

Synchronizing Automata

Mikhail Volkov

Ural State University, Ekaterinburg, Russia



Niš, Nov 10, 2009

Definitions and terminology

We consider DFA: $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ the transition function

\mathcal{A} is called **synchronizing** if there exists a word $w \in \Sigma^*$ whose action resets \mathcal{A} , that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \cdot w| = 1$. Here $Q \cdot v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a **reset word** for \mathcal{A} .

Other names:

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

Niš, Nov 10, 2009

Definitions and terminology

We consider DFA: $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ the transition function

\mathcal{A} is called **synchronizing** if there exists a word $w \in \Sigma^*$ whose action resets \mathcal{A} , that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \cdot w| = 1$. Here $Q \cdot v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a **reset word** for \mathcal{A} .

Other names:

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

Niš, Nov 10, 2009

Definitions and terminology

We consider DFA: $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ the transition function

\mathcal{A} is called **synchronizing** if there exists a word $w \in \Sigma^*$ whose action resets \mathcal{A} , that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \cdot w| = 1$. Here $Q \cdot v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a **reset word** for \mathcal{A} .

Other names:

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

Niš, Nov 10, 2009

Definitions and terminology

We consider DFA: $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ the transition function

\mathcal{A} is called **synchronizing** if there exists a word $w \in \Sigma^*$ whose action resets \mathcal{A} , that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \cdot w| = 1$. Here $Q \cdot v = \{\delta(q, v) \mid q \in Q\}$.

Any w with this property is a **reset word** for \mathcal{A} .

Other names:

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

Niš, Nov 10, 2009

Definitions and terminology

We consider DFA: $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$.

- Q the state set
- Σ the input alphabet
- $\delta : Q \times \Sigma \rightarrow Q$ the transition function

\mathcal{A} is called **synchronizing** if there exists a word $w \in \Sigma^*$ whose action resets \mathcal{A} , that is, leaves the automaton in one particular state no matter which state in Q it started at: $\delta(q, w) = \delta(q', w)$ for all $q, q' \in Q$.

$|Q \cdot w| = 1$. Here $Q \cdot v = \{\delta(q, v) \mid q \in Q\}$.

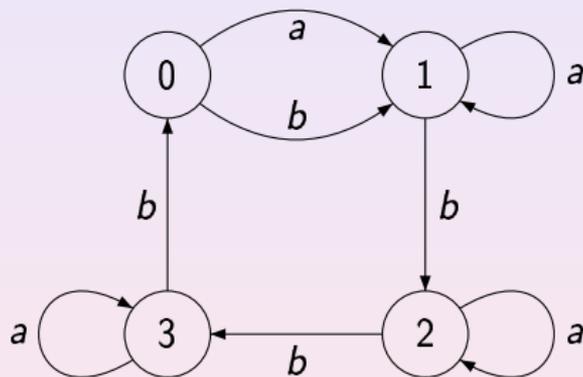
Any w with this property is a **reset word** for \mathcal{A} .

Other names:

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

Niš, Nov 10, 2009

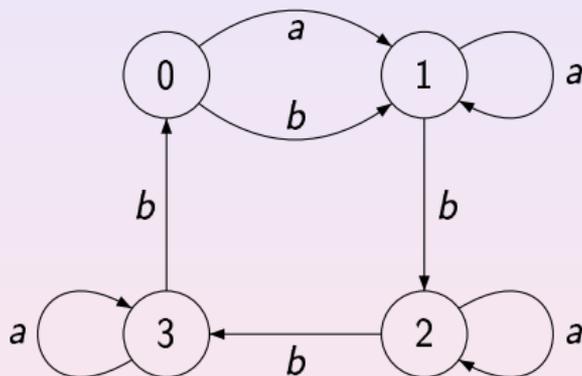
An example



A reset word is $abbabbbba$: applying it at any state brings the automaton to the state 1

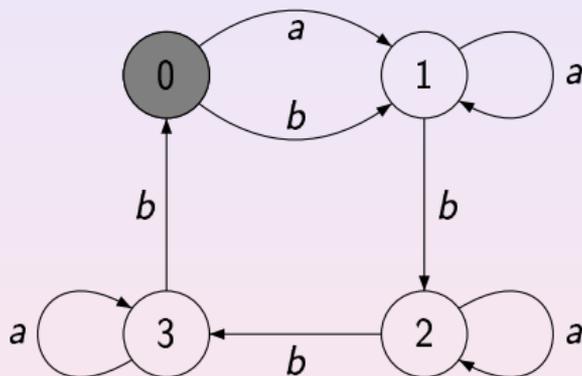
Niš, Nov 10, 2009

An example



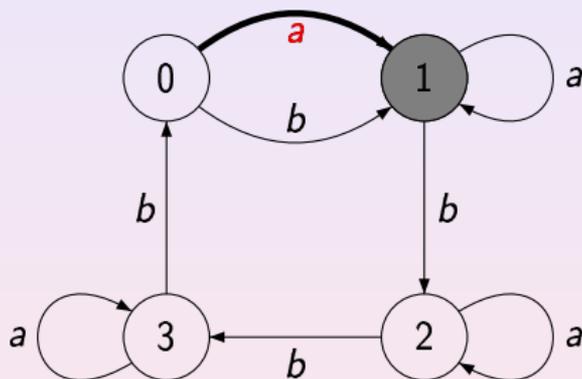
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



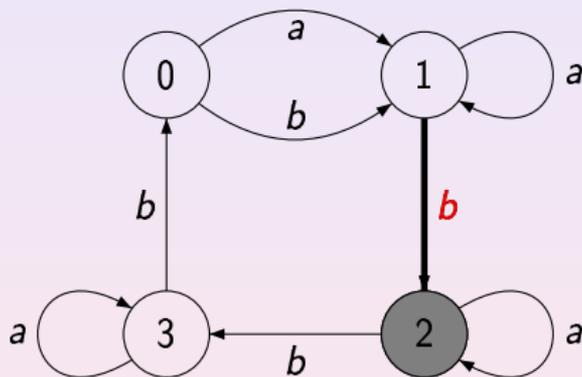
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



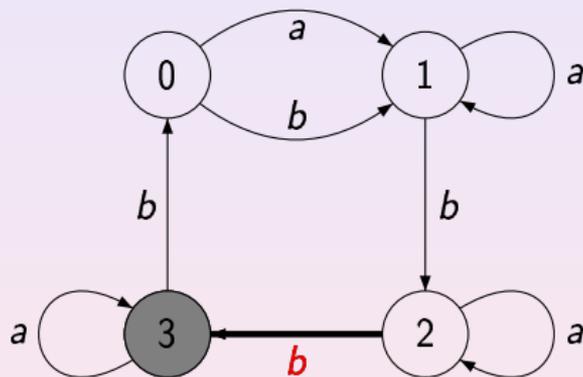
A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

An example



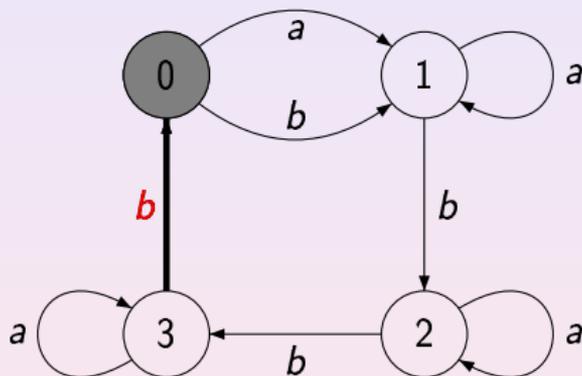
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



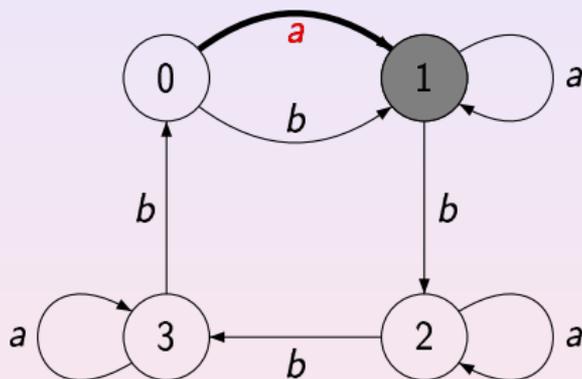
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



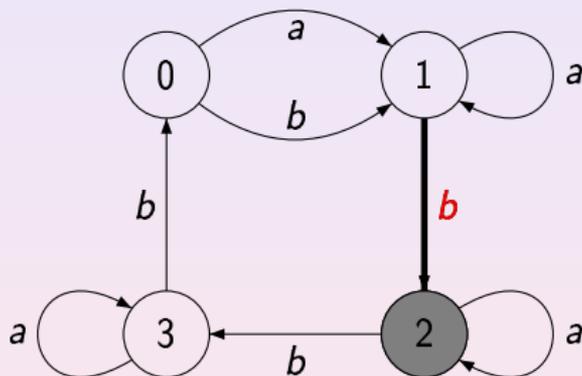
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



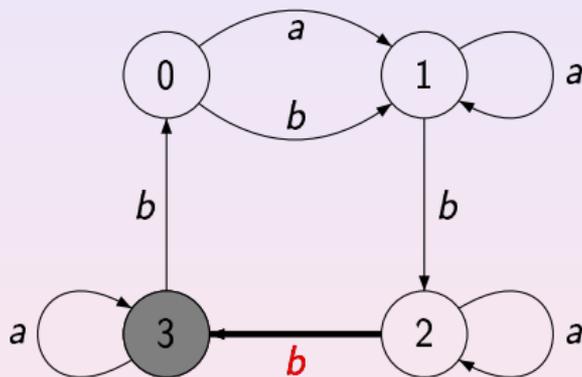
A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

An example



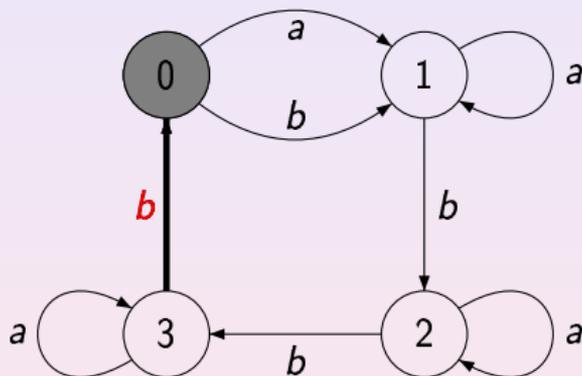
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



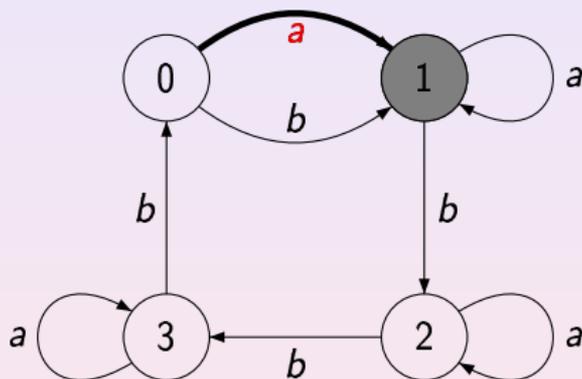
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



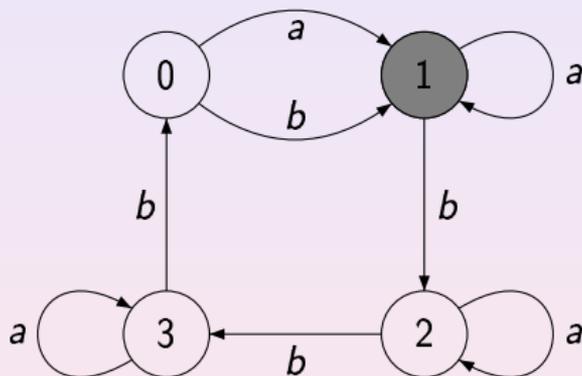
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



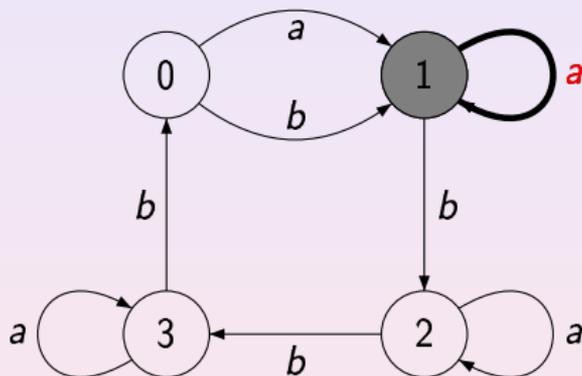
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



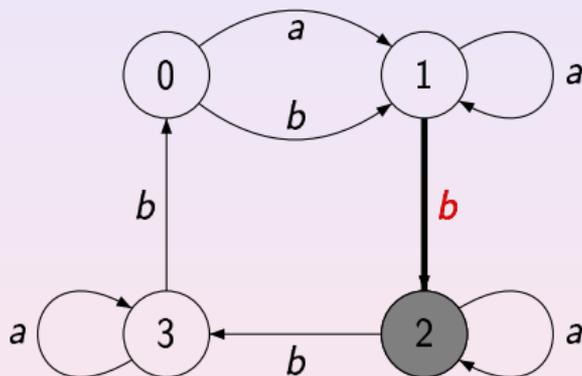
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



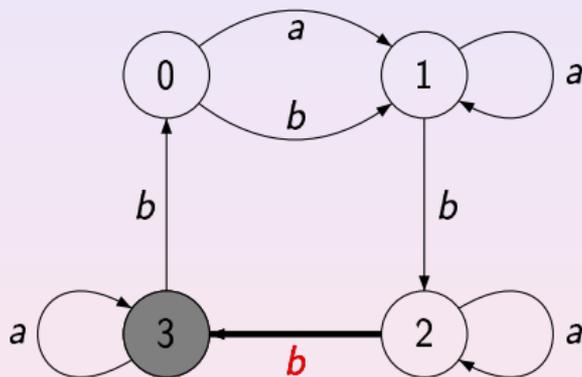
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



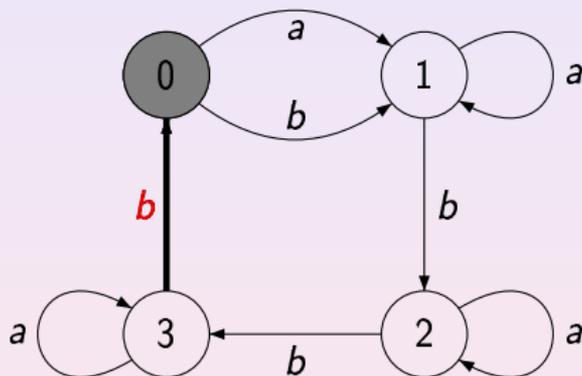
A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

An example



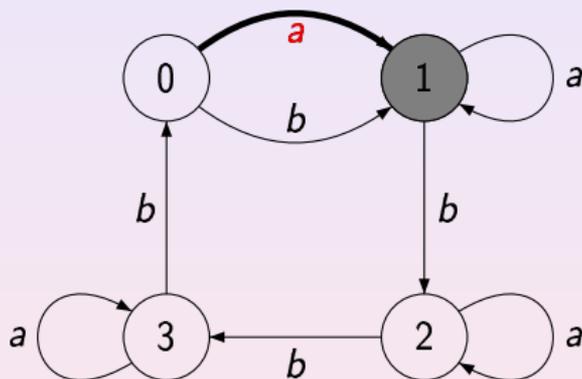
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



