Synchronizing Automata Preserving a Chain of Partial Orders

M. V. Volkov

Ural State University, Ekaterinburg, Russia

Synchronizing automata

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

Synchronizing automata

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

The DFA $\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at.

Synchronizing automata

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

The DFA $\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at.
$|Q\,.\,w| = 1$. Here $Q\,.\,v$ stands for $\{\delta(q, v) \mid q \in Q\}$.

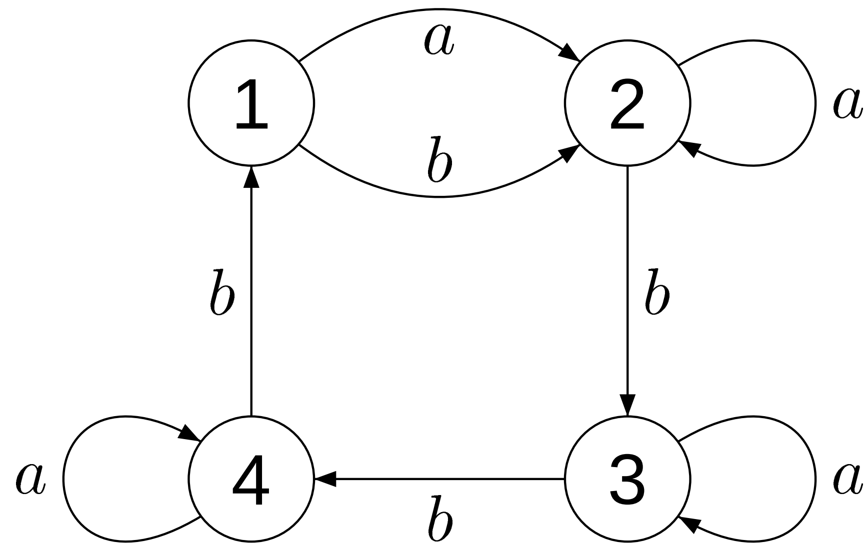
Synchronizing automata

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

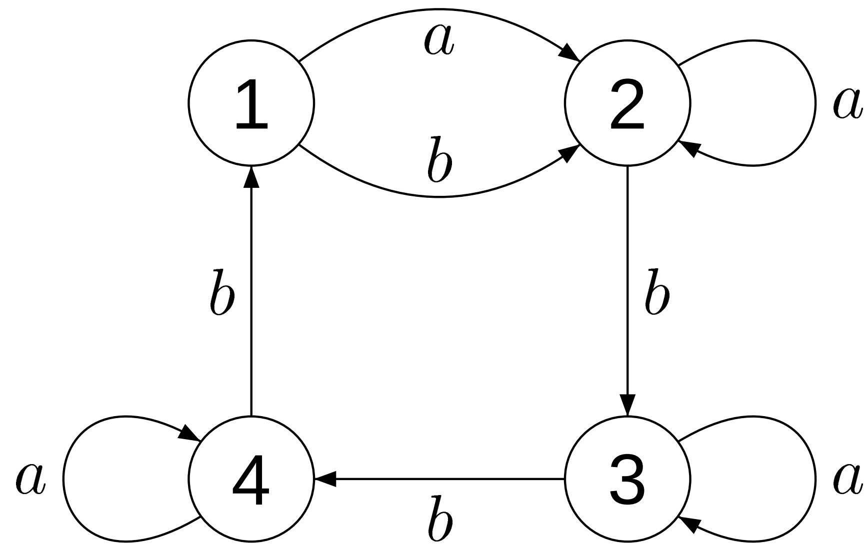
The DFA $\mathscr{A}$ is called *synchronizing* if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves the automaton in one particular state no matter which state in Q it started at.
$|Q\,.\,w| = 1$. Here $Q\,.\,v$ stands for $\{\delta(q, v) \mid q \in Q\}$.

Any word w with this property is said to be a *reset word* for the automaton.

Synchronizing automata



Synchronizing automata



A reset sequence of actions is $abbbabbba$. Applying it at any state brings the automaton to the state 2.

