A}$ is called synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which 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.

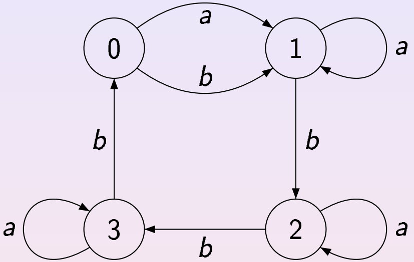
An Example



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.

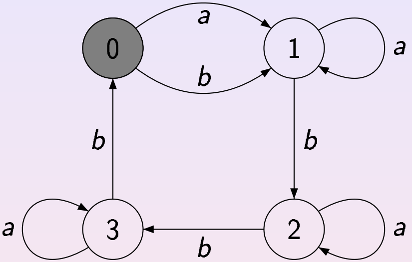
An Example



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.

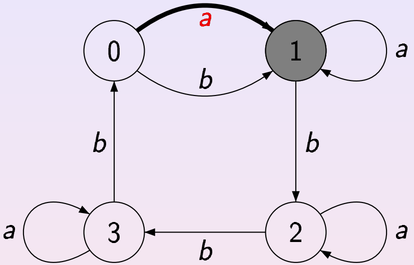
An Example



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.

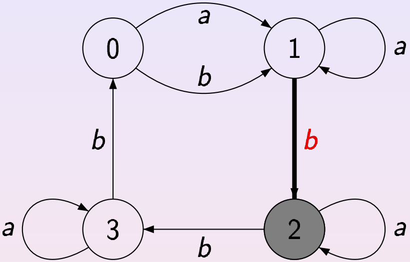
An Example



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.

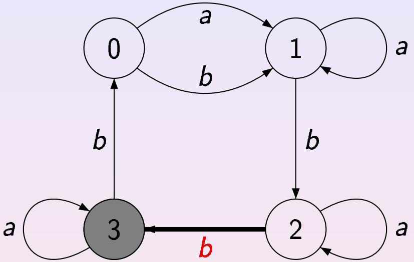
An Example



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.

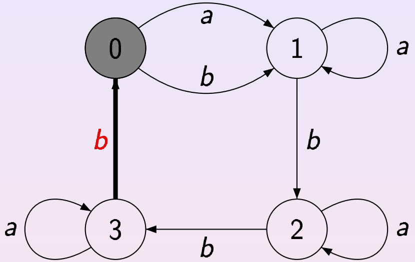
An Example



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.

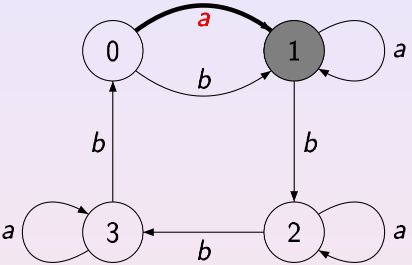
An Example



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.

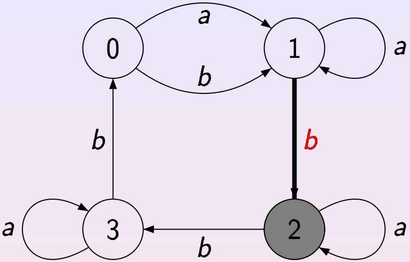
An Example



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.

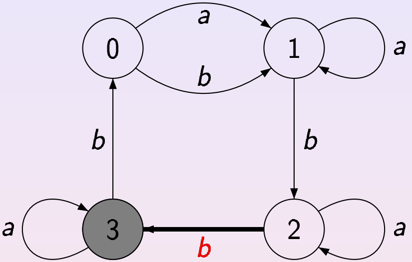
An Example



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.

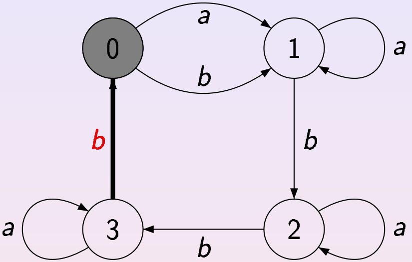
An Example



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.

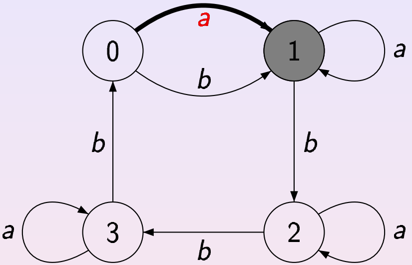
An Example



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.

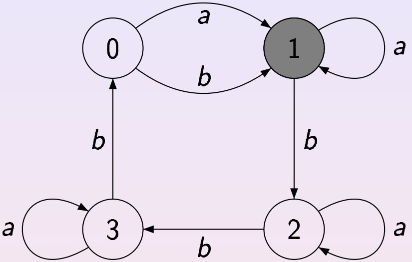
An Example



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.

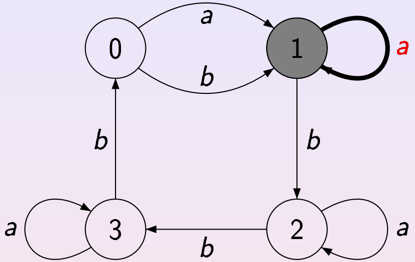
An Example



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.

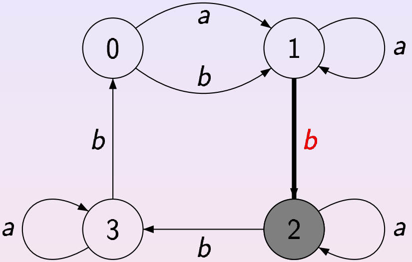
An Example



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.

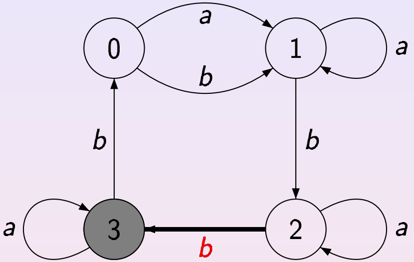
An Example



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.

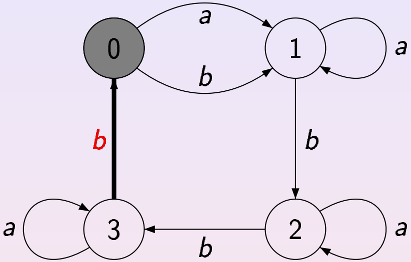
An Example



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.

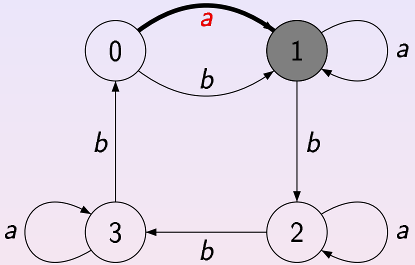
An Example



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.