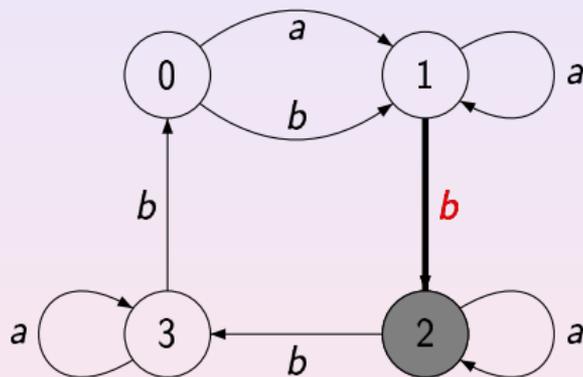
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



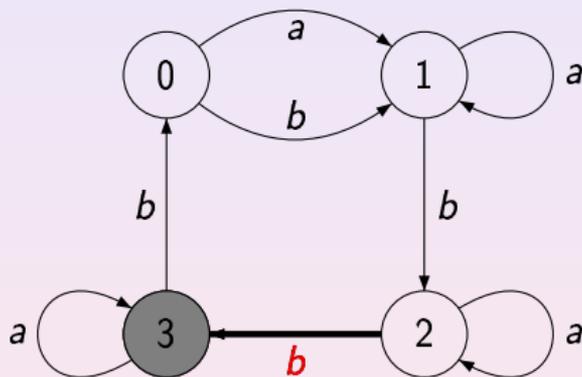
A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

An example



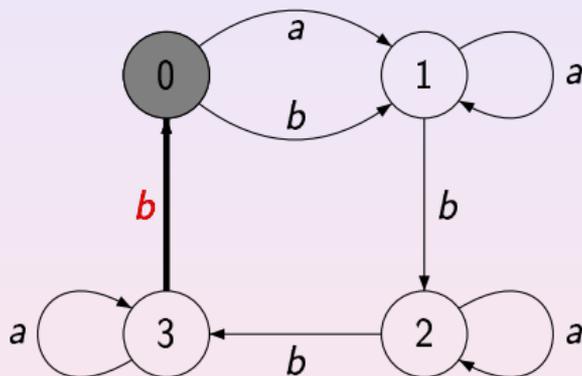
A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

An example



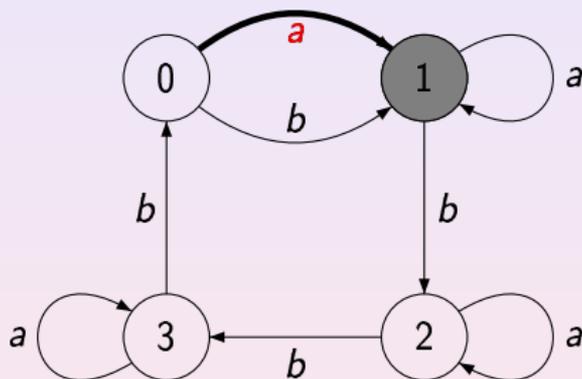
A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



A reset word is $abbbabba$: applying it at any state brings the automaton to the state 1

An example



A reset word is $abbabbba$: applying it at any state brings the automaton to the state 1

Černý paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym experimentom 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).

Niš, Nov 10, 2009

Černý paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym experimentom 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).

Niš, Nov 10, 2009

Černý paper

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym experimentom 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).

Niš, Nov 10, 2009

Other sources

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 1960s.
- 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.

Niš, Nov 10, 2009

Other sources

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 1960s.
- 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.

Niš, Nov 10, 2009

Other sources

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 1960s.
- 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.

Niš, Nov 10, 2009

Other sources

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 1960s.
- 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.

Niš, Nov 10, 2009

Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for **testing of reactive systems** (first circuits, later protocols).

In the 80s, 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.

Niš, Nov 10, 2009

Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for **testing of reactive systems** (first circuits, later protocols).

In the 80s, 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.

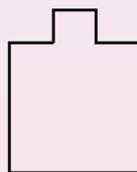
Niš, Nov 10, 2009

Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for **testing of reactive systems** (first circuits, later protocols).

In the 80s, 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.

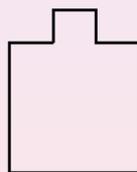
Niš, Nov 10, 2009

Re-inventing by Engineers

Since the 60s and till the 90s synchronizing automata were considered as a useful tool for **testing of reactive systems** (first circuits, later protocols).

In the 80s, 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.

Niš, Nov 10, 2009

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.

Niš, Nov 10, 2009

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.

Niš, Nov 10, 2009

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

Niš, Nov 10, 2009

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.



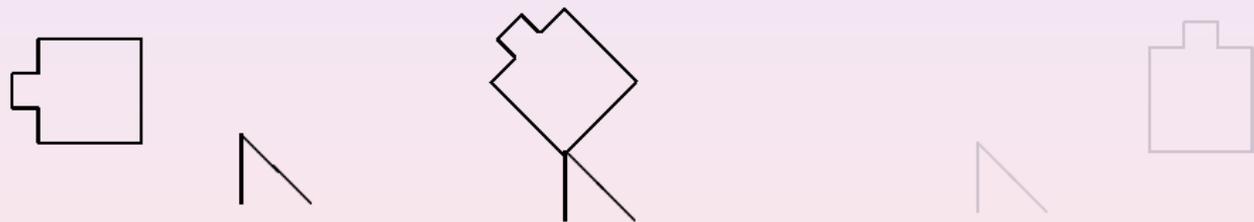
Being carried by the belt, the part then is forced to turn 90° clockwise.

Niš, Nov 10, 2009

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.



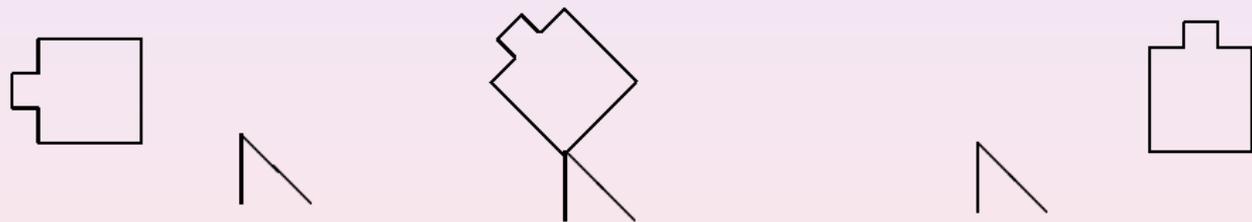
Being carried by the belt, the part then is forced to turn 90° clockwise.

Niš, Nov 10, 2009

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.

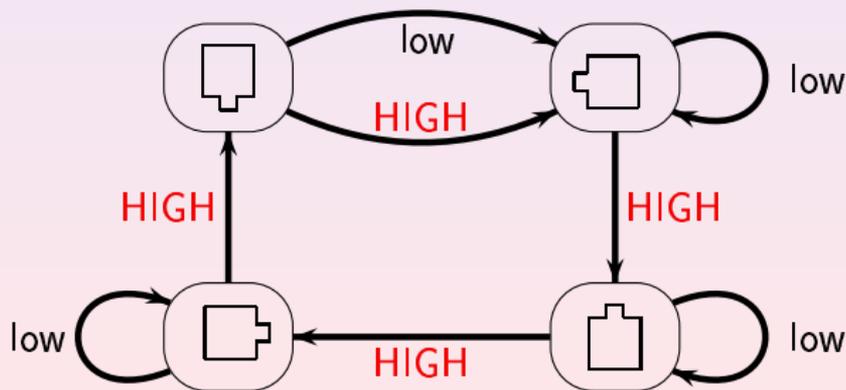


Being carried by the belt, the part then is forced to turn 90° clockwise.

Niš, Nov 10, 2009

Re-inventing by Engineers

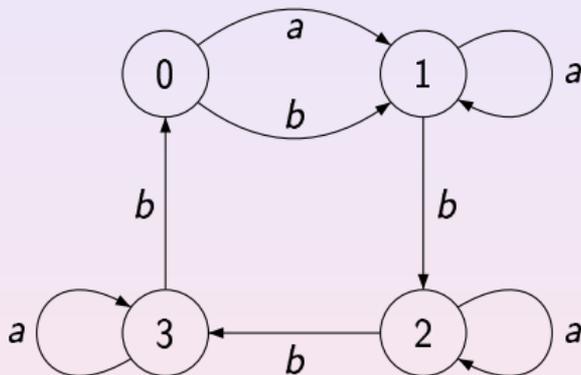
A low obstacle has the same effect whenever the part is in the “bump-down” orientation; otherwise it does not touch the part which therefore passes by without changing the orientation. The following schema summarizes how the obstacles effect the orientation of the part in question:



Niš, Nov 10, 2009

Re-inventing by Engineers

We met this picture a few slides ago:



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

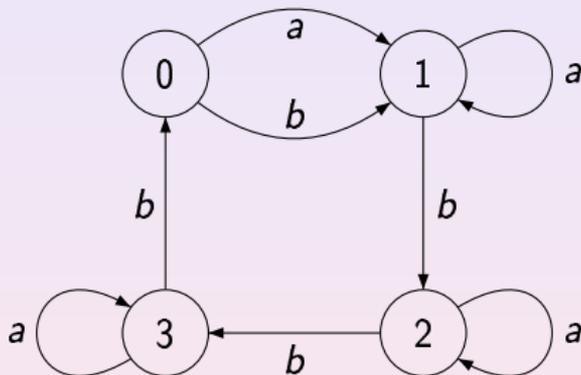
low-HIGH-HIGH-HIGH-low-HIGH-HIGH-HIGH-low

yields the desired sensorless orienter.

Niš, Nov 10, 2009

Re-inventing by Engineers

We met this picture a few slides ago:



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

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

yields the desired sensorless orienter.

Niš, Nov 10, 2009

Possible Use in Biocomputing

In **DNA-computing**, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc).

They have produced a solution containing 3×10^{12} identical DNA-based automata per μl . These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states. One has to feed the automata with an reset sequence (again encoded by a DNA-strand) in order to get them ready for a new use.

Niš, Nov 10, 2009

Possible Use in Biocomputing

In **DNA-computing**, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc).

They have produced a solution containing 3×10^{12} identical DNA-based automata per μl . These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states. One has to feed the automata with an reset sequence (again encoded by a DNA-strand) in order to get them ready for a new use.

Niš, Nov 10, 2009

Possible Use in Biocomputing

In **DNA-computing**, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc).

They have produced a solution containing 3×10^{12} identical DNA-based automata per μl . These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states. One has to feed the automata with an reset sequence (again encoded by a DNA-strand) in order to get them ready for a new use.

Niš, Nov 10, 2009

Possible Use in Biocomputing

In **DNA-computing**, there is a fast progressing work by Ehud Shapiro's group on "*soup of automata*" (Programmable and autonomous computing machine made of biomolecules, Nature 414, no.1 (November 22, 2001) 430–434; DNA molecule provides a computing machine with both data and fuel, Proc. National Acad. Sci. USA 100 (2003) 2191–2196, etc).

They have produced a solution containing 3×10^{12} identical DNA-based automata per μl . These automata can work in parallel on different inputs (DNA strands), thus ending up in different and unpredictable states. One has to feed the automata with an reset sequence (again encoded by a DNA-strand) in order to get them ready for a new use.

Niš, Nov 10, 2009

Outline of the talk

- From the viewpoint of applications, real or yet imaginary, **algorithmic issues** are of crucial importance.
- Synchronizing automata constitute an interesting combinatorial object. Their studies are mainly motivated by the **Černý conjecture**.

Niš, Nov 10, 2009

Outline of the talk

- From the viewpoint of applications, real or yet imaginary, **algorithmic issues** are of crucial importance.
- Synchronizing automata constitute an interesting combinatorial object. Their studies are mainly motivated by the **Černý conjecture**.

Niš, Nov 10, 2009

Power automaton

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

The *power automaton* $\mathcal{P}(\mathcal{A})$ of a given DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$:

- states are the non-empty subsets of Q ,
- $\delta(P, a) = P \cdot a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA \mathcal{A} iff w labels a path in $\mathcal{P}(\mathcal{A})$ starting at Q and ending at a singleton.

Power automaton

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

The *power automaton* $\mathcal{P}(\mathcal{A})$ of a given DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$:

- states are the non-empty subsets of Q ,
- $\delta(P, a) = P \cdot a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA \mathcal{A} iff w labels a path in $\mathcal{P}(\mathcal{A})$ starting at Q and ending at a singleton.

Power automaton

Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

The *power automaton* $\mathcal{P}(\mathcal{A})$ of a given DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$:

- states are the non-empty subsets of Q ,
- $\delta(P, a) = P \cdot a = \{\delta(p, a) \mid p \in P\}$

A $w \in \Sigma^*$ is a reset word for the DFA \mathcal{A} iff w labels a path in $\mathcal{P}(\mathcal{A})$ starting at Q and ending at a singleton.

Power automaton

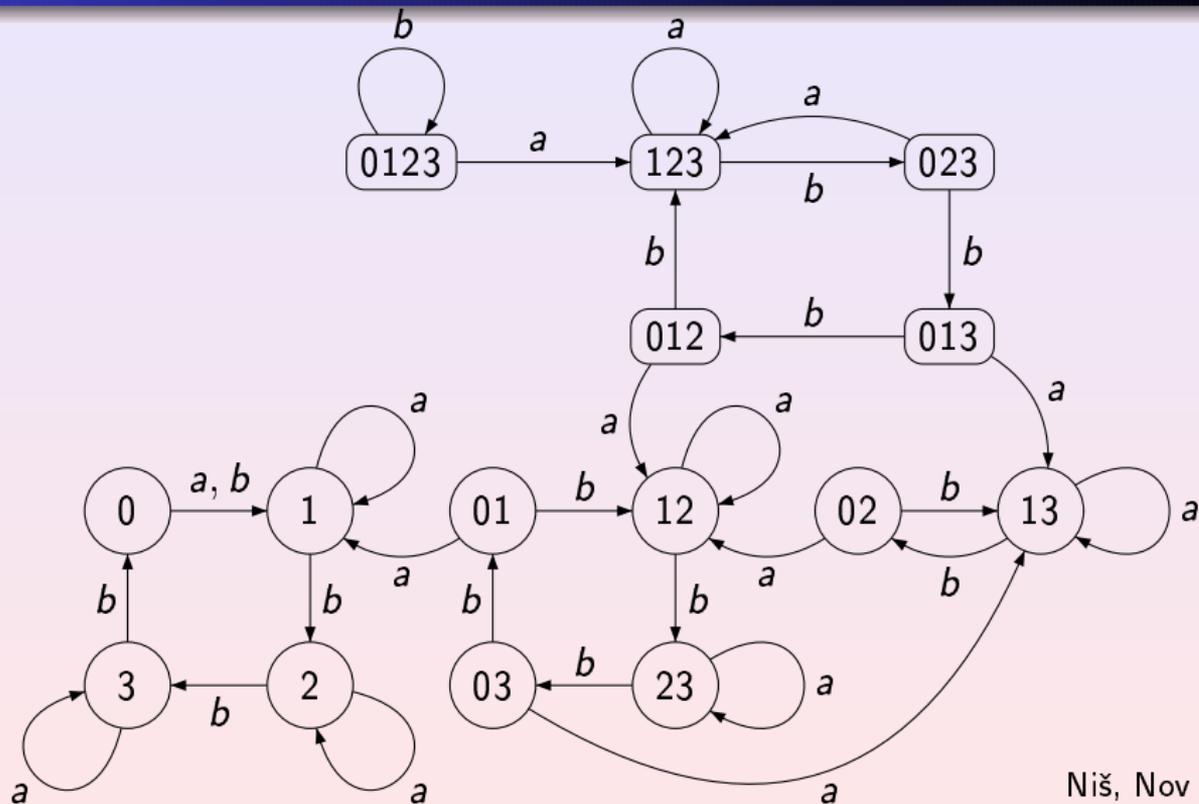
Not every DFA is synchronizing. Therefore, the very first question is the following one: *given an automaton, how to determine whether or not it is synchronizing?* This question is easy, and a straightforward solution comes from the classic power automaton construction.