Synchronizing Automata

The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konecnými automatami, Mat.-Fyz. Cas. Slovensk. Akad. Vied. 14 (1964) 208–216) though implicitly it had been studied since 1956.

Synchronizing Automata

- The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konecnými automatami, Mat.-Fyz. Cas. Slovensk. Akad. Vied. 14 (1964) 208–216) though implicitly it had been studied since 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.

Synchronizing Automata

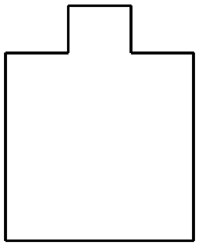
- The notion was formalized in 1964 in a paper by Jan Černý (Poznámka k homogénnym eksperimentom s konecnými automatami, Mat.-Fyz. Cas. Slovensk. Akad. Vied. 14 (1964) 208–216) though implicitly it had been studied since 1956.
- The idea of synchronization is pretty natural and of obvious importance: 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).

Engineering Applications

In the 80s, the notion was reinvented by engineers working in *robotics* or, more precisely, *robotic manipulation* which deals with part handling problems in industrial automation such as part feeding, fixturing, loading, assembly and packing (and which is therefore of utmost and direct practical importance).

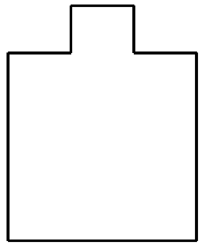
Engineering Applications

In the 80s, the notion was reinvented by engineers working in *robotics* or, more precisely, *robotic manipulation* which deals with part handling problems in industrial automation such as part feeding, fixturing, loading, assembly and packing (and which is therefore of utmost and direct practical importance).
Suppose that one of the parts of a certain device has the following shape:



Engineering Applications

In the 80s, the notion was reinvented by engineers working in *robotics* or, more precisely, *robotic manipulation* which deals with part handling problems in industrial automation such as part feeding, fixturing, loading, assembly and packing (and which is therefore of utmost and direct practical importance).
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.

Engineering Applications

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

Engineering Applications

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

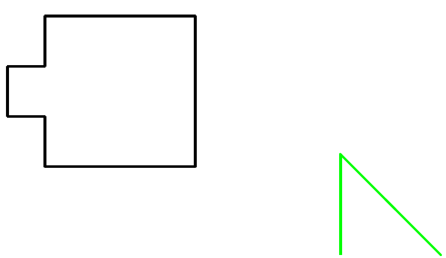
Suppose that prior the assembly the part should take the “bump-left” orientation (the second one on the picture). Thus, one has to construct an orienter which action will put the part in the prescribed position independently of its initial orientation.

Engineering Applications

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

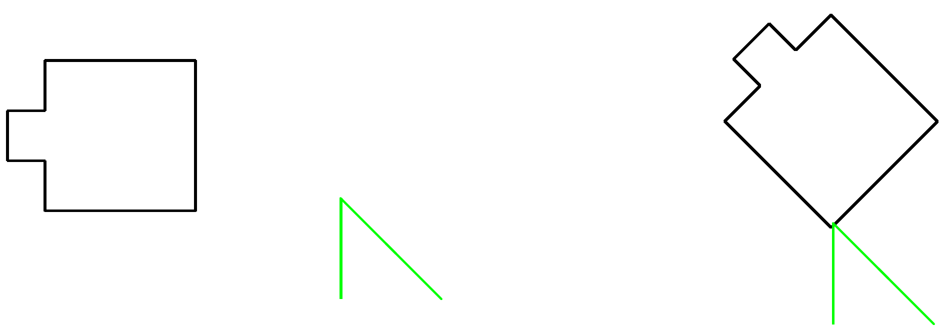
Engineering Applications

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the details 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.