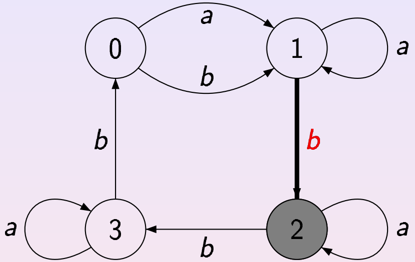
An Example



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.

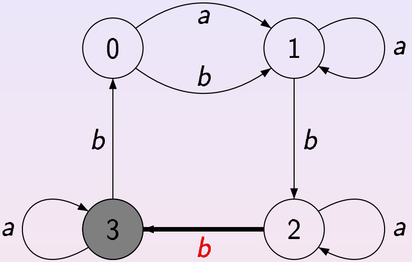
An Example



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.

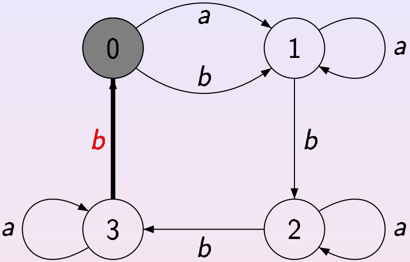
An Example



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

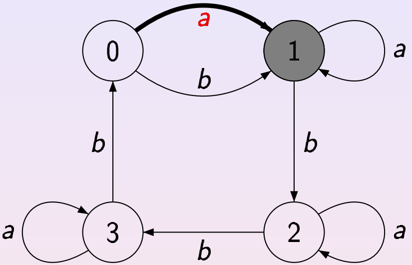
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

An Example



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

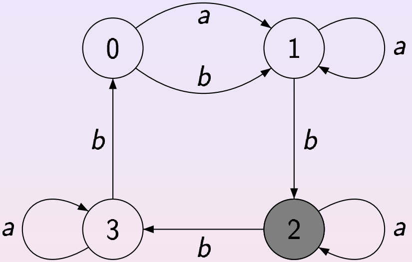
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.



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.

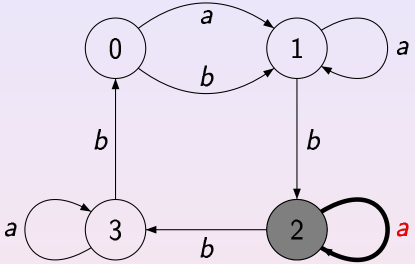
An Example



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

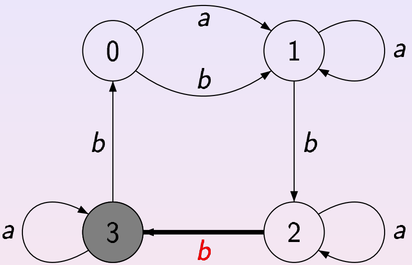
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

An Example



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

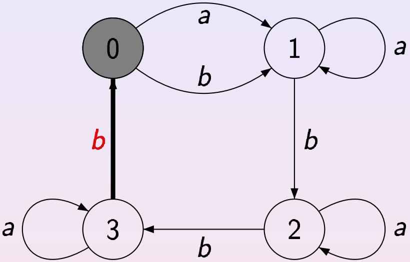
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.



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.

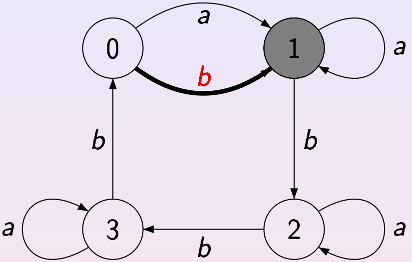
An Example



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.

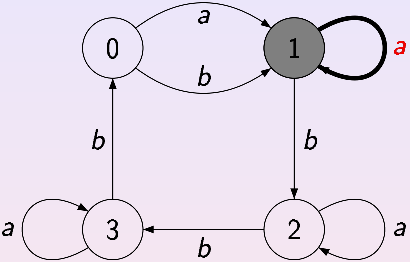
An Example



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.

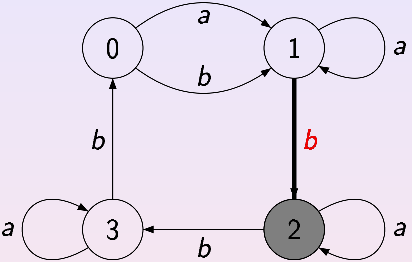
An Example



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.

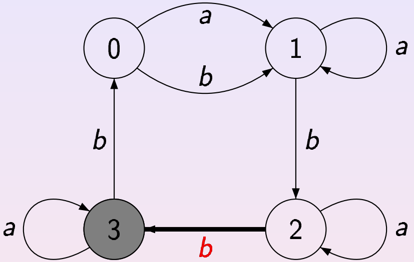
An Example



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.

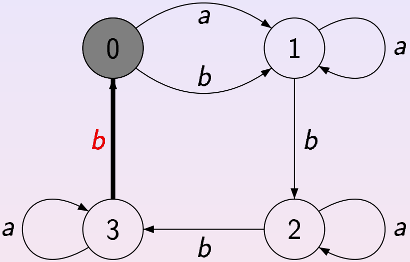
An Example



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.

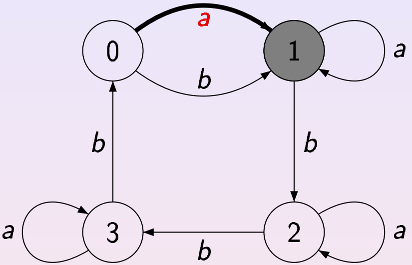
An Example



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.

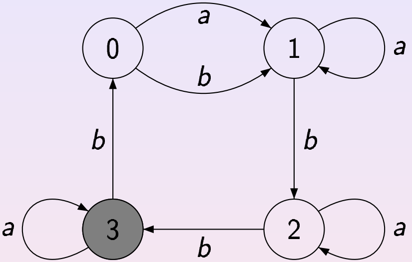
An Example



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.

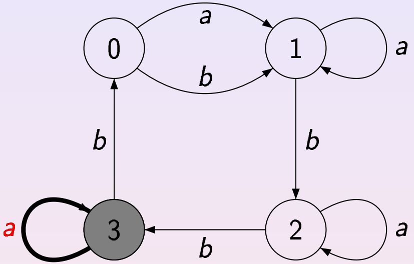
An Example



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.

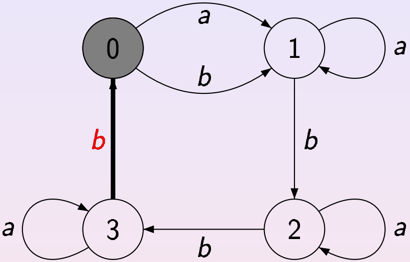
An Example



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.