The *power automaton* $\mathcal{P}(\mathcal{A})$ of a given DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$:

- states are the non-empty subsets of Q ,
- $\delta(P, a) = P \cdot a = \{\delta(p, a) \mid p \in P\}$

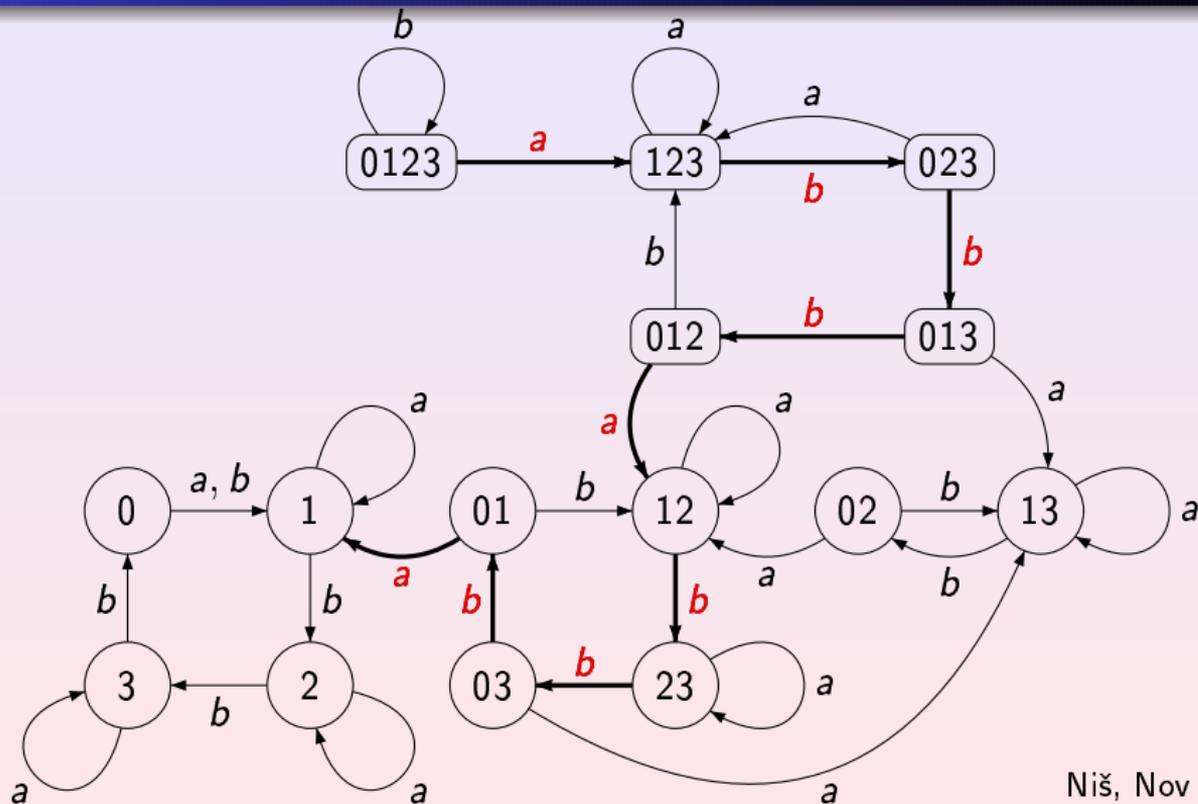
A $w \in \Sigma^*$ is a reset word for the DFA \mathcal{A} iff w labels a path in $\mathcal{P}(\mathcal{A})$ starting at Q and ending at a singleton.

An example



Niš, Nov 10, 2009

An example



Niš, Nov 10, 2009

Polynomial algorithm

Thus, the question of whether or not a given DFA \mathcal{A} is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathcal{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of \mathcal{A} .

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing iff for every $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.*

Polynomial algorithm

Thus, the question of whether or not a given DFA \mathcal{A} is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathcal{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of \mathcal{A} .

The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing iff for every $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.*

Polynomial algorithm

Thus, the question of whether or not a given DFA \mathcal{A} is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathcal{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of \mathcal{A} . The following result by Černý gives a polynomial algorithm:

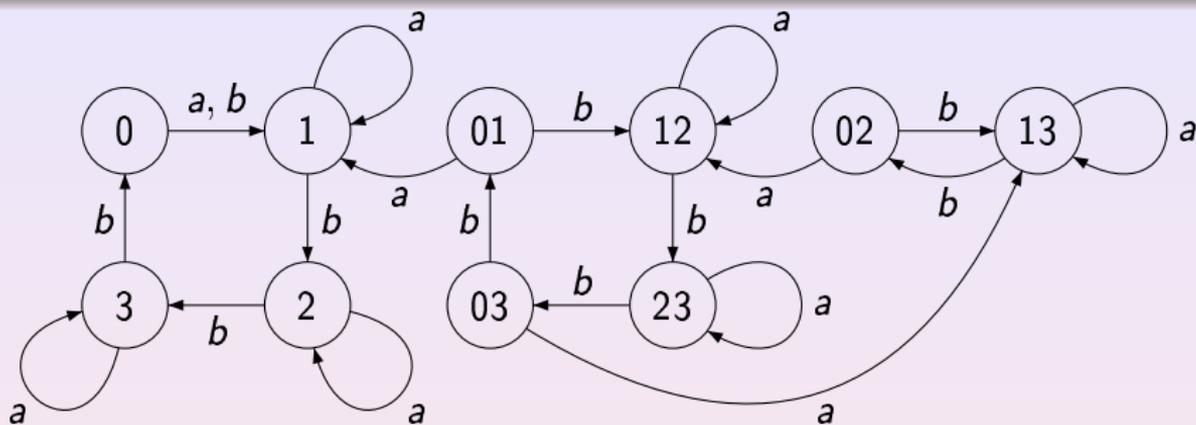
Proposition. *A DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing iff for every $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.*

Polynomial algorithm

Thus, the question of whether or not a given DFA \mathcal{A} is synchronizing reduces to the following reachability question in the underlying digraph of the power automaton $\mathcal{P}(\mathcal{A})$: is there a path from Q to a singleton? The latter question can be easily answered by BFS. This algorithm is however exponential w.r.t. the size of \mathcal{A} . The following result by Černý gives a polynomial algorithm:

Proposition. *A DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing iff for every $q, q' \in Q$ there exists a word $w \in \Sigma^*$ such that $\delta(q, w) = \delta(q', w)$.*

An example



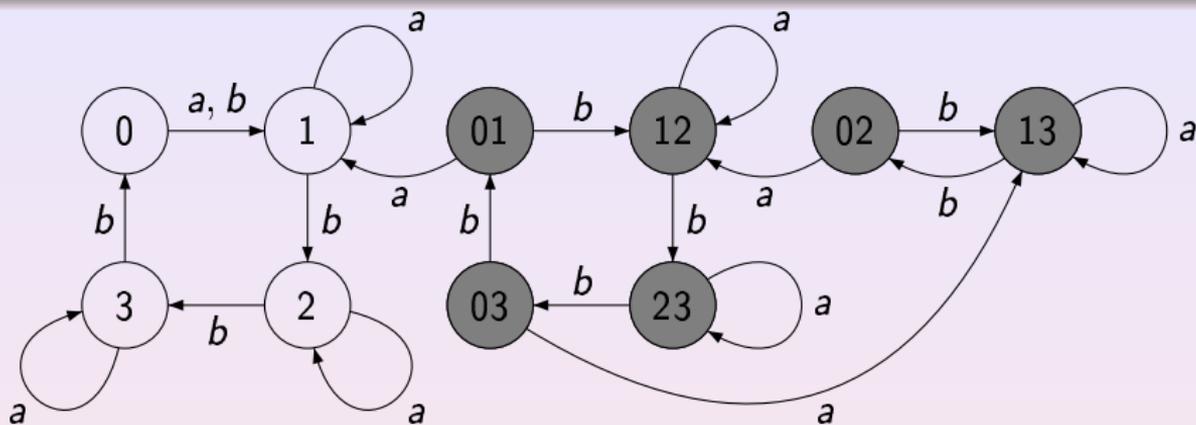
$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



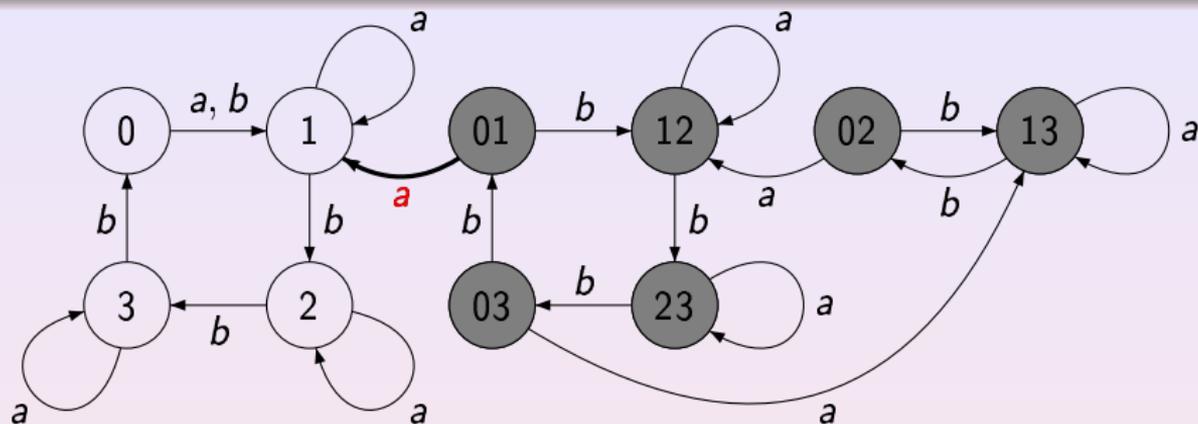
$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$

$abba \cdot babbba, Q \cdot abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



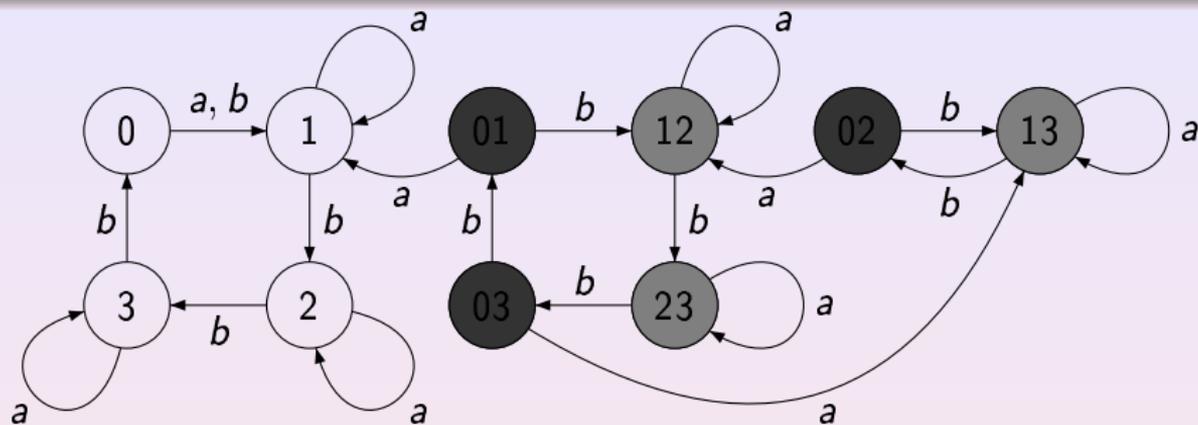
$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$

$abba \cdot babbba, Q \cdot abbababbba = \{1\}$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



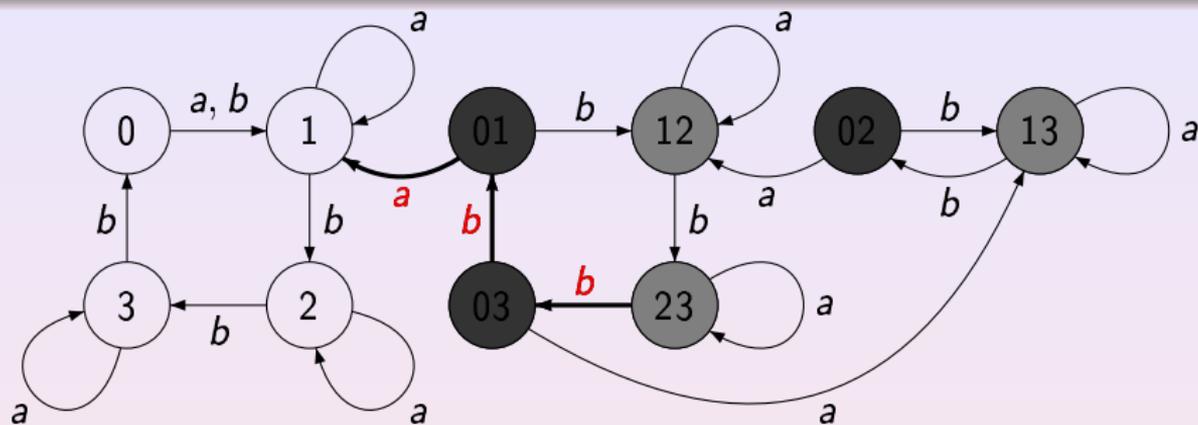
$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



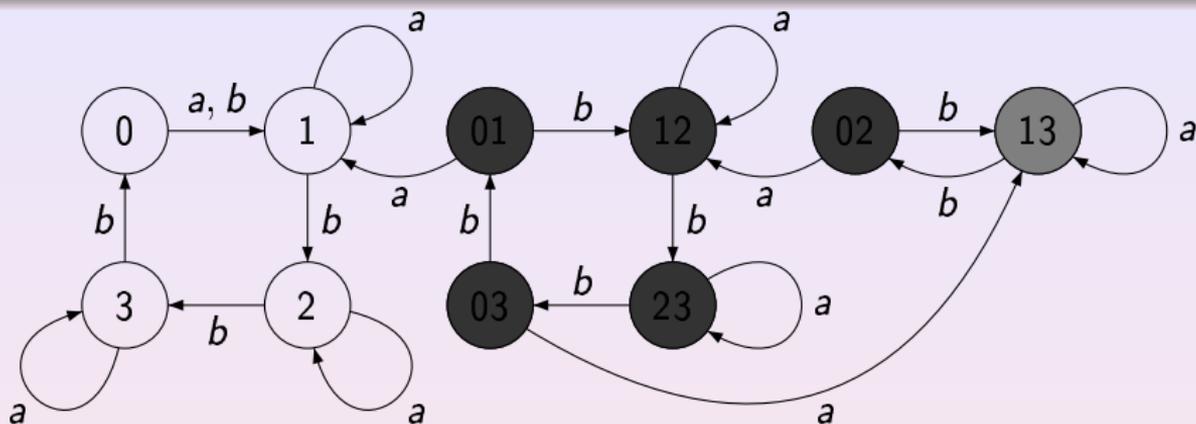
$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