Engineering Applications

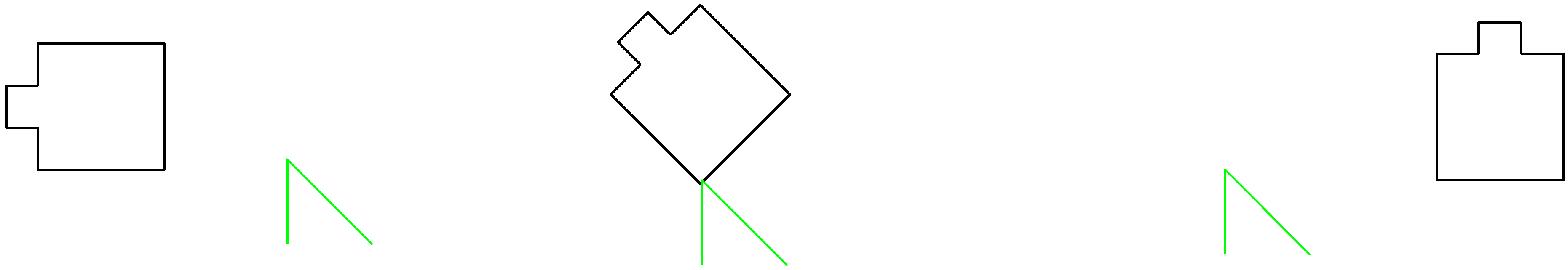
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the details encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by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



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

Engineering Applications

We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the details 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.



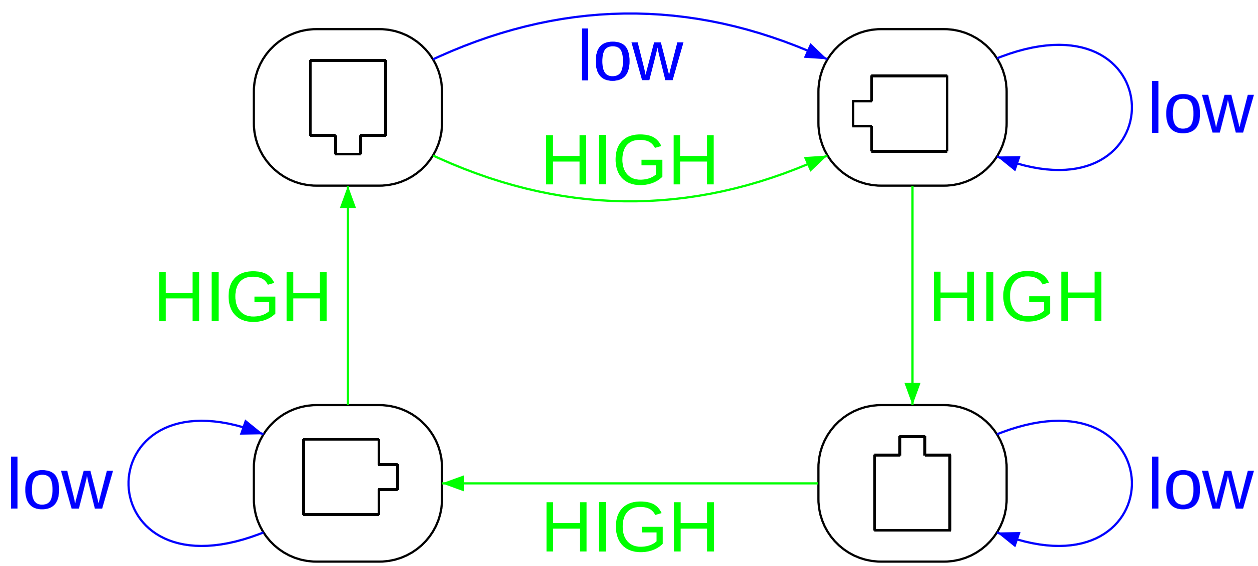
Being curried by the belt, the part then is forced to turn 90° clockwise.

Engineering Applications

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.