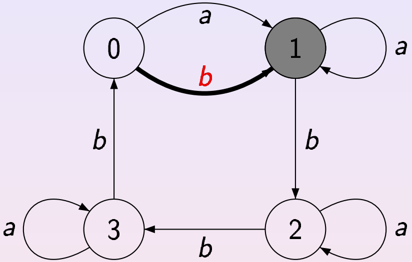
An Example



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.

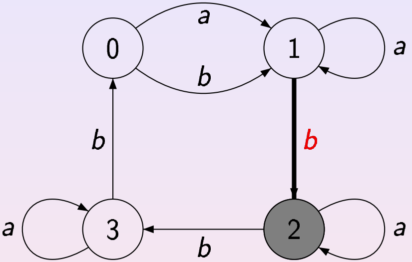
An Example



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.

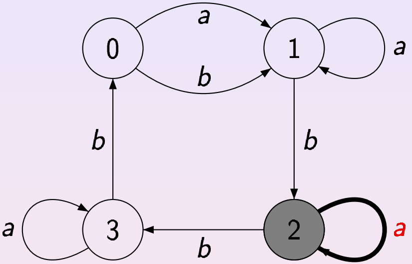
An Example



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.

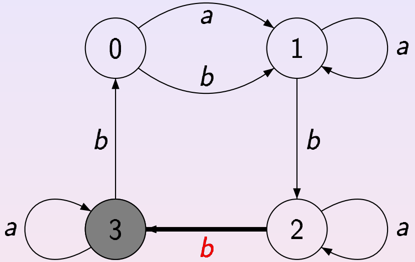
An Example



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.

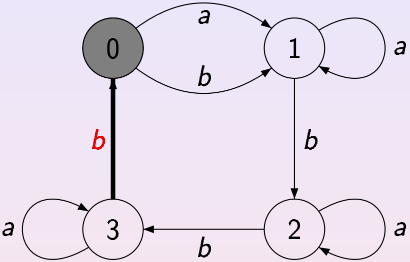
An Example



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.

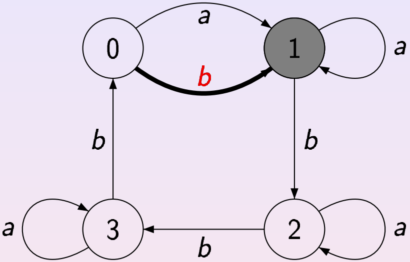
An Example



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.

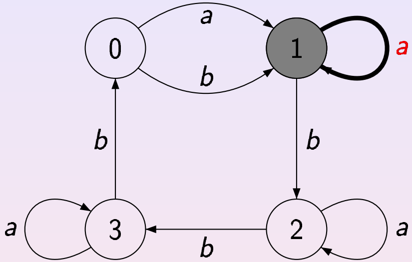
An Example



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.

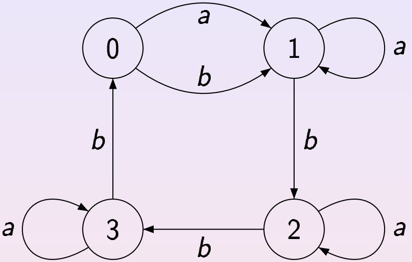
An Example



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 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 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).

Since the 1960s synchronizing automata have been considered as a useful tool for testing of reactive systems (first circuits, later protocols).

Černý's Paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 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).

Since the 1960s synchronizing automata have been considered as a useful tool for testing of reactive systems (first circuits, later protocols).

Černý's Paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 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).

Since the 1960s synchronizing automata have been considered as a useful tool for testing of reactive systems (first circuits, later protocols).

Černý's Paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konečnými automatami, Matematicko-fyzikalny Časopis Slovensk. Akad. Vied, 14, no.3, 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).

Since the 1960s synchronizing automata have been considered as a useful tool for testing of reactive systems (first circuits, later protocols).

A Frequently Discovered Notion

It is not surprising that synchronizing automata were re-invented a number of times:

- The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the early 1960s: Moore, Ginsburg.
- Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Example: A. E. Laemmel, B. Rudner, Study of the application of coding theory, Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale, N.Y., 94 pp.

A Frequently Discovered Notion

It is not surprising that synchronizing automata were re-invented a number of times:

• The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the early 1960s: Moore, Ginsburg.

• Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Example: A. E. Laemmel, B. Rudner, Study of the application of coding theory, Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale, N.Y., 94 pp.

A Frequently Discovered Notion

It is not surprising that synchronizing automata were re-invented a number of times:

• The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the early 1960s: Moore, Ginsburg.

• Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Example: A. E. Laemmel, B. Rudner, Study of the application of coding theory, Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale, N.Y., 94 pp.

A Frequently Discovered Notion

It is not surprising that synchronizing automata were re-invented a number of times:

• The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the early 1960s: Moore, Ginsburg.

• Černý's paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

Example: A. E. Laemmel, B. Rudner, Study of the application of coding theory, Report PIBEP-69-034, Polytechnic Inst. Brooklyn, Dept. Electrophysics, Farmingdale, N.Y., 94 pp.

A prefix code over a finite alphabet Σ is a set X of words in Σ^* such that no word of X is a prefix of another word of X. A prefix code is maximal if it is not contained in another prefix code over the same alphabet. A maximal prefix code X over Σ is synchronized if there is a word $x \in X^*$ such that for any word $w \in \Sigma^*$, one has $wx \in X^*$. Such a word x is called a synchronizing word for X.

The advantage of synchronized codes is that they are able to recover after a loss of synchronization between the decoder and the coder caused by channel errors.

A prefix code over a finite alphabet Σ is a set X of words in Σ^* such that no word of X is a prefix of another word of X. A prefix code is maximal if it is not contained in another prefix code over the same alphabet. A maximal prefix code X over Σ is synchronized if there is a word $x \in X^*$ such that for any word $w \in \Sigma^*$, one has $wx \in X^*$. Such a word x is called a synchronizing word for X.

The advantage of synchronized codes is that they are able to recover after a loss of synchronization between the decoder and the coder caused by channel errors.

Crash Course in Coding Theory

A prefix code over a finite alphabet Σ is a set X of words in Σ^* such that no word of X is a prefix of another word of X. A prefix code is maximal if it is not contained in another prefix code over the same alphabet. A maximal prefix code X over Σ is synchronized if there is a word $x \in X^*$ such that for any word $w \in \Sigma^*$, one has $wx \in X^*$. Such a word x is called a synchronizing word for X.

The advantage of synchronized codes is that they are able to recover after a loss of synchronization between the decoder and the coder caused by channel errors.

Crash Course in Coding Theory

A prefix code over a finite alphabet Σ is a set X of words in Σ^* such that no word of X is a prefix of another word of X. A prefix code is maximal if it is not contained in another prefix code over the same alphabet. A maximal prefix code X over Σ is synchronized if there is a word $x \in X^*$ such that for any word $w \in \Sigma^*$, one has $wx \in X^*$. Such a word x is called a synchronizing word for X.