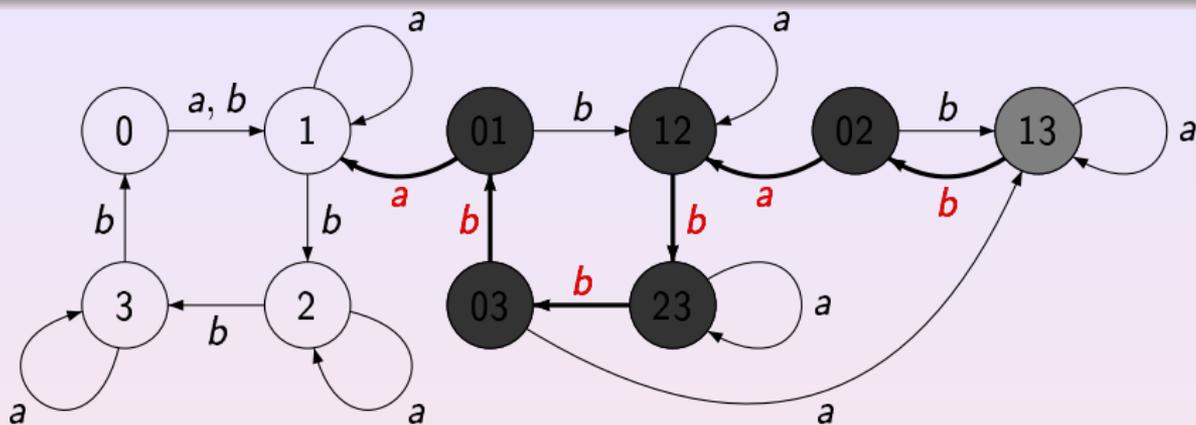
$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton

Niš, Nov 10, 2009

An example



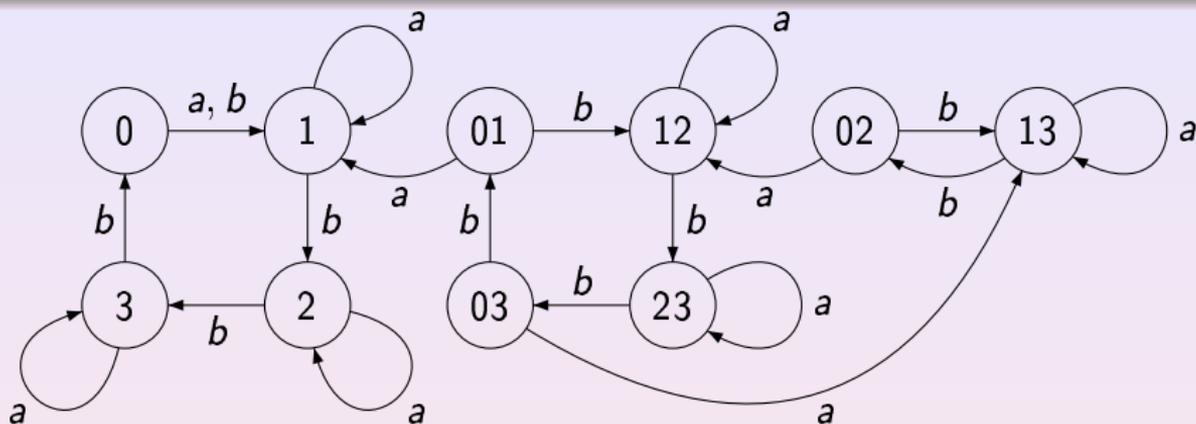
$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10 while we know a reset word of length 9 for this automaton.

Niš, Nov 10, 2009

An example



$$a, Q \cdot a = \{1, 2, 3\}; \quad a \cdot bba, Q \cdot abba = \{1, 3\}$$

$$abba \cdot babbba, Q \cdot abbababbba = \{1\}$$

Observe that the reset word constructed this way is of length 10
while we know a reset word of length 9 for this automaton.

Niš, Nov 10, 2009

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

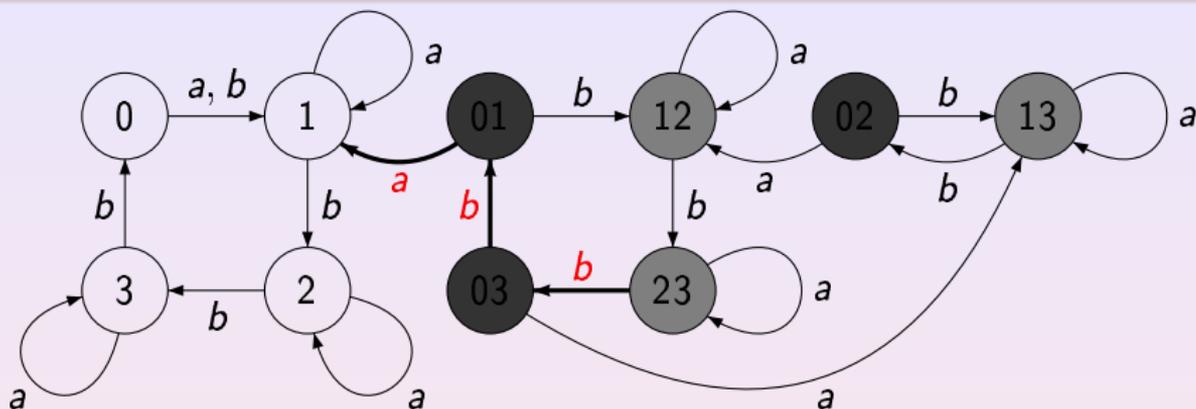
Results

Thus, recognizing synchronizability reduces to a reachability problem in the automaton whose states are the 2-subsets and the 1-subsets of Q . The latter can be solved by BFS in $O(n^2 \cdot |\Sigma|)$ time where $n = |Q|$.

If one also wants to produce a reset word, one needs $O(n^3 + n^2 \cdot |\Sigma|)$ time.

Clearly, the resulting reset word has length $O(n^3)$: the algorithm makes at most $n - 1$ steps and the length of the segment added in the step when k states are still to be compressed ($n \geq k \geq 2$) is at most $1 + \#$ of dark-grey 2-subsets, i.e. $1 + \binom{n}{2} - \binom{k}{2}$. This gives the upper bound $\frac{n^3 - n}{3}$. Can we do better? What is the exact bound?

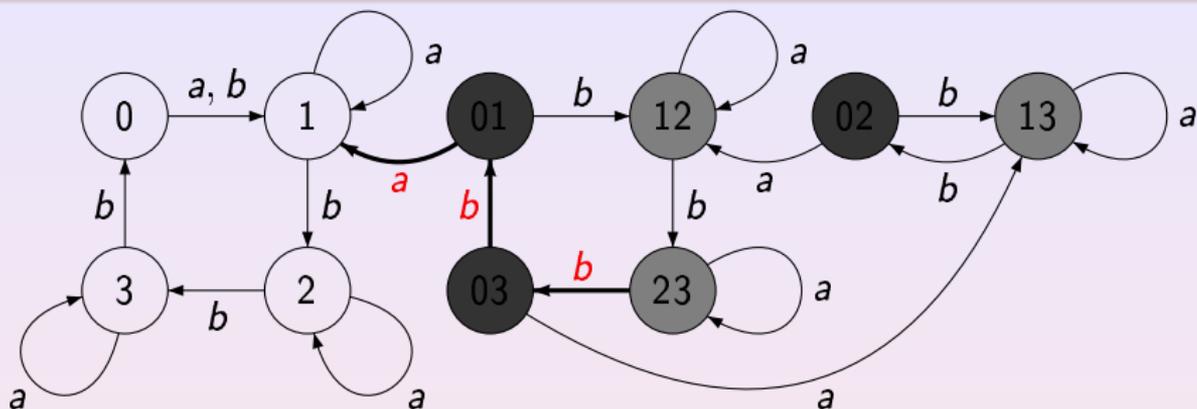
A resource for improvement



We see that the shortest path from a light-grey 2-subset to a singleton do not necessarily pass through all dark-grey 2-subsets. Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$. What is the minimum length of a word $v \in \Sigma^*$ such that $|P \cdot v| < k$?

Niš, Nov 10, 2009

A resource for improvement

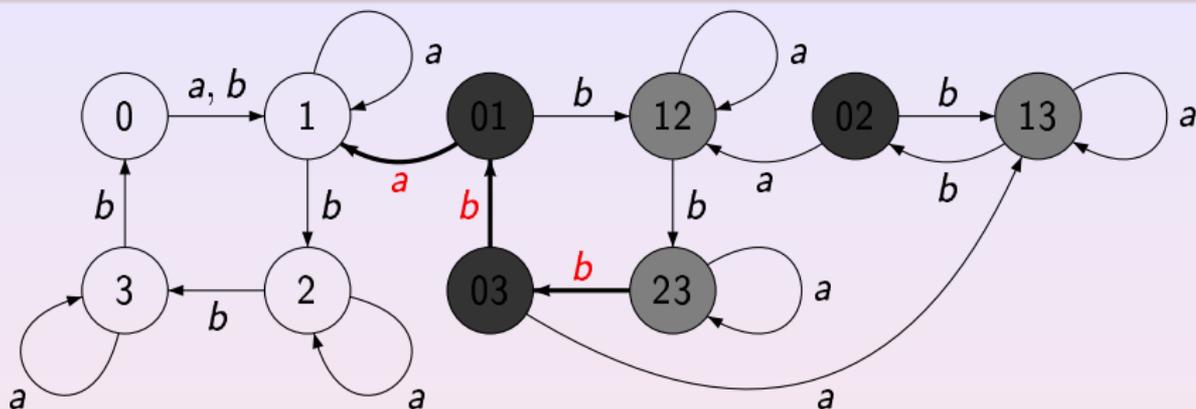


We see that the shortest path from a light-grey 2-subset to a singleton do not necessarily pass through all dark-grey 2-subsets.

Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$. What is the minimum length of a word $v \in \Sigma^*$ such that $|P \cdot v| < k$?

Niš, Nov 10, 2009

A resource for improvement



We see that the shortest path from a light-grey 2-subset to a singleton do not necessarily pass through all dark-grey 2-subsets. Consider a generic step of the algorithm at which states to be compressed form a set P with $|P| = k > 1$. What is the minimum length of a word $v \in \Sigma^*$ such that $|P \cdot v| < k$?

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\begin{aligned} & \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \dots = \binom{n+1}{3} = \frac{n^3-n}{6} \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\begin{aligned} \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \\ \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \\ \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \dots = \binom{n+1}{3} = \frac{n^3-n}{6} \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\begin{aligned} & \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \dots = \binom{n+1}{3} = \frac{n^3-n}{6}. \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\begin{aligned} \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \\ \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \\ \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} &= \dots = \binom{n+1}{3} = \frac{n^3-n}{6}. \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3-n}{6}$:

$$\begin{aligned} & \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \dots = \binom{n+1}{3} = \frac{n^3-n}{6}. \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3 - n}{6}$:

$$\begin{aligned} & \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \dots = \binom{n+1}{3} = \frac{n^3 - n}{6}. \end{aligned}$$

Niš, Nov 10, 2009

Results

In the step when k states are still to be compressed, the compression can always be achieved by applying a suitable word of length $\leq \binom{n-k+2}{2}$. Jean-Eric Pin, 1983, based on a non-trivial combinatorial result by Peter Frankl (An extremal problem for two families of sets, Eur. J. Comb., 3 (1982) 125–127).

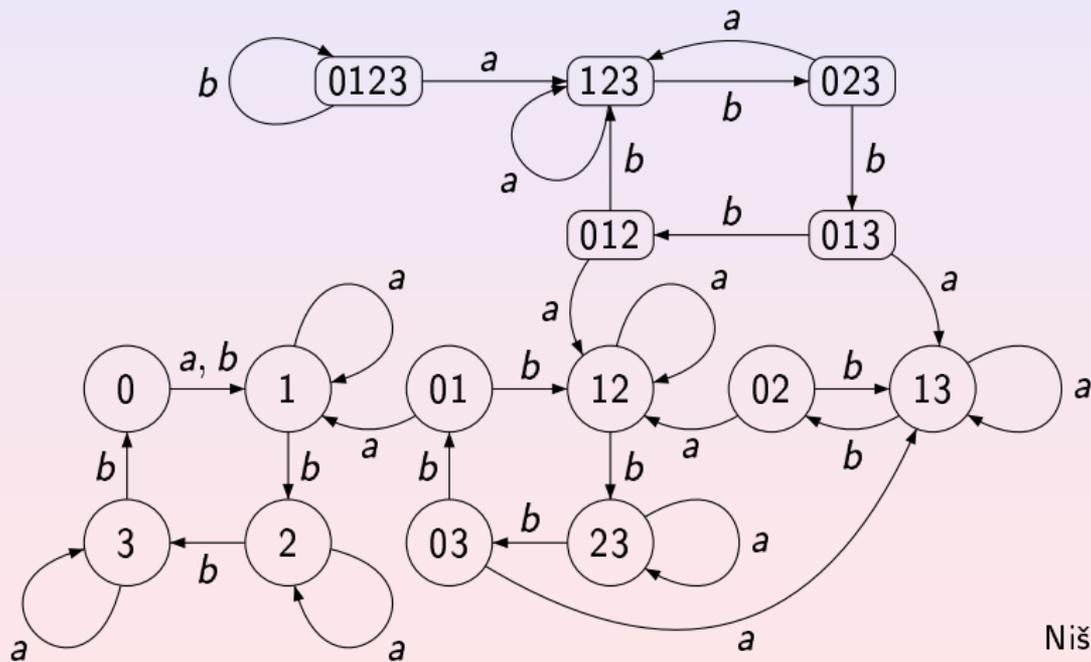
Summing up over $k = n, \dots, 2$, we see that the greedy algorithm always returns a reset word of length $\leq \frac{n^3 - n}{6}$:

$$\begin{aligned} & \binom{2}{2} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{3}{3} + \binom{3}{2} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \\ & \binom{4}{3} + \binom{4}{2} + \dots + \binom{n-1}{2} + \binom{n}{2} = \dots = \binom{n+1}{3} = \frac{n^3 - n}{6}. \end{aligned}$$

Niš, Nov 10, 2009

Example revisited

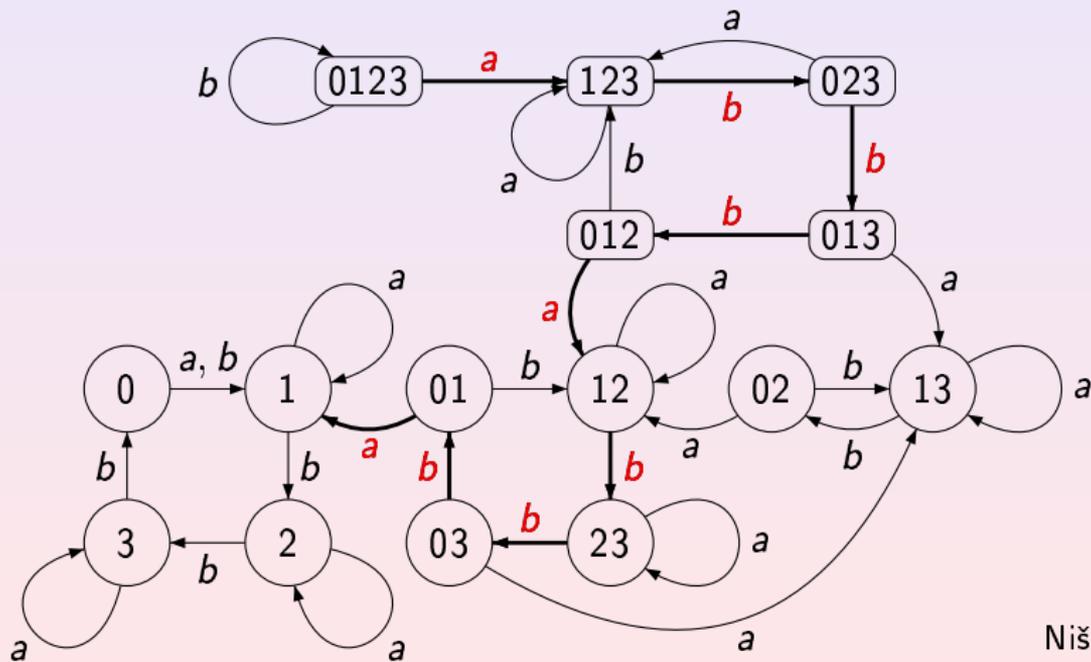
We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Niš, Nov 10, 2009