Engineering Applications

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:

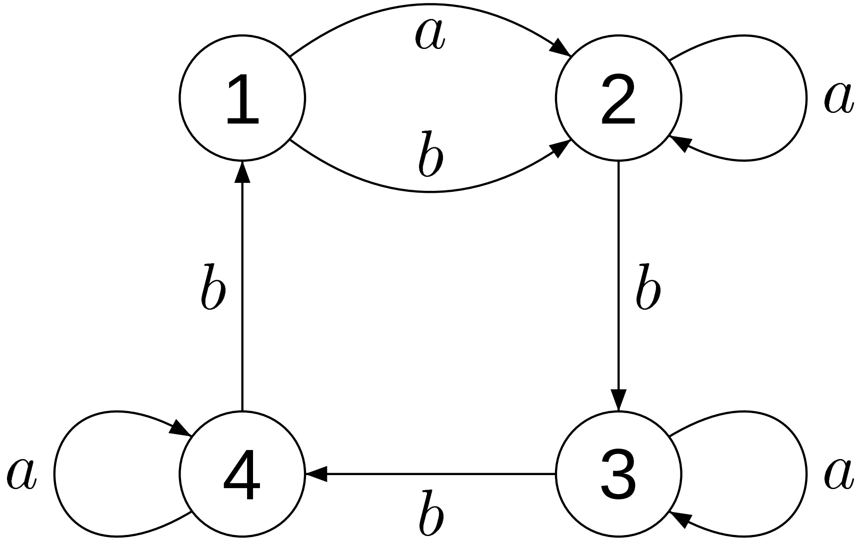


Engineering Applications

We met this picture a few slides ago:

Engineering Applications

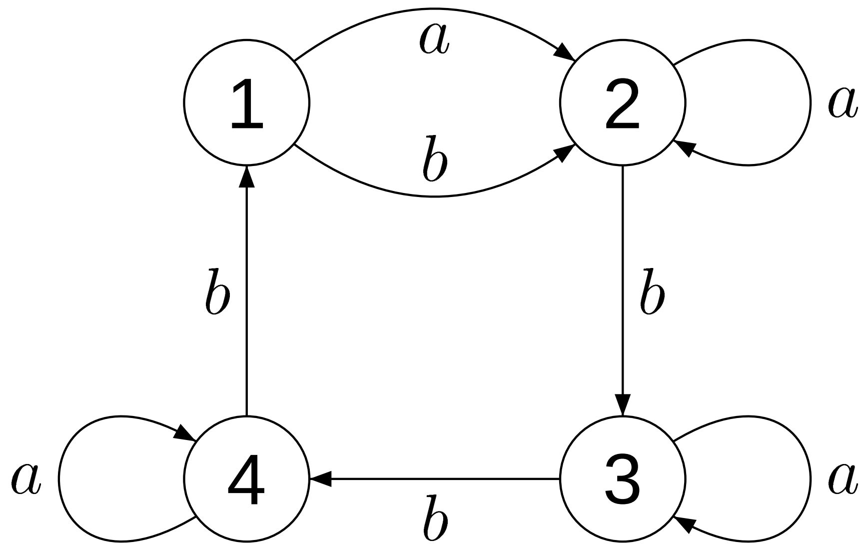
We met this picture a few slides ago:



– this was our example of a synchronizing automaton, and we saw that $abbbabbba$ is a reset sequence of actions.

Engineering Applications

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 sensorless orienter.

Further Applications

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).

Further Applications

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.

Further Applications

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.

Further Applications

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.

The Černý conjecture

Clearly, from the viewpoint of the above applications (as well as from the mathematical point of view) the following question is of importance:

The Černý conjecture

Clearly, from the viewpoint of the above applications (as well as from the mathematical point of view) the following question is of importance:
Suppose a synchronizing automaton has n *states. What is the length of its shortest reset sequence?*

The Černý conjecture

Clearly, from the viewpoint of the above applications (as well as from the mathematical point of view) the following question is of importance:
Suppose a synchronizing automaton has n *states. What is the length of its shortest reset sequence?*
In the example above the automaton has 4 states and there is a reset sequence of length 9. In fact, this was the shortest possible reset sequence.

The Černý conjecture

Clearly, from the viewpoint of the above applications (as well as from the mathematical point of view) the following question is of importance:
Suppose a synchronizing automaton has n states. What is the length of its shortest reset sequence?
In the example above the automaton has 4 states and there is a reset sequence of length 9. In fact, this was the shortest possible reset sequence.