The advantage of synchronized codes is that they are able to recover after a loss of synchronization between the decoder and the coder caused by channel errors.

Synchronized Codes

$\Sigma = \{0, 1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$.

Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent 0 0 0

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent   0 0 0 | 0 0 1 0

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent   0 0 0 | 0 0 1 0 | 0 1 1 1 | ...

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | 0 1 1 1 | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | 0 1 1 1 | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$.
Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, . . . is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | 0 1 1 1 | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .
Decoded	1 0

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0, 1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | 0 1 1 1 | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .
Decoded	1 0 | 0 0 0

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, . . . is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | 0 1 1 1 | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .
Decoded	1 0 | 0 0 0 | 1 0

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0, 1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | **0 1 1 1** | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .
Decoded	1 0 | 0 0 0 | 1 0 | **0 1 1 1** | . . .

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

Synchronized Codes

$\Sigma = \{0,1\}$, $X = \{000, 0010, 0011, 010, 0110, 0111, 10, 110, 111\}$. Then X is a maximal prefix code and one can easily check that each of the words 010, 011110, 011111110, ... is a synchronizing word for X.

Sent	0 0 0 | 0 0 1 0 | **0 1 1 1** | . . .
Received	1 0 0 0 0 1 0 0 1 1 1 . . .
Decoded	1 0 | 0 0 0 | 1 0 | **0 1 1 1** | . . .

The vertical lines show the partition of each stream into code words and the boldfaced code words indicate the position at which the decoder resynchronizes.

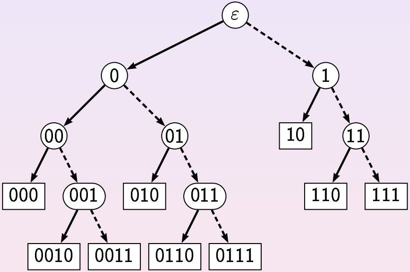
Codes vs Automata

If X is a finite maximal prefix code, then its decoding can be implemented by a DFA.

Synchronized codes precisely correspond to synchronizing automata!

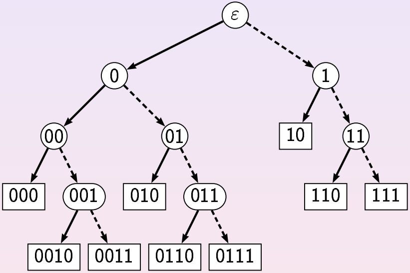
Codes vs Automata

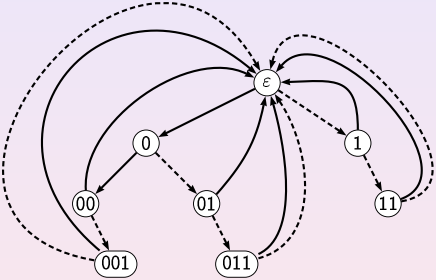
If X is a finite maximal prefix code, then its decoding can be implemented by a DFA.



Synchronized codes precisely correspond to synchronizing automata!

If X is a finite maximal prefix code, then its decoding can be implemented by a DFA.

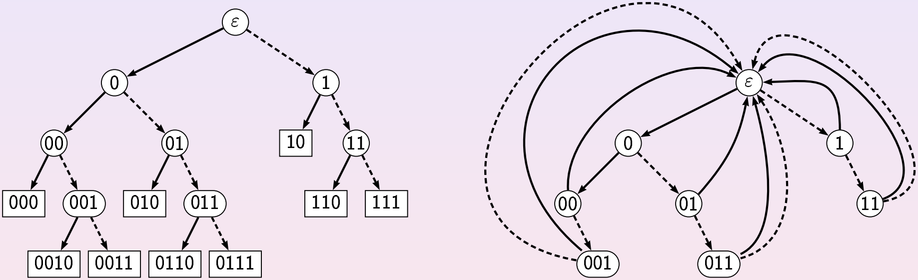




Synchronized codes precisely correspond to synchronizing automata!

Codes vs Automata

If X is a finite maximal prefix code, then its decoding can be implemented by a DFA.



Synchronized codes precisely correspond to synchronizing automata!

Re-inventing by Engineers

In the 1980s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

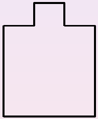
Suppose that one of the parts of a certain device has the following shape:

Such parts arrive at manufacturing sites in boxes and they need to be sorted and oriented before assembly.

Re-inventing by Engineers

In the 1980s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

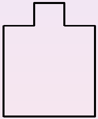
Suppose that one of the parts of a certain device has the following shape:



Such parts arrive at manufacturing sites in boxes and they need to be sorted and oriented before assembly.

In the 1980s, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

Suppose that one of the parts of a certain device has the following shape:



Such parts arrive at manufacturing sites in boxes and they need to be sorted and oriented before assembly.

Re-inventing by Engineers

Assume that only four initial orientations of the part shown above are possible, namely, the following ones:

Suppose that prior the assembly the part should take the “bump-left” orientation (the second one in the picture). Thus, one has to construct an orienter which action will put the part in the prescribed position independently of its initial orientation.

Re-inventing by Engineers

Assume that only four initial orientations of the part shown above are possible, namely, the following ones:

Suppose that prior the assembly the part should take the “bump-left” orientation (the second one in the picture). Thus, one has to construct an orienter which action will put the part in the prescribed position independently of its initial orientation.

Re-inventing by Engineers

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.

A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.

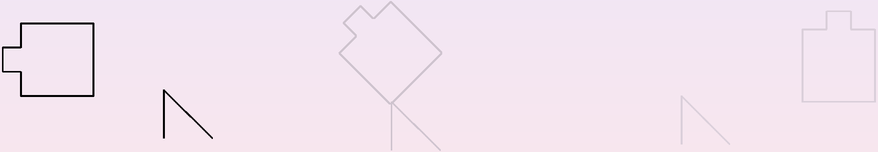
Being curried by the belt, the part then is forced to turn 90° clockwise.

RuFiDim, St Petersburg, September 22, 2011

Re-inventing by Engineers

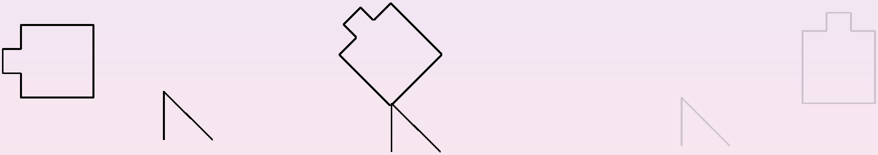
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.

A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



Being curried by the belt, the part then is forced to turn 90° clockwise.