Example revisited

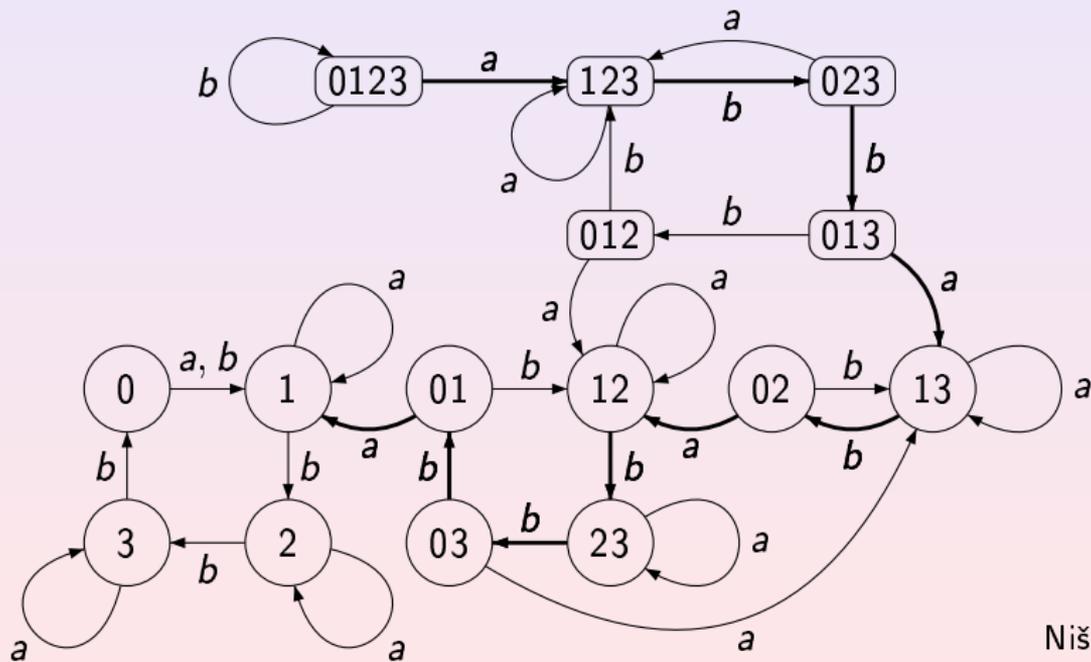
We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Niš, Nov 10, 2009

Example revisited

We have already seen that the greedy algorithm fails to find a reset word of minimum length.



Niš, Nov 10, 2009

Short reset words are hard to find

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n - 1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like $NP \neq coNP$) no polynomial algorithm, even non-deterministic, can find the minimum length of reset words for synchronizing automata.

Short reset words are hard to find

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n - 1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like $NP \neq coNP$) no polynomial algorithm, even non-deterministic, can find the minimum length of reset words for synchronizing automata.

Niš, Nov 10, 2009

Short reset words are hard to find

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n - 1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like $NP \neq coNP$) no polynomial algorithm, **even non-deterministic**, can find the minimum length of reset words for synchronizing automata.

Short reset words are hard to find

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n - 1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like $NP \neq coNP$) no polynomial algorithm, **even non-deterministic**, can find the minimum length of reset words for synchronizing automata.

Short reset words are hard to find

Actually, the gap between the minimum length of a reset word and the length of the word produced by the greedy algorithm may be arbitrarily large: for each $n > 1$ there exists a synchronizing automaton with n states whose shortest reset word has length $(n - 1)^2$ while the greedy algorithm produces a reset word of length $\Omega(n^2 \log n)$.

The behaviour of the greedy algorithm on average is not yet understood; practically it behaves rather well.

Under standard assumptions (like $NP \neq coNP$) no polynomial algorithm, **even non-deterministic**, can find the minimum length of reset words for synchronizing automata.

Non-approximability

However, all known results were consistent with the existence of very good polynomial approximation algorithms for the problem!

Recently, Mikhail Berlinkov, a PhD student of mine, has shown that under $NP \neq P$, for no k , there may exist a polynomial algorithm that, given a synchronizing automaton, produces a reset word whose length is less than $k \times$ minimum possible length of a reset word.

Niš, Nov 10, 2009

Non-approximability

However, all known results were consistent with the existence of very good polynomial approximation algorithms for the problem!

Recently, Mikhail Berlinkov, a PhD student of mine, has shown that under $NP \neq P$, for no k , there may exist a polynomial algorithm that, given a synchronizing automaton, produces a reset word whose length is less than $k \times$ minimum possible length of a reset word.

Niš, Nov 10, 2009

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series \mathcal{C}_n , $n = 2, 3, \dots$, of synchronizing automata over 2 letters.

The states of \mathcal{C}_n are the residues modulo n , and the input letters a and b act as follows:

$$\delta(0, a) = 1, \delta(m, a) = m \text{ for } 0 < m < n, \delta(m, b) = m+1 \pmod{n}.$$

The automaton used as an example above is \mathcal{C}_4 .

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series \mathcal{C}_n , $n = 2, 3, \dots$, of synchronizing automata over 2 letters.

The states of \mathcal{C}_n are the residues modulo n , and the input letters a and b act as follows:

$$\delta(0, a) = 1, \delta(m, a) = m \text{ for } 0 < m < n, \delta(m, b) = m+1 \pmod{n}.$$

The automaton used as an example above is \mathcal{C}_4 .

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series \mathcal{C}_n , $n = 2, 3, \dots$, of synchronizing automata over 2 letters.

The states of \mathcal{C}_n are the residues modulo n , and the input letters a and b act as follows:

$$\delta(0, a) = 1, \delta(m, a) = m \text{ for } 0 < m < n, \delta(m, b) = m+1 \pmod{n}.$$

The automaton used as an example above is \mathcal{C}_4 .

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series \mathcal{C}_n , $n = 2, 3, \dots$, of synchronizing automata over 2 letters.

The states of \mathcal{C}_n are the residues modulo n , and the input letters a and b act as follows:

$$\delta(0, a) = 1, \delta(m, a) = m \text{ for } 0 < m < n, \delta(m, b) = m+1 \pmod{n}.$$

The automaton used as an example above is \mathcal{C}_4 .

The Černý automata

Suppose a synchronizing automaton has n states. What is the length of its shortest reset word?

We know an upper bound: there always exists a reset word of length $\frac{n^3-n}{6}$. What about a lower bound?

In his 1964 paper Jan Černý constructed a series \mathcal{C}_n , $n = 2, 3, \dots$, of synchronizing automata over 2 letters.

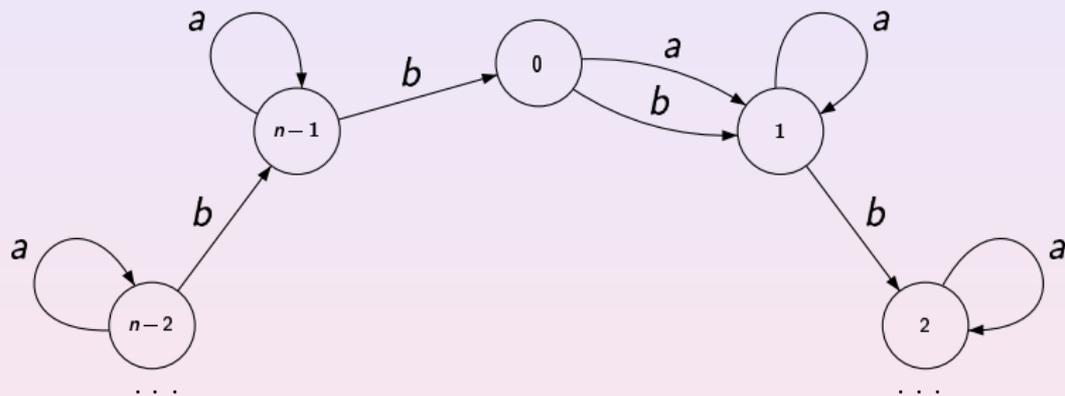
The states of \mathcal{C}_n are the residues modulo n , and the input letters a and b act as follows:

$$\delta(0, a) = 1, \delta(m, a) = m \text{ for } 0 < m < n, \delta(m, b) = m+1 \pmod{n}.$$

The automaton used as an example above is \mathcal{C}_4 .

The Černý automata

Here is a generic automaton from the Černý series:

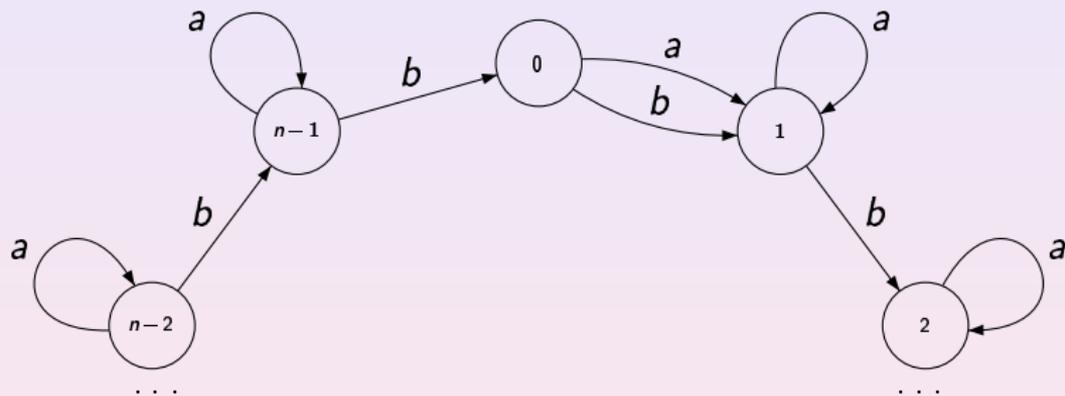


Černý has proved that the shortest reset word for \mathcal{C}_n is $(ab^{n-1})^{n-2}a$ of length $(n-1)^2$. As other results from Černý's paper of 1964, this nice series of automata has been rediscovered many times.

Niš, Nov 10, 2009

The Černý automata

Here is a generic automaton from the Černý series:

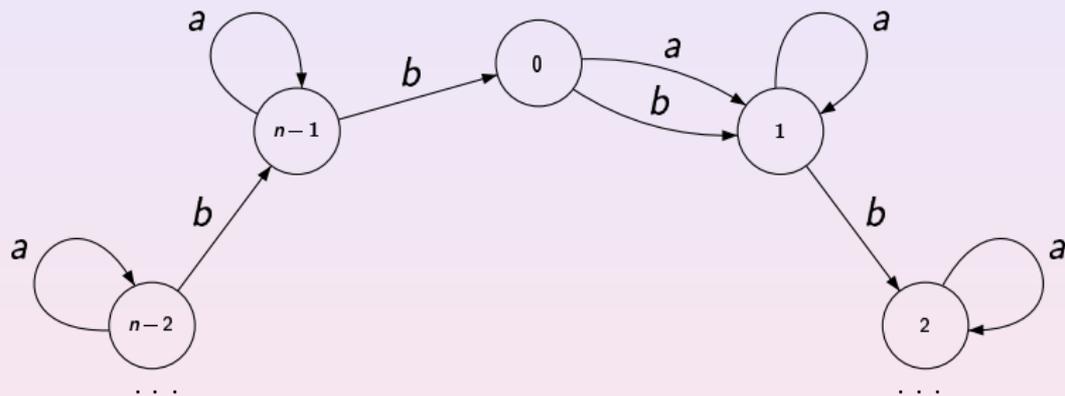


Černý has proved that the shortest reset word for \mathcal{C}_n is $(ab^{n-1})^{n-2}a$ of length $(n-1)^2$. As other results from Černý's paper of 1964, this nice series of automata has been rediscovered many times.

Niš, Nov 10, 2009

The Černý automata

Here is a generic automaton from the Černý series:



Černý has proved that the shortest reset word for \mathcal{C}_n is $(ab^{n-1})^{n-2}a$ of length $(n-1)^2$. As other results from Černý's paper of 1964, this nice series of automata has been rediscovered many times.

Niš, Nov 10, 2009

The Černý function

Define the Černý function $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathcal{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n - 1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n - 1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata. Everything we know about the conjecture in general can be summarized in one line:

$$(n - 1)^2 \leq C(n) \leq \frac{n^3 - n}{6}.$$

Niš, Nov 10, 2009

The Černý function

Define the Černý function $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathcal{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n - 1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n - 1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata. Everything we know about the conjecture in general can be summarized in one line:

$$(n - 1)^2 \leq C(n) \leq \frac{n^3 - n}{6}.$$

The Černý function

Define the Černý function $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathcal{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n - 1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n - 1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata. Everything we know about the conjecture in general can be summarized in one line:

$$(n - 1)^2 \leq C(n) \leq \frac{n^3 - n}{6}.$$

The Černý function

Define the Černý function $C(n)$ as the maximum length of shortest reset words for synchronizing automata with n states. The above property of the series $\{\mathcal{C}_n\}$, $n = 2, 3, \dots$, yields the inequality $C(n) \geq (n - 1)^2$.

The Černý conjecture is the claim that in fact the equality $C(n) = (n - 1)^2$ holds true. This simply looking conjecture is arguably the most longstanding open problem in the combinatorial theory of finite automata. Everything we know about the conjecture in general can be summarized in one line:

$$(n - 1)^2 \leq C(n) \leq \frac{n^3 - n}{6}.$$

A discussion

Why is the problem so surprisingly difficult?

- **non-locality**: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails);
- **combinatorics of finite sets** is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare. The only known infinite series of n -state synchronizing automata with shortest reset words of length $(n-1)^2$ is the Černý series \mathcal{C}_n , $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

Niš, Nov 10, 2009

A discussion

Why is the problem so surprisingly difficult?

- **non-locality**: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails);
- **combinatorics of finite sets** is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare. The only known infinite series of n -state synchronizing automata with shortest reset words of length $(n - 1)^2$ is the Černý series \mathcal{C}_n , $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

Niš, Nov 10, 2009

A discussion

Why is the problem so surprisingly difficult?

- **non-locality**: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails);
- **combinatorics of finite sets** is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare. The only known infinite series of n -state synchronizing automata with shortest reset words of length $(n - 1)^2$ is the Černý series \mathcal{C}_n , $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

Niš, Nov 10, 2009

A discussion

Why is the problem so surprisingly difficult?

- **non-locality**: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails);
- **combinatorics of finite sets** is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare. The only known infinite series of n -state synchronizing automata with shortest reset words of length $(n - 1)^2$ is the Černý series \mathcal{C}_n , $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

Niš, Nov 10, 2009

A discussion

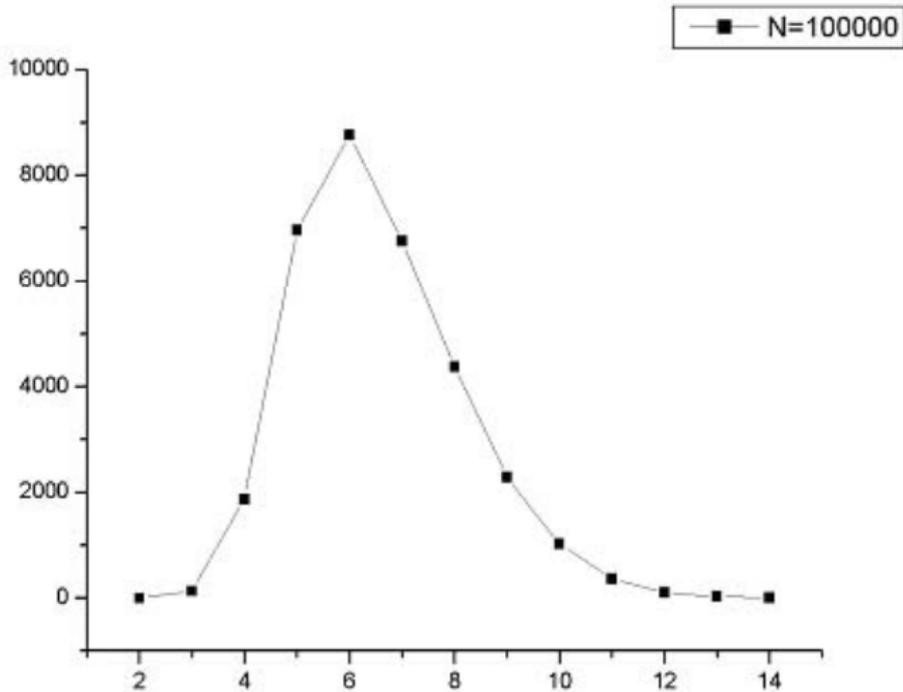
Why is the problem so surprisingly difficult?