In 1964, Černý conjectured that every synchronizing automaton with n states has a reset sequence of length $(n-1)^2$ — as in our example where $9 = (4-1)^2$.

The Černý conjecture

Clearly, from the viewpoint of the above applications (as well as from the mathematical point of view) the following question is of importance:
Suppose a synchronizing automaton has n states. What is the length of its shortest reset sequence?
In the example above the automaton has 4 states and there is a reset sequence of length 9. In fact, this was the shortest possible reset sequence.

In 1964, Černý conjectured that every synchronizing automaton with n states has a reset sequence of length $(n-1)^2$ — as in our example where $9 = (4-1)^2$.

The simply looking conjecture is still open in general!!

The Černý conjecture

The best upper bound known so far is $(n^3 - n)/6$ (J.-E. Pin, 1983).

The Černý conjecture

The best upper bound known so far is $(n^3 - n)/6$ (J.-E. Pin, 1983). It is also known that the problem is hard from the computational complexity point of view.

The Černý conjecture

The best upper bound known so far is $(n^3 - n)/6$ (J.-E. Pin, 1983). It is also known that the problem is hard from the computational complexity point of view.

Given a DFA $\mathscr{A}$ and a positive integer ℓ, the problem whether or not $\mathscr{A}$ has a reset word of length $\leq \ell$ is NP-complete (D. Eppstein, 1990; P. Goralčik and V. Koubek, 1995; A. Salomaa, 2003).

The Černý conjecture

The best upper bound known so far is $(n^3 - n)/6$ (J.-E. Pin, 1983). It is also known that the problem is hard from the computational complexity point of view.

Given a DFA $\mathscr{A}$ and a positive integer ℓ, the problem whether or not $\mathscr{A}$ has a reset word of length $\leq \ell$ is NP-complete (D. Eppstein, 1990; P. Goralčik and V. Koubek, 1995; A. Salomaa, 2003).

Given a DFA $\mathscr{A}$ and a positive integer ℓ, the problem whether or not the shortest reset word for $\mathscr{A}$ has length ℓ is co-NP-hard (W. Samotij, 2007).

Aperiodic Automata

Some progress has been achieved for various restricted classes of synchronizing automata.

Aperiodic Automata

Some progress has been achieved for various restricted classes of synchronizing automata. In particular, consider the class $\mathbf{Ap}$ of *aperiodic automata*.

Aperiodic Automata

Some progress has been achieved for various restricted classes of synchronizing automata. In particular, consider the class $\mathbf{Ap}$ of *aperiodic automata*.

Recall that the *transition monoid* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ consists of all transformations $\delta(\textvisiblespace, w) : Q \to Q$ induced by words $w \in \Sigma^*$.

Aperiodic Automata

Some progress has been achieved for various restricted classes of synchronizing automata. In particular, consider the class $\mathbf{Ap}$ of *aperiodic automata*.

Recall that the *transition monoid* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ consists of all transformations $\delta(\sqcup, w) : Q \to Q$ induced by words $w \in \Sigma^*$. A monoid is said to be *aperiodic* if all its subgroups are singletons.

Aperiodic Automata

Some progress has been achieved for various restricted classes of synchronizing automata. In particular, consider the class $\mathbf{Ap}$ of *aperiodic automata*.

Recall that the *transition monoid* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ consists of all transformations $\delta(\sqcup, w) : Q \to Q$ induced by words $w \in \Sigma^*$. A monoid is said to be *aperiodic* if all its subgroups are singletons. A DFA is called *aperiodic* (or *counter-free*) if its transition monoid is aperiodic.

Aperiodic Automata

- Some progress has been achieved for various restricted classes of synchronizing automata. In particular, consider the class $\mathbf{Ap}$ of *aperiodic automata*.
- Recall that the *transition monoid* of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ consists of all transformations $\delta(\sqcup, w) : Q \to Q$ induced by words $w \in \Sigma^*$. A monoid is said to be *aperiodic* if all its subgroups are singletons. A DFA is called *aperiodic* (or *counter-free*) if its transition monoid is aperiodic.
- Synchronization issues remain difficult when restricted to $\mathbf{Ap}$.