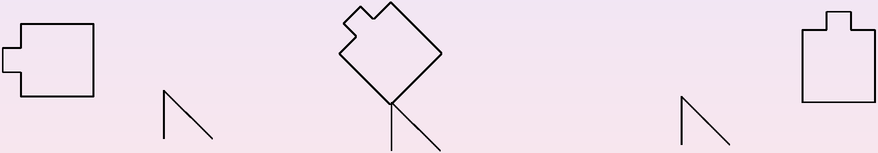
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



Being curried by the belt, the part then is forced to turn 90° clockwise.

Re-inventing by Engineers

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.

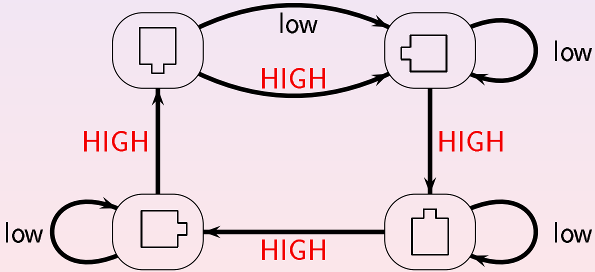


Being curried by the belt, the part then is forced to turn 90° clockwise.

A low obstacle has the same effect whenever the part is in the "bump-down" orientation; otherwise it does not touch the part which therefore passes by without changing the orientation.

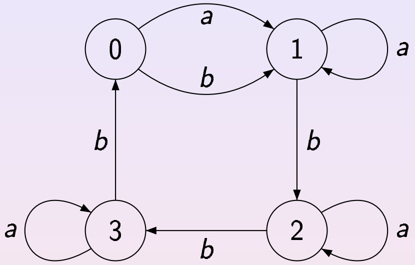
Re-inventing by Engineers

A low obstacle has the same effect whenever the part is in the "bump-down" orientation; otherwise it does not touch the part which therefore passes by without changing the orientation. The following schema summarizes how the obstacles effect the orientation of the part in question:



Re-inventing by Engineers

We met this picture a few slides ago:



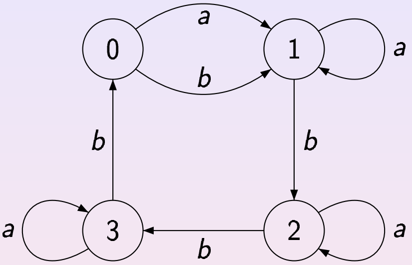
– this was our example of a synchronizing automaton, and we saw that *abbbabbba* is a reset sequence of actions. Hence the series of obstacles

low-HIGH-HIGH-HIGH-low-HIGH-HIGH-HIGH-low

yields the desired orienter.

Re-inventing by Engineers

We met this picture a few slides ago:



– this was our example of a synchronizing automaton, and we saw that *abbbabbba* is a reset sequence of actions. Hence the series of obstacles

low-HIGH-HIGH-HIGH-low-HIGH-HIGH-HIGH-low

yields the desired orienter.

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

$$\begin{array}{ccc} 0 & \mapsto & 11 \\ 1 & \mapsto & 12 \\ 2 & \mapsto & 20 \end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

$$\begin{array}{ccccc} 0 & \mapsto & 11 & \mapsto & 1212 \\ 1 & \mapsto & 12 & \mapsto & 1220 \\ 2 & \mapsto & 20 & \mapsto & 2011 \end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

$$\begin{array}{ccccccc} 0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 \\ 1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 \\ 2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 \end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

$$\begin{array}{ccccccccc} 0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 1220201112202011 \\ 1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 1220201120111212 \\ 2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 2011121212201220 \end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

$$\begin{array}{ccccccccc}
0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 1220201112202011 \\
1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 1220201120111212 \\
2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 2011121212201220
\end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

A substitution on a finite alphabet X is a map $\sigma : X \to X^+$; the substitution is said to be of constant length if all words $\sigma(x)$, $x \in X$, have the same length. One says that σ satisfies the coincidence condition if there exist positive integers m and k such that all words $\sigma^k(x)$ have the same letter in the m-th position. For an example, consider the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$. Calculate the iterations of τ up to τ^4:

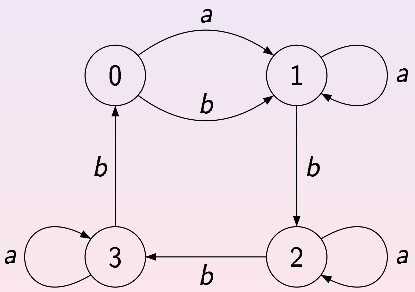
$$\begin{array}{ccccccccc} 0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 1220201112202011 \\ 1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 1220201120111212 \\ 2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 2011121212201220 \end{array}$$

Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

There is a straightforward bijection between DFAs and constant length substitutions. Each DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with $\Sigma = \{a_1, \dots, a_\ell\}$ defines a length ℓ substitution on Q that maps every $q \in Q$ to the word $(q \,.\, a_1) \dots (q \,.\, a_\ell) \in Q^+$.

Re-inventing by Dynamics Theorists

There is a straightforward bijection between DFAs and constant length substitutions. Each DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with $\Sigma = \{a_1, \ldots, a_\ell\}$ defines a length ℓ substitution on Q that maps every $q \in Q$ to the word $(q \, . \, a_1) \ldots (q \, . \, a_\ell) \in Q^+$. For instance, the automaton



induces the substitution $0 \mapsto 11, \; 1 \mapsto 12, \; 2 \mapsto 23, \; 3 \mapsto 30$.

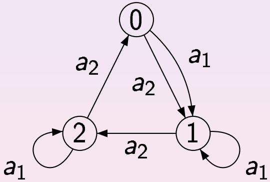
Re-inventing by Dynamics Theorists

Conversely, each substitution $\sigma : X \to X^+$ such that all words $\sigma(x)$, $x \in X$, have the same length ℓ gives rise to a DFA for which X is the state set and which has ℓ input letters $a_1, \dots, a_\ell$ acting on X as follows: $x \,.\, a_i$ is the symbol in the i-th position of the word $\sigma(x)$.

Under this bijection substitutions satisfying the coincidence condition correspond precisely to synchronizing automata, and moreover, given a substitution, the step at which the coincidence first occurs is equal to the reset threshold of the corresponding automaton.

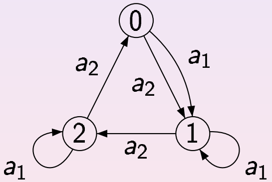
Re-inventing by Dynamics Theorists

Conversely, each substitution $\sigma : X \to X^+$ such that all words $\sigma(x)$, $x \in X$, have the same length ℓ gives rise to a DFA for which X is the state set and which has ℓ input letters $a_1, \dots, a_\ell$ acting on X as follows: $x \,.\, a_i$ is the symbol in the i-th position of the word $\sigma(x)$. For instance, the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11$, $1 \mapsto 12$, $2 \mapsto 20$ induces the automaton:



Under this bijection substitutions satisfying the coincidence condition correspond precisely to synchronizing automata, and moreover, given a substitution, the step at which the coincidence first occurs is equal to the reset threshold of the corresponding automaton.