- **non-locality**: prefixes of optimal solutions need not be optimal (that's why the greedy algorithm fails);
- **combinatorics of finite sets** is encoded in the problem.

Yet another reason: “slowly” synchronizing automata turn out to be extremely rare. The only known infinite series of n -state synchronizing automata with shortest reset words of length $(n - 1)^2$ is the Černý series \mathcal{C}_n , $n = 2, 3, \dots$, with a few sporadic examples for $n \leq 6$.

Niš, Nov 10, 2009

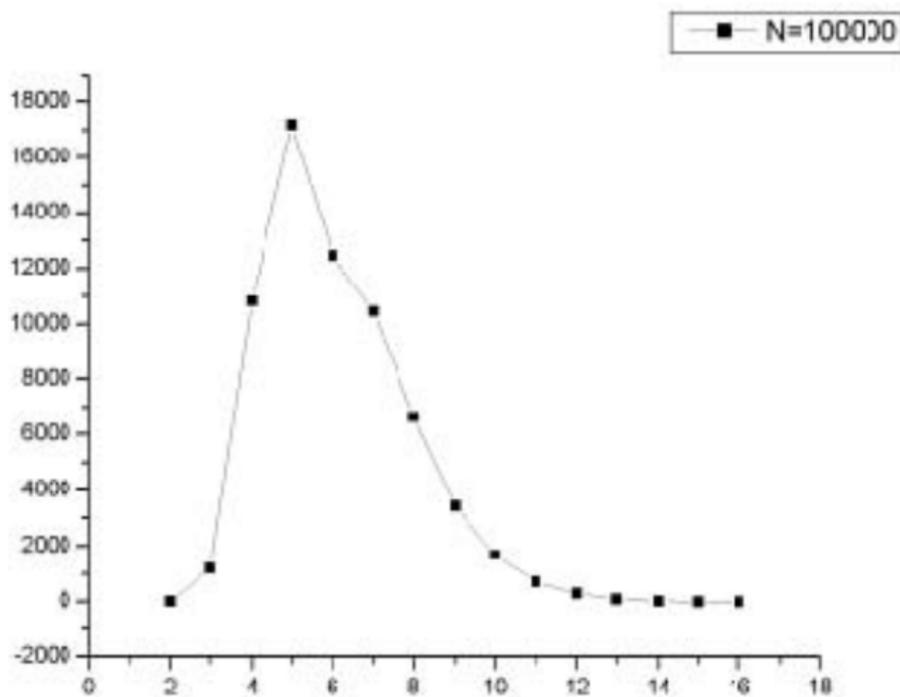
20-State Experiment



10, 2009



30-State Experiment



10, 2009



Random Automata

A (partial) explanation of these experimental observations: if Q is an n -set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\geq 2n$.

Thus, “slowly” synchronizing automata cannot be discovered via a random sampling.

Niš, Nov 10, 2009

Random Automata

A (partial) explanation of these experimental observations: if Q is an n -set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\geq 2n$.

Thus, “slowly” synchronizing automata cannot be discovered via a random sampling.

Niš, Nov 10, 2009

Random Automata

A (partial) explanation of these experimental observations: if Q is an n -set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\geq 2n$.

Thus, “slowly” synchronizing automata cannot be discovered via a random sampling.

Niš, Nov 10, 2009

Random Automata

A (partial) explanation of these experimental observations: if Q is an n -set (with n large enough), then, on average, any product of $2n$ randomly chosen transformations of Q is a constant map (Peter Higgins, The range order of a product of i transformations from a finite full transformation semigroup, Semigroup Forum, 37 (1988) 31–36). In automata-theoretic terms, this fact means that a randomly chosen DFA with n states and a sufficiently large input alphabet tends to be synchronizing and is reset by any word of length $\geq 2n$.

Thus, “slowly” synchronizing automata cannot be discovered via a random sampling.

Sporadic Examples: $n = 2$

A synchronizing automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from \mathcal{A} by erasing any letter in Σ are synchronizing. E.g., the Černý automata \mathcal{C}_n with $n > 2$ are proper while \mathcal{C}_2 is not.

A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n - 1)^2$. We present here all known proper synchronizing automata beyond the Černý series \mathcal{C}_n , $n = 3, 4, \dots$, that reach the Černý bound.

For the sake of completeness, we start with $n = 2$:

Sporadic Examples: $n = 2$

A synchronizing automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from \mathcal{A} by erasing any letter in Σ are synchronizing. E.g., the Černý automata \mathcal{C}_n with $n > 2$ are proper while \mathcal{C}_2 is not.

A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n - 1)^2$. We present here all known proper synchronizing automata beyond the Černý series \mathcal{C}_n , $n = 3, 4, \dots$, that reach the Černý bound.

For the sake of completeness, we start with $n = 2$:

Sporadic Examples: $n = 2$

A synchronizing automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from \mathcal{A} by erasing any letter in Σ are synchronizing. E.g., the Černý automata \mathcal{C}_n with $n > 2$ are proper while \mathcal{C}_2 is not.

A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n - 1)^2$. We present here all known proper synchronizing automata beyond the Černý series \mathcal{C}_n , $n = 3, 4, \dots$, that reach the Černý bound.

For the sake of completeness, we start with $n = 2$:

Sporadic Examples: $n = 2$

A synchronizing automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from \mathcal{A} by erasing any letter in Σ are synchronizing. E.g., the Černý automata \mathcal{C}_n with $n > 2$ are proper while \mathcal{C}_2 is not.

A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n - 1)^2$. We present here all known proper synchronizing automata beyond the Černý series \mathcal{C}_n , $n = 3, 4, \dots$, that reach the Černý bound.

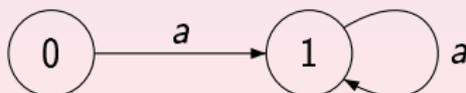
For the sake of completeness, we start with $n = 2$:

Sporadic Examples: $n = 2$

A synchronizing automaton $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is *proper* if none of the automata obtained from \mathcal{A} by erasing any letter in Σ are synchronizing. E.g., the Černý automata \mathcal{C}_n with $n > 2$ are proper while \mathcal{C}_2 is not.

A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of its reset words is $(n - 1)^2$. We present here all known proper synchronizing automata beyond the Černý series \mathcal{C}_n , $n = 3, 4, \dots$, that reach the Černý bound.

For the sake of completeness, we start with $n = 2$:



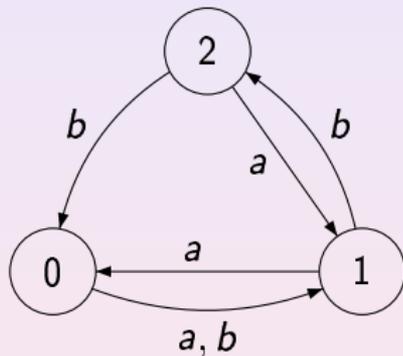
Sporadic Examples: $n = 3$

For $n = 3$ we have three sporadic automata:

Niš, Nov 10, 2009

Sporadic Examples: $n = 3$

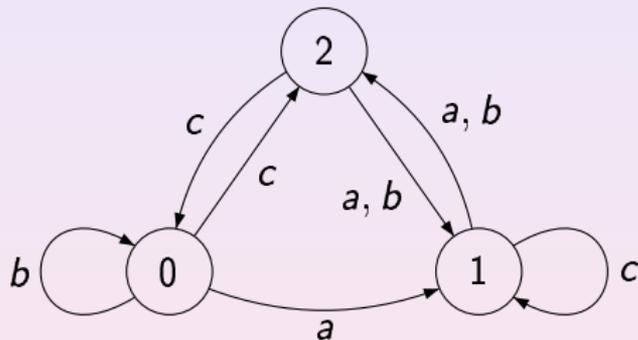
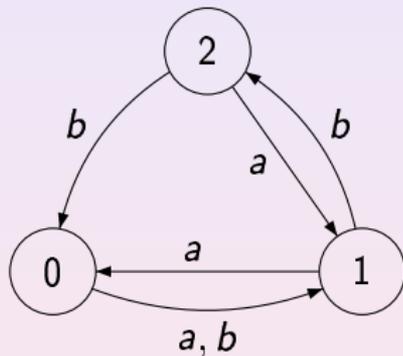
For $n = 3$ we have three sporadic automata:



Niš, Nov 10, 2009

Sporadic Examples: $n = 3$

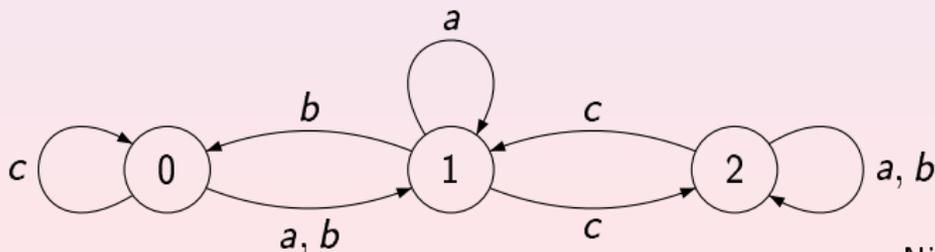
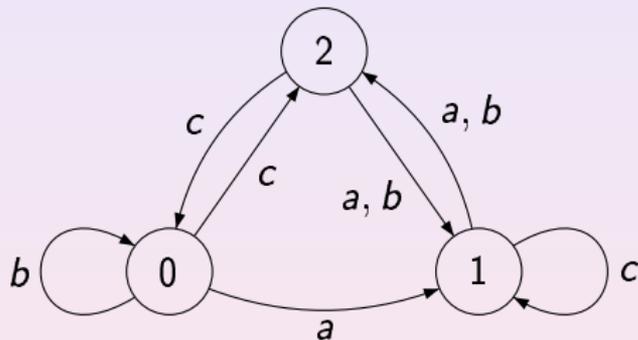
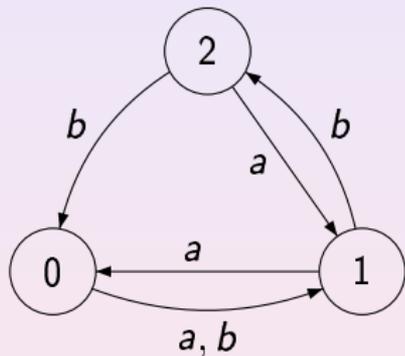
For $n = 3$ we have three sporadic automata:



Niš, Nov 10, 2009

Sporadic Examples: $n = 3$

For $n = 3$ we have three sporadic automata:



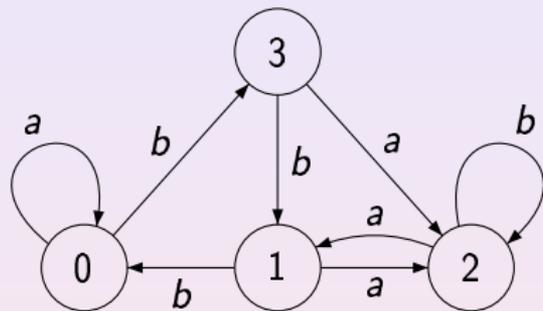
Niš, Nov 10, 2009

Sporadic Examples: $n = 4$

Also for $n = 4$ three sporadic automata are known:

Sporadic Examples: $n = 4$

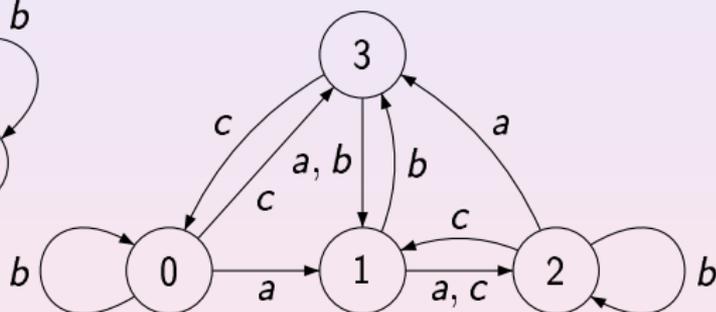
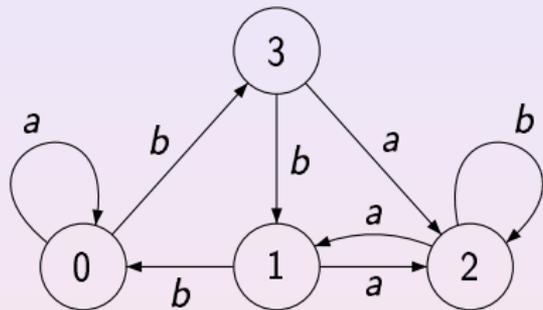
Also for $n = 4$ three sporadic automata are known:



Niš, Nov 10, 2009

Sporadic Examples: $n = 4$

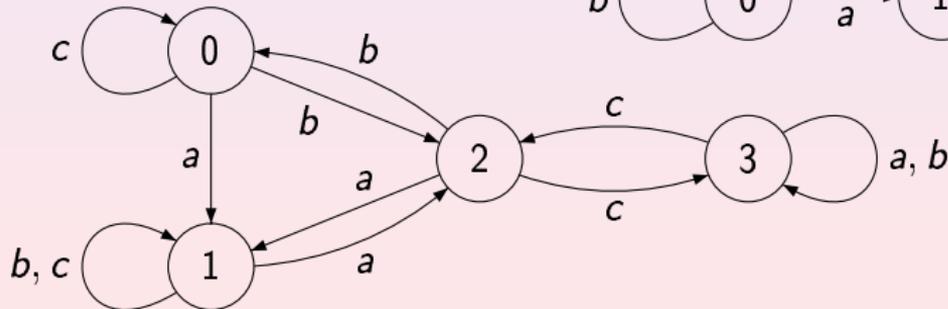
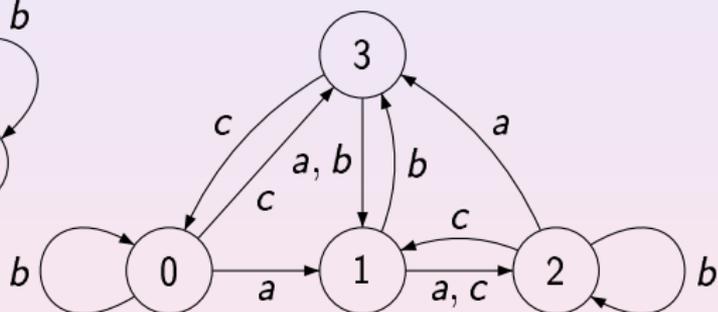
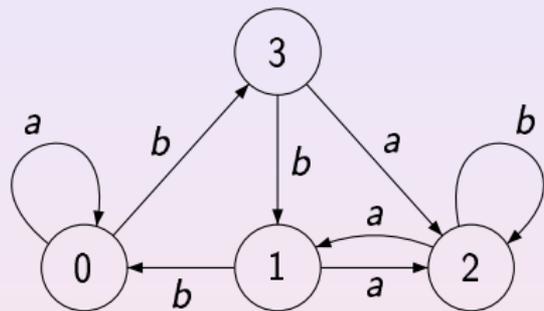
Also for $n = 4$ three sporadic automata are known:



Niš, Nov 10, 2009

Sporadic Examples: $n = 4$

Also for $n = 4$ three sporadic automata are known:



Niš, Nov 10, 2009

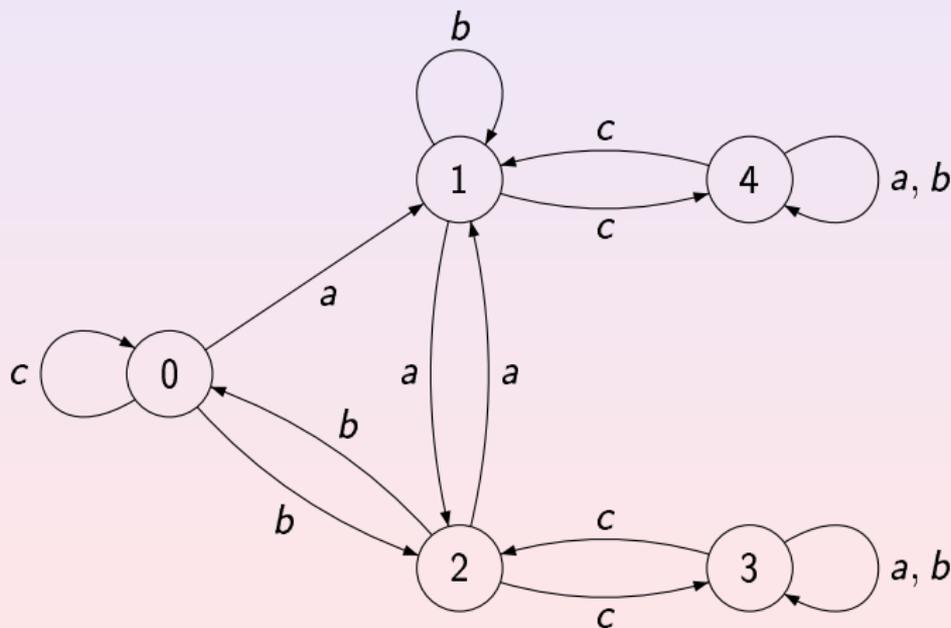
Roman's automaton

A proper 5-state automaton reaching the Černý bound has been recently discovered by Adam Roman.

Niš, Nov 10, 2009

Roman's automaton

A proper 5-state automaton reaching the Černý bound has been recently discovered by Adam Roman.



Niš, Nov 10, 2009

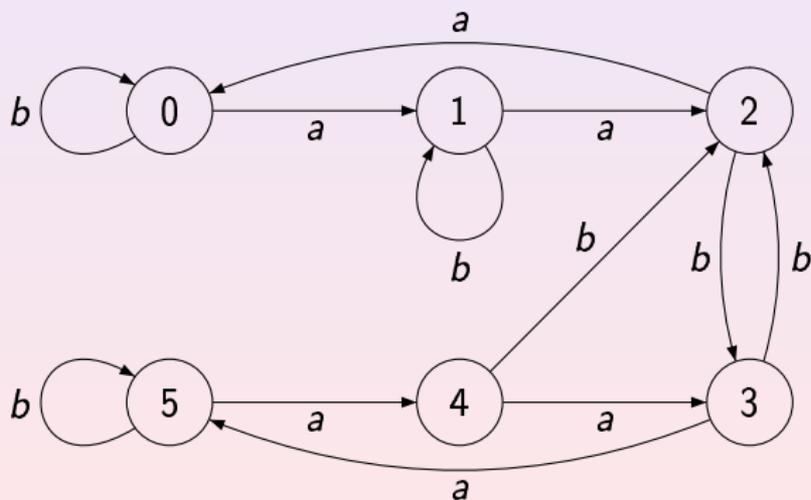
Kari's automaton

The last in our list and the most remarkable example was published in 2001 by Jarkko Kari (A counter example to a conjecture concerning synchronizing words in finite automata, EATCS Bull., 73, 146).

Niš, Nov 10, 2009

Kari's automaton

The last in our list and the most remarkable example was published in 2001 by Jarkko Kari (A counter example to a conjecture concerning synchronizing words in finite automata, EATCS Bull., 73, 146).



Niš, Nov 10, 2009

Pin's conjecture

Kari's automaton \mathcal{K}_6 has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \cdot w| = k$, $1 \leq k \leq n$, then \mathcal{A} possesses a word of length at most $(n - k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

However, in \mathcal{K}_6 there is no word w of length $16 = (6 - 2)^2$ such that $|Q \cdot w| = 2$.

Niš, Nov 10, 2009

Pin's conjecture

Kari's automaton \mathcal{K}_6 has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \cdot w| = k$, $1 \leq k \leq n$, then \mathcal{A} possesses a word of length at most $(n - k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

However, in \mathcal{K}_6 there is no word w of length $16 = (6 - 2)^2$ such that $|Q \cdot w| = 2$.

Pin's conjecture

Kari's automaton \mathcal{K}_6 has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \cdot w| = k$, $1 \leq k \leq n$, then \mathcal{A} possesses a word of length at most $(n - k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

However, in \mathcal{K}_6 there is no word w of length $16 = (6 - 2)^2$ such that $|Q \cdot w| = 2$.

Pin's conjecture

Kari's automaton \mathcal{K}_6 has refuted several conjectures.

The most well known of them was suggested by Jean-Eric Pin in 1978. Pin conjectured that if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states admits a word $w \in \Sigma^*$ such that $|Q \cdot w| = k$, $1 \leq k \leq n$, then \mathcal{A} possesses a word of length at most $(n - k)^2$ with the same property. (The Černý conjecture corresponds to the case $k = 1$.)

However, in \mathcal{K}_6 there is no word w of length $16 = (6 - 2)^2$ such that $|Q \cdot w| = 2$.

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does not refute the rank conjecture!

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does not refute the rank conjecture!

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does not refute the rank conjecture!

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does **not** refute the rank conjecture!

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does **not** refute the rank conjecture!

Rank conjecture

The *rank* of a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is the minimum cardinality of the sets $Q \cdot w$ where w runs over Σ^* . Synchronizing automata are precisely those of rank 1.

A corrected (and perhaps correct) version of Pin's conjecture is the following **rank conjecture**: if a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ with n states has rank k , then there exists a word $w \in \Sigma^*$ of length at most $(n - k)^2$ such that $|Q \cdot w| = k$. Again, the Černý conjecture corresponds to the case $k = 1$.

Kari's automaton does **not** refute the rank conjecture!

Extensibility conjecture

Yet another hope killed by Kari's example is the **extensibility conjecture**. In a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$, a subset $P \subset Q$ is *extensible* if $P = R \cdot w$ for some $w \in \Sigma^*$ of length at most $n = |Q|$ and some $R \subseteq Q$ with $|R| > |P|$. It was conjectured that in synchronizing automata every proper non-singleton subset is extensible.

Niš, Nov 10, 2009

Extensibility conjecture

Yet another hope killed by Kari's example is the **extensibility conjecture**. In a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$, a subset $P \subset Q$ is *extensible* if $P = R \cdot w$ for some $w \in \Sigma^*$ of length at most $n = |Q|$ and some $R \subseteq Q$ with $|R| > |P|$. It was conjectured that in synchronizing automata every proper non-singleton subset is extensible.

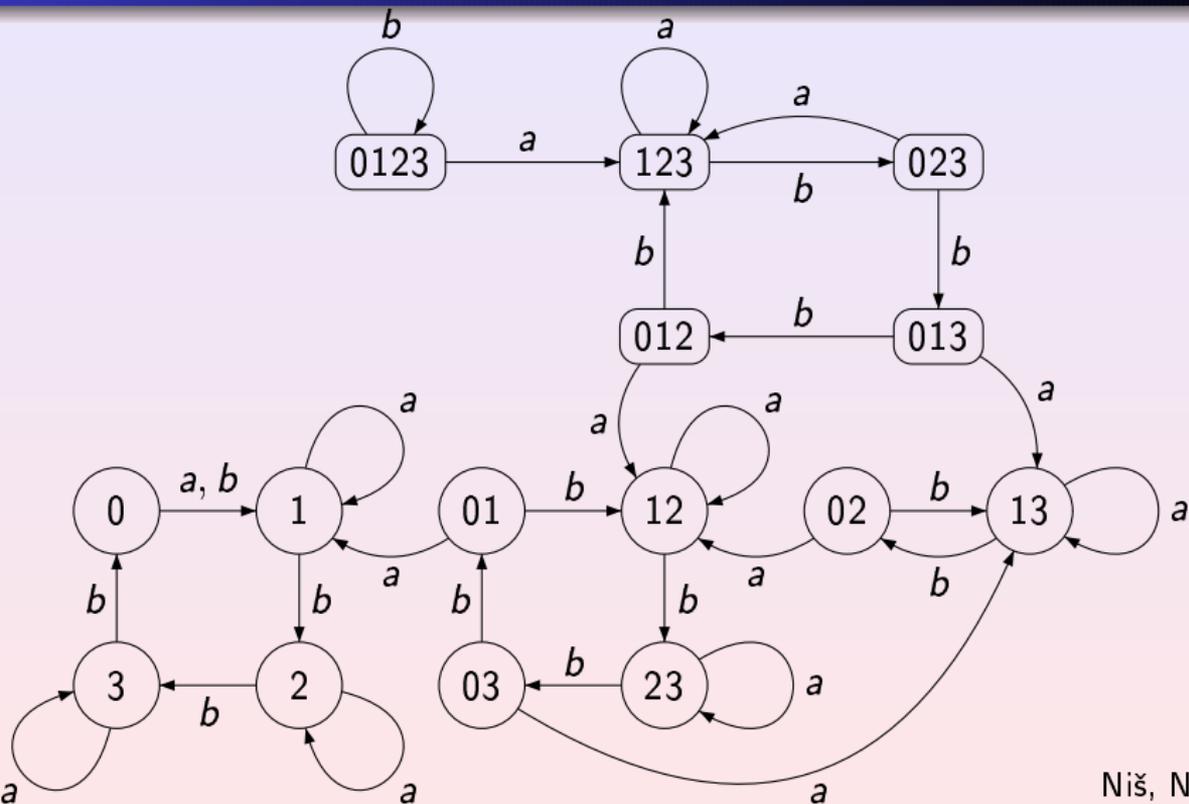
Niš, Nov 10, 2009

Extensibility conjecture

Yet another hope killed by Kari's example is the **extensibility conjecture**. In a DFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$, a subset $P \subset Q$ is *extensible* if $P = R \cdot w$ for some $w \in \Sigma^*$ of length at most $n = |Q|$ and some $R \subseteq Q$ with $|R| > |P|$. It was conjectured that in synchronizing automata every proper non-singleton subset is extensible.

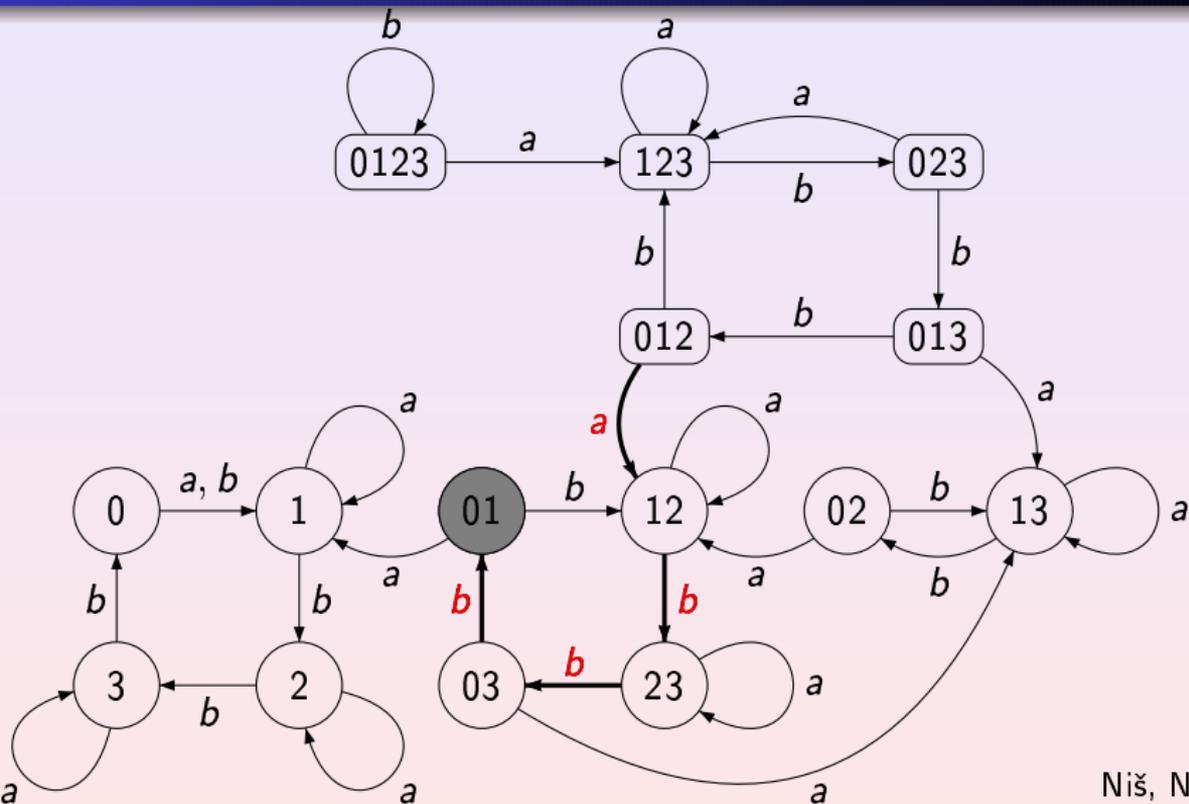
Niš, Nov 10, 2009

Extensibility



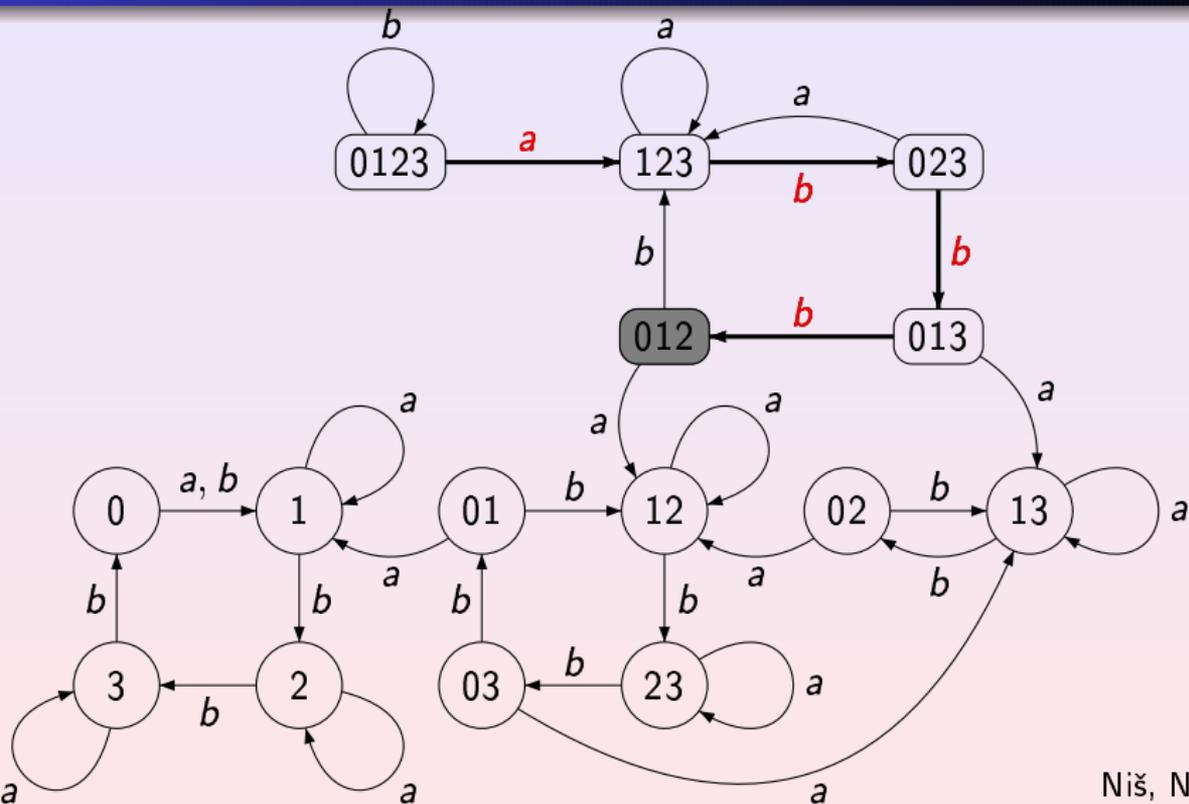
Niš, Nov 10, 2009

Extensibility



Niš, Nov 10, 2009

Extensibility



Niš, Nov 10, 2009

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should send two states $q, q' \in Q$ to the same state p . Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \cdot w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence P_0, P_1, P_2, \dots reaches Q and

$$Q \cdot w_{n-1} w_{n-2} \cdots w_1 a = \{p\},$$

that is, $w_{n-1} w_{n-2} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n-2) + 1 = (n-1)^2$.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should send two states $q, q' \in Q$ to the same state p . Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \cdot w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence P_0, P_1, P_2, \dots reaches Q and

$$Q \cdot w_{n-1} w_{n-2} \cdots w_1 a = \{p\},$$

that is, $w_{n-1} w_{n-2} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n - 2) + 1 = (n - 1)^2$.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should send two states $q, q' \in Q$ to the same state p . Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \cdot w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence P_0, P_1, P_2, \dots reaches Q and

$$Q \cdot w_{n-1} w_{n-2} \cdots w_1 a = \{p\},$$

that is, $w_{n-1} w_{n-2} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n - 2) + 1 = (n - 1)^2$.