Aperiodic Automata

Good news: Recently A. Trakhtman has proved that every synchronizing aperiodic automaton with n states admits a reset word of length at most $\frac{n(n-1)}{2}$.

Aperiodic Automata

Good news: Recently A. Trakhtman has proved that every synchronizing aperiodic automaton with n states admits a reset word of length at most $\frac{n(n-1)}{2}$.

Bad news: No precise bound for $\mathrm{SAS}(n)$, the minimum length of reset words for synchronizing aperiodic automata with n states, has been found so far.

Aperiodic Automata

Good news: Recently A. Trakhtman has proved that every synchronizing aperiodic automaton with n states admits a reset word of length at most $\frac{n(n-1)}{2}$.

Bad news: No precise bound for $\mathrm{SAS}(n)$, the minimum length of reset words for synchronizing aperiodic automata with n states, has been found so far.

(Trakhtman) $\dfrac{n(n-1)}{2} \geq \mathrm{SAS}(n)$

Aperiodic Automata

Good news: Recently A. Trakhtman has proved that every synchronizing aperiodic automaton with n states admits a reset word of length at most $\frac{n(n-1)}{2}$.

Bad news: No precise bound for $\mathrm{SAS}(n)$, the minimum length of reset words for synchronizing aperiodic automata with n states, has been found so far.

(Trakhtman) $\dfrac{n(n-1)}{2} \geq \mathrm{SAS}(n) \geq n + \left\lfloor \dfrac{n}{2} \right\rfloor - 2$ (Ananichev)

Aperiodic Automata

Good news: Recently A. Trakhtman has proved that every synchronizing aperiodic automaton with n states admits a reset word of length at most $\frac{n(n-1)}{2}$.

Bad news: No precise bound for $\mathrm{SAS}(n)$, the minimum length of reset words for synchronizing aperiodic automata with n states, has been found so far.

(Trakhtman) $\dfrac{n(n-1)}{2} \geq \mathrm{SAS}(n) \geq n + \left\lfloor \dfrac{n}{2} \right\rfloor - 2$ (Ananichev)

The gap between the upper and the lower bounds is rather drastic.

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata.

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata. The question of whether or not a given DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is aperiodic is PSPACE-complete (Cho and Huynh, 1991).

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata. The question of whether or not a given DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is aperiodic is PSPACE-complete (Cho and Huynh, 1991). Practically, there is no way to check the aperiodicity of $\mathscr{A}$ avoiding the calculation of its transition monoid, and the cardinality of the monoid can reach $|\boldsymbol{Q}|^{|\boldsymbol{Q}|}$.

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata. The question of whether or not a given DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is aperiodic is PSPACE-complete (Cho and Huynh, 1991). Practically, there is no way to check the aperiodicity of $\mathscr{A}$ avoiding the calculation of its transition monoid, and the cardinality of the monoid can reach $|\boldsymbol{Q}|^{|\boldsymbol{Q}|}$. Hence, no hope that experiments can help.

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata. The question of whether or not a given DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is aperiodic is PSPACE-complete (Cho and Huynh, 1991). Practically, there is no way to check the aperiodicity of $\mathscr{A}$ avoiding the calculation of its transition monoid, and the cardinality of the monoid can reach $|\boldsymbol{Q}|^{|\boldsymbol{Q}|}$. Hence, no hope that experiments can help. On the other hand, all attempts to reduce the upper bound have failed so far.

Aperiodic Automata

Producing lower bounds for $\mathrm{SAS}(n)$ is difficult because it is quite difficult to produce aperiodic automata. The question of whether or not a given DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is aperiodic is PSPACE-complete (Cho and Huynh, 1991). Practically, there is no way to check the aperiodicity of $\mathscr{A}$ avoiding the calculation of its transition monoid, and the cardinality of the monoid can reach $|\boldsymbol{Q}|^{|\boldsymbol{Q}|}$. Hence, no hope that experiments can help. On the other hand, all attempts to reduce the upper bound have failed so far.