Conversely, each substitution $\sigma : X \to X^+$ such that all words $\sigma(x)$, $x \in X$, have the same length ℓ gives rise to a DFA for which X is the state set and which has ℓ input letters $a_1, \dots, a_\ell$ acting on X as follows: $x \,.\, a_i$ is the symbol in the i-th position of the word $\sigma(x)$. For instance, the substitution τ on $X = \{0, 1, 2\}$ defined by $0 \mapsto 11,\ 1 \mapsto 12,\ 2 \mapsto 20$ induces the automaton:



Under this bijection substitutions satisfying the coincidence condition correspond precisely to synchronizing automata, and moreover, given a substitution, the step at which the coincidence first occurs is equal to the reset threshold of the corresponding automaton.

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

Vladimir Gusev, 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. He has used an approach due to Marco Almeida, Nelma Moreira and Rogério Reis, Enumeration and generation with a string automata representation, Theor. Comput. Sci., 387 (2007) 93–102.

RuFiDim, St Petersburg, September 22, 2011

Episode IV: A New Hope

Vladimir Gusev, 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. He has used an approach due to Marco Almeida, Nelma Moreira and Rogério Reis, Enumeration and generation with a string automata representation, Theor. Comput. Sci., 387 (2007) 93–102.

Second Gap

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.

Second Gap

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

Second Gap

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

Second Gap

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

Second Gap

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

Advantage of Being Old

Thus, the pattern is:

$(n-1)^2$ the first gap the "island" 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:

$(n-1)^2$ the first gap the "island" 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:

$(n-1)^2$ the first gap the "island" 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.

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

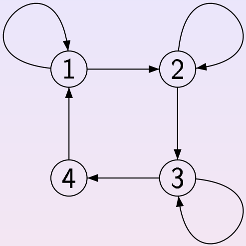
A directed graph (digraph) is a pair $D = \langle V, E \rangle$.

- 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.

For instance, the matrix of the digraph



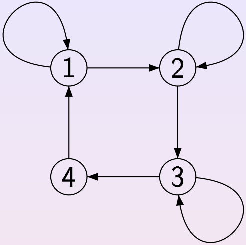
(with respect to the chosen numbering of its vertices) is $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$.

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 $\{1, 2, \dots, n\}$ 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



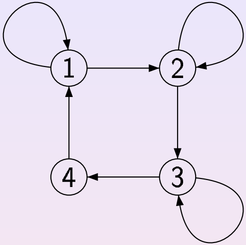
(with respect to the chosen numbering of its vertices) is $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$.

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 $\{1, 2, \dots, n\}$ 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 $p_{ij} > 0$.

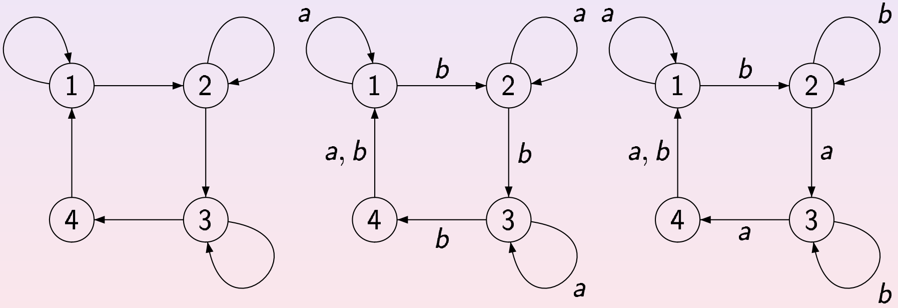
This 'two-way' correspondence allows us to reformulate in terms of digraphs several important notions and results which originated in the classical Perron–Frobenius theory of non-negative matrices.

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 in 2007. The solution is published in: 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 in 2007. The solution is published in: 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 in 2007. The solution is published in: 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 in 2007. The solution is published in: 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 $\gamma(D)$.

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

RuFiDim, St Petersburg, September 22, 2011

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 $\gamma(D)$.

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 $n^2-4n+6 < \gamma(D) < (n-1)^2$.

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

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 $\gamma(D)$.

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

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 $\gamma(D)$.

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

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 $\gamma(D)$.

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 $n^2-4n+6<\gamma(D)<(n-1)^2$.

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

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 $\gamma(D)$.

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

Exponents vs Reset Lengths

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

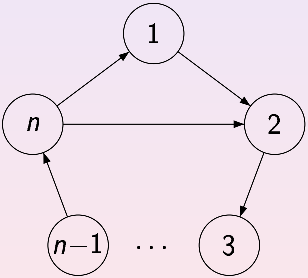
N	65	64	63	62	61	60	59	58	57	56	55	54	53	52	51
# of primitive digraphs with exponent N	1	1	0	0	0	0	0	1	1	2	0	0	0	0	4
# of 2-letter synchronizing automata with reset threshold N	0	1	0	0	0	0	0	1	2	3	0	0	0	4	4

The Wielandt Automaton

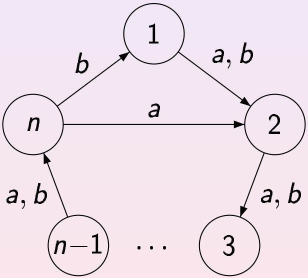
The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\gamma(W_n) = (n-1)^2 + 1$. Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.

The Wielandt Automaton

The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\gamma(W_n) = (n-1)^2 + 1$. 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 $(n, 2)$.

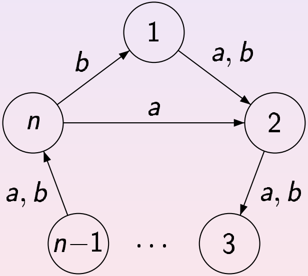


The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\gamma(W_n) = (n-1)^2 + 1$. Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



The Wielandt Automaton

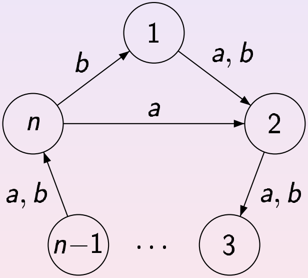
The Wielandt automaton $\mathscr{W}_n$ is a (unique) coloring of the Wielandt digraph W_n with $\gamma(W_n) = (n-1)^2 + 1$. Wielandt digraph has n vertices $1, 2, \ldots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \ldots, n-1$, $(n, 1)$, and $(n, 2)$.



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 $\gamma(W_n) = (n-1)^2 + 1$. Wielandt digraph has n vertices $1, 2, \dots, n$, say, and the following $n+1$ edges: $(i, i+1)$ for $i = 1, \dots, n-1$, $(n, 1)$, 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, every digraph with large exponent generates slowly synchronizing automata.