Extensibility

Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p . Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \cdot w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence P_0, P_1, P_2, \dots reaches Q and

$$Q \cdot w_{n-1} w_{n-2} \cdots w_1 a = \{p\},$$

that is, $w_{n-1} w_{n-2} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n - 2) + 1 = (n - 1)^2$.

Extensibility

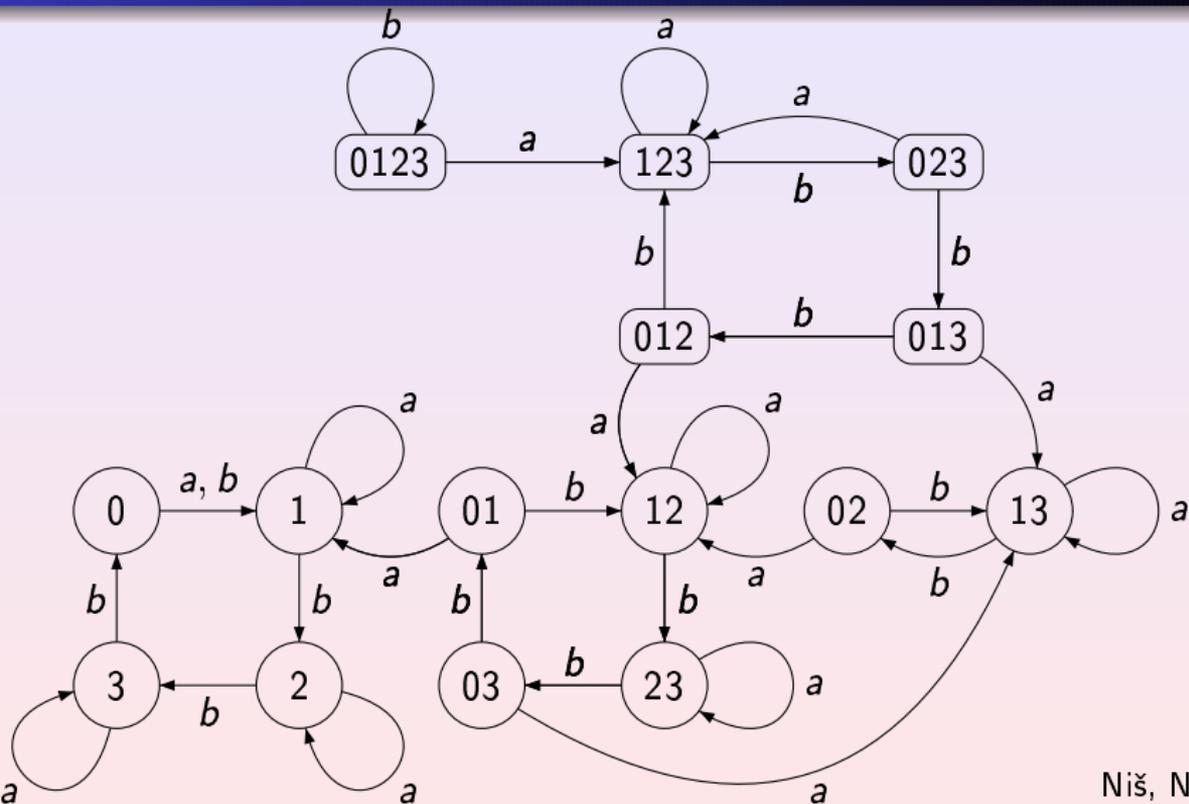
Observe that the extensibility conjecture implies the Černý conjecture.

Indeed, if $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is synchronizing, then some letter $a \in \Sigma$ should sent two states $q, q' \in Q$ to the same state p . Let $P_0 = \{q, q'\}$ and, for $i > 0$, let P_i be such that $|P_i| > |P_{i-1}|$ and $P_{i-1} = P_i \cdot w_i$ for some word w_i of length $\leq n$. Then in at most $n - 2$ steps the sequence P_0, P_1, P_2, \dots reaches Q and

$$Q \cdot w_{n-1} w_{n-2} \cdots w_1 a = \{p\},$$

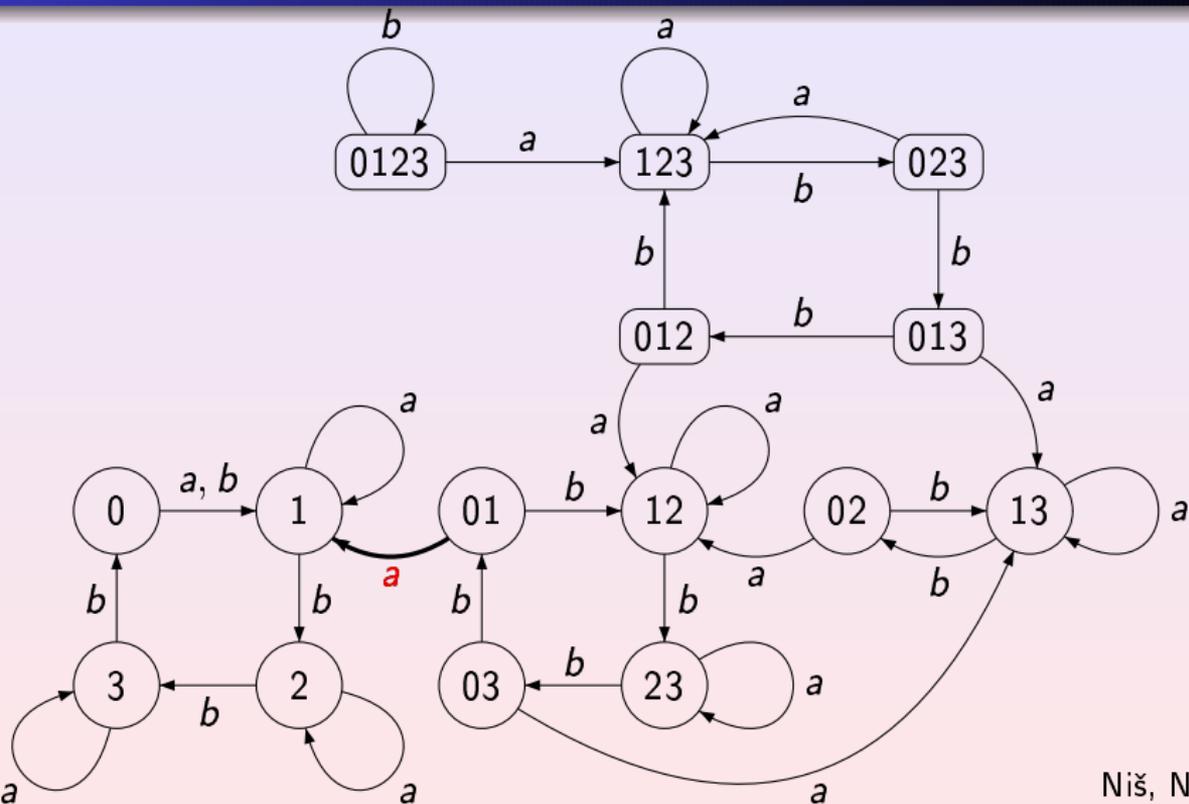
that is, $w_{n-1} w_{n-2} \cdots w_1 a$ is a reset word. The length of this reset word is at most $n(n - 2) + 1 = (n - 1)^2$.

Extensibility



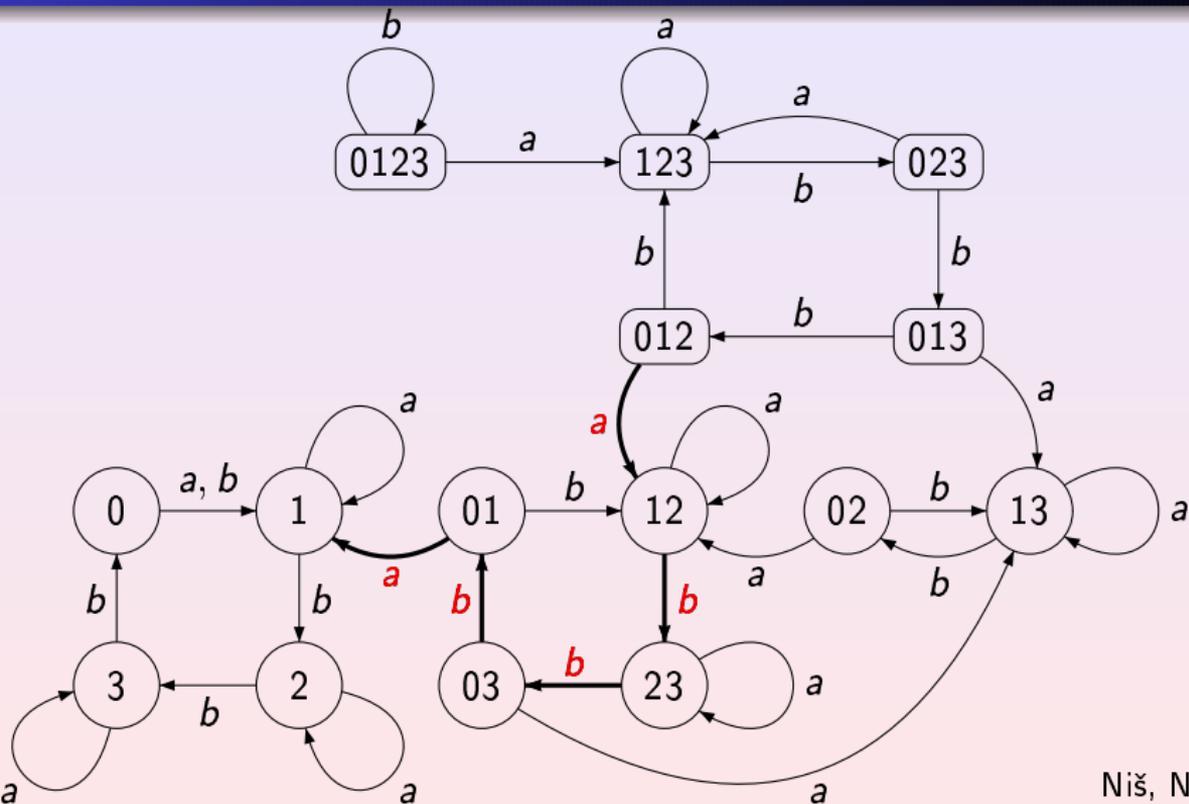
Niš, Nov 10, 2009

Extensibility



Niš, Nov 10, 2009

Extensibility



Niš, Nov 10, 2009

Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

- Louis Dubuc's result for automata in which a letter acts on the state set Q as a cyclic permutation of order $|Q|$ (Sur le automates circulaires et la conjecture de Černý, RAIRO Inform. Theor. Appl., 32 (1998) 21–34 [in French]).
- Jarkko Kari's result for automata with Eulerian digraphs (Synchronizing finite automata on Eulerian digraphs, Theoret. Comput. Sci., 295 (2003) 223–232.)

Niš, Nov 10, 2009

Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

- Louis Dubuc's result for automata in which a letter acts on the state set Q as a cyclic permutation of order $|Q|$ (Sur le automates circulaires et la conjecture de Černý, RAIRO Inform. Theor. Appl., 32 (1998) 21–34 [in French]).
- Jarkko Kari's result for automata with Eulerian digraphs (Synchronizing finite automata on Eulerian digraphs, Theoret. Comput. Sci., 295 (2003) 223–232.)

Extensibility

Several important results confirming the Černý conjecture for various partial cases have been proved by verifying the extensibility conjecture for the corresponding automata. This includes, in particular:

- Louis Dubuc's result for automata in which a letter acts on the state set Q as a cyclic permutation of order $|Q|$ (Sur le automates circulaires et la conjecture de Černý, RAIRO Inform. Theor. Appl., 32 (1998) 21–34 [in French]).
- Jarkko Kari's result for automata with Eulerian digraphs (Synchronizing finite automata on Eulerian digraphs, Theoret. Comput. Sci., 295 (2003) 223–232.)

Extensibility vs Kari's Example

However, in \mathcal{H}_6 there exists a 2-subset that cannot be extended to a larger subset by any word of length 6 (and even by any word of length 7).

Thus, the extensibility conjecture fails, and the approach based on it cannot prove the Černý conjecture in general.

However, studying the extensibility phenomenon in synchronizing automata appears to be worthwhile: if there is a **linear** bound on the minimum length of words extending non-singleton proper subsets of a synchronizing automaton, then there is a **quadratic** bound on the minimum length of reset words for the automaton.

Extensibility vs Kari's Example

However, in \mathcal{H}_6 there exists a 2-subset that cannot be extended to a larger subset by any word of length 6 (and even by any word of length 7).

Thus, the extensibility conjecture fails, and the approach based on it cannot prove the Černý conjecture in general.

However, studying the extensibility phenomenon in synchronizing automata appears to be worthwhile: if there is a **linear** bound on the minimum length of words extending non-singleton proper subsets of a synchronizing automaton, then there is a **quadratic** bound on the minimum length of reset words for the automaton.

Extensibility vs Kari's Example

However, in \mathcal{H}_6 there exists a 2-subset that cannot be extended to a larger subset by any word of length 6 (and even by any word of length 7).

Thus, the extensibility conjecture fails, and the approach based on it cannot prove the Černý conjecture in general.

However, studying the extensibility phenomenon in synchronizing automata appears to be worthwhile: if there is a **linear** bound on the minimum length of words extending non-singleton proper subsets of a synchronizing automaton, then there is a **quadratic** bound on the minimum length of reset words for the automaton.