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring.
In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

We conjecture that this bound is tight:
The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

RuFiDim, St Petersburg, September 22, 2011

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring. In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

We conjecture that this bound is tight:
The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring.
In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

We conjecture that this bound is tight:
The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring. In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

We conjecture that this bound is tight:
The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring. In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

We conjecture that this bound is tight:

The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

Hybrid Conjecture

The Wielandt digraph admits an essentially unique coloring. In general, a digraph can be colored in many ways.

The Hybrid Problem: What is the minimum reset threshold for synchronizing colorings of a primitive digraph with n vertices?

The Wielandt digraph provides a lower bound $n^2 - 3n + 3$.

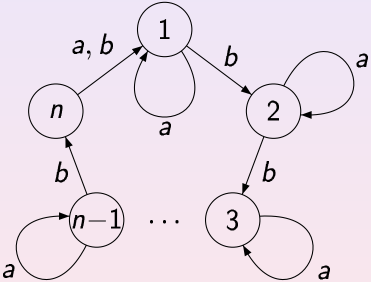
We conjecture that this bound is tight:
The Hybrid Conjecture: Every primitive digraph with n vertices admits a synchronizing coloring with reset threshold at most $n^2 - 3n + 3$.

The Černý Automaton

It turns out that the Černý automaton $\mathscr{C}_n$ is closely related to the Wielandt automaton $\mathscr{W}_n$.

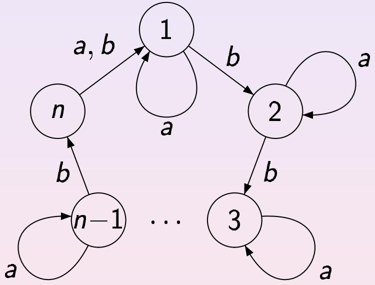
The Černý Automaton

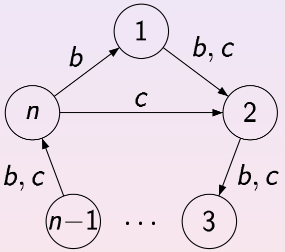
It turns out that the Černý automaton $\mathscr{C}_n$ is closely related to the Wielandt automaton $\mathscr{W}_n$.



The Černý Automaton

It turns out that the Černý automaton $\mathscr{C}_n$ is closely related to the Wielandt automaton $\mathscr{W}_n$.

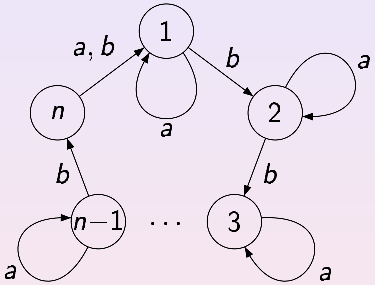


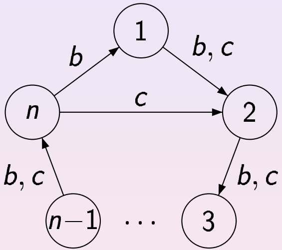


$\mathscr{C}_n$ becomes $\mathscr{W}_n$ under the action of b and $c = ab$.

The Černý Automaton

It turns out that the Černý automaton $\mathscr{C}_n$ is closely related to the Wielandt automaton $\mathscr{W}_n$.





$\mathscr{C}_n$ becomes $\mathscr{W}_n$ under the action of b and $c = ab$.
It is easy to see that every shortest reset word of $\mathscr{C}_n$ transforms into a reset word of $\mathscr{W}_n$, and this allows one to readily recover the reset threshold bound $(n-1)^2$ for $\mathscr{C}_n$.

Exponents vs Reset Thresholds

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	4
# of 2-letter synchronizing automata with reset threshold N	0	1	0	0	0	0	0	1	2	3	0	0	0	4	4

Exponents vs Reset Thresholds

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	4
# of 2-letter synchronizing automata with reset threshold N	0	1	0	0	0	0	0	1	2	3	0	0	0	4	4

Exponents vs Reset Thresholds

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	4
# of 2-letter synchronizing automata with reset threshold N	0	1	0	0	0	0	0	1	2	3	0	0	0	4	4

1-Exponents

The notion of exponent is too weak for isolating synchronizing automata with maximal reset threshold in the class of Eulerian automata. For this class the Černý Conjecture has been proved by Jarkko Kari (Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295 (2003) 223–232) but lower and upper bounds for reset thresholds of Eulerian automata known so far do not match each other. The best lower bound is due to Pavel Martyugin (unpublished).

The reason is that we discard too much information when passing from synchronizability to primitivity—we forget anything but length about paths labeled by reset words. Thus, we have tried another approach in which more information is preserved, namely, the Parikh vectors of the paths are taken into account.

1-Exponents

The notion of exponent is too weak for isolating synchronizing automata with maximal reset threshold in the class of Eulerian automata. For this class the Černý Conjecture has been proved by Jarkko Kari (Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295 (2003) 223–232) but lower and upper bounds for reset thresholds of Eulerian automata known so far do not match each other. The best lower bound is due to Pavel Martyugin (unpublished).

The reason is that we discard too much information when passing from synchronizability to primitivity—we forget anything but length about paths labeled by reset words. Thus, we have tried another approach in which more information is preserved, namely, the Parikh vectors of the paths are taken into account.

1-Exponents

The notion of exponent is too weak for isolating synchronizing automata with maximal reset threshold in the class of Eulerian automata. For this class the Černý Conjecture has been proved by Jarkko Kari (Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295 (2003) 223–232) but lower and upper bounds for reset thresholds of Eulerian automata known so far do not match each other. The best lower bound is due to Pavel Martyugin (unpublished).

The reason is that we discard too much information when passing from synchronizability to primitivity—we forget anything but length about paths labeled by reset words. Thus, we have tried another approach in which more information is preserved, namely, the Parikh vectors of the paths are taken into account.

1-Exponents

The notion of exponent is too weak for isolating synchronizing automata with maximal reset threshold in the class of Eulerian automata. For this class the Černý Conjecture has been proved by Jarkko Kari (Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295 (2003) 223–232) but lower and upper bounds for reset thresholds of Eulerian automata known so far do not match each other. The best lower bound is due to Pavel Martyugin (unpublished).

The reason is that we discard too much information when passing from synchronizability to primitivity—we forget anything but length about paths labeled by reset words. Thus, we have tried another approach in which more information is preserved, namely, the Parikh vectors of the paths are taken into account.

1-Exponents

The notion of exponent is too weak for isolating synchronizing automata with maximal reset threshold in the class of Eulerian automata. For this class the Černý Conjecture has been proved by Jarkko Kari (Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295 (2003) 223–232) but lower and upper bounds for reset thresholds of Eulerian automata known so far do not match each other. The best lower bound is due to Pavel Martyugin (unpublished).

The reason is that we discard too much information when passing from synchronizability to primitivity—we forget anything but length about paths labeled by reset words. Thus, we have tried another approach in which more information is preserved, namely, the Parikh vectors of the paths are taken into account.

1-Exponents

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA with $|\Sigma| = k$ and fix some ordering of the letters in Σ. Define a subset $\mathrm{E}_1(\mathscr{A})$ of $\mathbb{N}_0^k$ as follows: a vector $\mathrm{v} \in \mathbb{N}_0^k$ belongs to $\mathrm{E}_1(\mathscr{A})$ if and only if there is $r \in Q$ such that for every $p \in Q$, there exists a path from p to r such that v is the Parikh vector of the path's label. If $\mathrm{E}_1(\mathscr{A}) \neq \varnothing$ then $\mathscr{A}$ is called 1-primitive.

The minimum value of the sum $i_1 + i_2 + \cdots + i_k$ over all k-tuples $(i_1, i_2, \ldots, i_k) \in \mathrm{E}_1(\mathscr{A})$ is denoted by $\exp_1(\mathscr{A})$. Clearly, every synchronizing automaton $\mathscr{A}$ is 1-primitive and $\exp_1(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

RuFiDim, St Petersburg, September 22, 2011

1-Exponents

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA with $|\Sigma| = k$ and fix some ordering of the letters in Σ. Define a subset $\mathrm{E}_1(\mathscr{A})$ of $\mathbb{N}_0^k$ as follows: a vector $\mathrm{v} \in \mathbb{N}_0^k$ belongs to $\mathrm{E}_1(\mathscr{A})$ if and only if there is $r \in Q$ such that for every $p \in Q$, there exists a path from p to r such that v is the Parikh vector of the path's label. If $\mathrm{E}_1(\mathscr{A}) \neq \varnothing$ then $\mathscr{A}$ is called 1-primitive.

The minimum value of the sum $i_1 + i_2 + \cdots + i_k$ over all k-tuples $(i_1, i_2, \ldots, i_k) \in \mathrm{E}_1(\mathscr{A})$ is denoted by $\exp_1(\mathscr{A})$. Clearly, every synchronizing automaton $\mathscr{A}$ is 1-primitive and $\exp_1(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

1-Exponents

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA with $|\Sigma| = k$ and fix some ordering of the letters in Σ. Define a subset $\mathrm{E}_1(\mathscr{A})$ of $\mathbb{N}_0^k$ as follows: a vector $\mathrm{v} \in \mathbb{N}_0^k$ belongs to $\mathrm{E}_1(\mathscr{A})$ if and only if there is $r \in Q$ such that for every $p \in Q$, there exists a path from p to r such that v is the Parikh vector of the path's label. If $\mathrm{E}_1(\mathscr{A}) \neq \varnothing$ then $\mathscr{A}$ is called 1-primitive.

The minimum value of the sum $i_1 + i_2 + \cdots + i_k$ over all k-tuples $(i_1, i_2, \ldots, i_k) \in \mathrm{E}_1(\mathscr{A})$ is denoted by $\exp_1(\mathscr{A})$. Clearly, every synchronizing automaton $\mathscr{A}$ is 1-primitive and $\exp_1(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

1-Exponents

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA with $|\Sigma| = k$ and fix some ordering of the letters in Σ. Define a subset $\mathrm{E}_1(\mathscr{A})$ of $\mathbb{N}_0^k$ as follows: a vector $\mathrm{v} \in \mathbb{N}_0^k$ belongs to $\mathrm{E}_1(\mathscr{A})$ if and only if there is $r \in Q$ such that for every $p \in Q$, there exists a path from p to r such that v is the Parikh vector of the path's label. If $\mathrm{E}_1(\mathscr{A}) \neq \varnothing$ then $\mathscr{A}$ is called 1-primitive.

The minimum value of the sum $i_1 + i_2 + \cdots + i_k$ over all k-tuples $(i_1, i_2, \ldots, i_k) \in \mathrm{E}_1(\mathscr{A})$ is denoted by $\exp_1(\mathscr{A})$. Clearly, every synchronizing automaton $\mathscr{A}$ is 1-primitive and $\exp_1(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

1-Exponents

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA with $|\Sigma| = k$ and fix some ordering of the letters in Σ. Define a subset $\mathrm{E}_1(\mathscr{A})$ of $\mathbb{N}_0^k$ as follows: a vector $\mathrm{v} \in \mathbb{N}_0^k$ belongs to $\mathrm{E}_1(\mathscr{A})$ if and only if there is $r \in Q$ such that for every $p \in Q$, there exists a path from p to r such that v is the Parikh vector of the path's label. If $\mathrm{E}_1(\mathscr{A}) \neq \varnothing$ then $\mathscr{A}$ is called 1-primitive.

The minimum value of the sum $i_1 + i_2 + \cdots + i_k$ over all k-tuples $(i_1, i_2, \ldots, i_k) \in \mathrm{E}_1(\mathscr{A})$ is denoted by $\exp_1(\mathscr{A})$. Clearly, every synchronizing automaton $\mathscr{A}$ is 1-primitive and $\exp_1(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

k-Exponents

Synchronizing automata — there exists a target state which one can reach from all states by paths labeled by the same word

RuFiDim, St Petersburg, September 22, 2011

k-Exponents

Synchronizing automata — there exists a target state which one can reach from all states by paths labeled by the same word

Primitive automata — there exists a target state which one can reach from all states by paths of the same length

k-Exponents

Synchronizing automata — there exists a target state which one can reach from all states by paths labeled by the same word

Primitive automata — there exists a target state which one can reach from all states by paths of the same length

1-primitive automata — there exists a target state which one can reach from all states by paths labeled by words with the same Parikh vector

k-Exponents

Synchronizing automata — there exists a target state which one can reach from all states by paths labeled by the same word

Primitive automata — there exists a target state which one can reach from all states by paths of the same length

1-primitive automata — there exists a target state which one can reach from all states by paths labeled by words with the same Parikh vector

k-primitive automata — there exists a target state which one can reach from all states by paths labeled by words in which every factor of length at most k occurs the same number of times

k-Exponents

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \dots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \dots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.

k-Exponents

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \dots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.

k-Exponents

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \dots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.

k-Exponents

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \dots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.

k-Exponents

The minimal length of words that witness k-primitivity of a DFA $\mathscr{A}$ is denoted by $\exp_k(\mathscr{A})$.

It is clear that every synchronizing automaton $\mathscr{A}$ is k-primitive for all k and each $\exp_k(\mathscr{A})$ serves as a lower bound for the reset threshold of $\mathscr{A}$.

We have the non-decreasing exponent sequence:

$$\exp_1(\mathscr{A}) \leq \cdots \leq \exp_k(\mathscr{A}) \leq \exp_{k+1}(\mathscr{A}) \leq \ldots .$$

At every next step we require that words labeling coterminal paths get more and more similar to each other. Eventually these words "converge" to a reset word and the sequence stabilizes at the reset threshold of $\mathscr{A}$. Our hope is that studying exponent sequences may shed new light on the Černý conjecture.