An idea: consider certain properties that guarantee aperiodicity and are easier to check.

Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

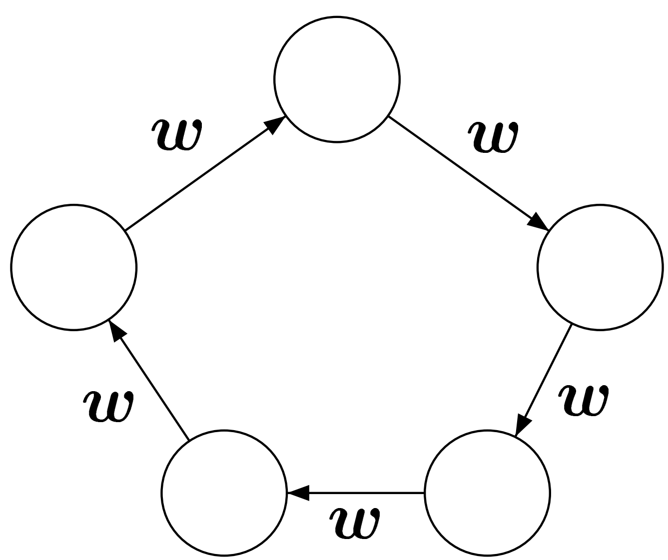
Monotonic automata are aperiodic (known and easy).

Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).

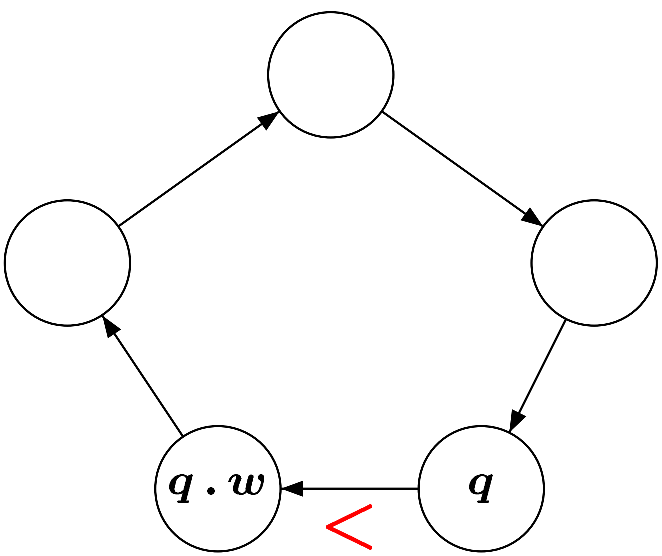


Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).

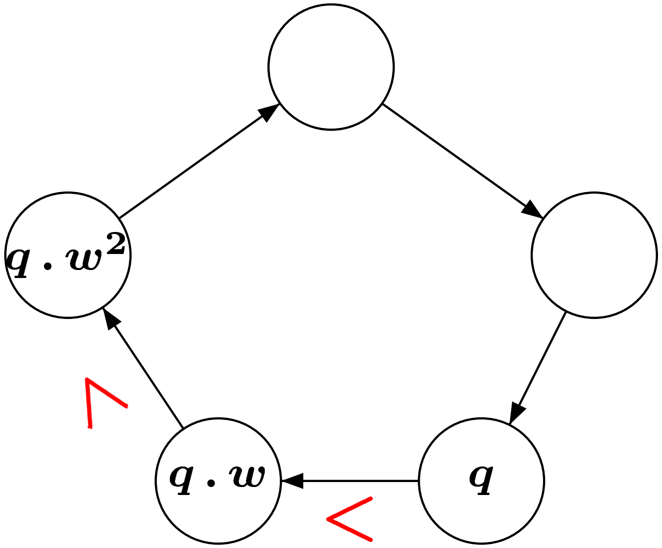


Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).

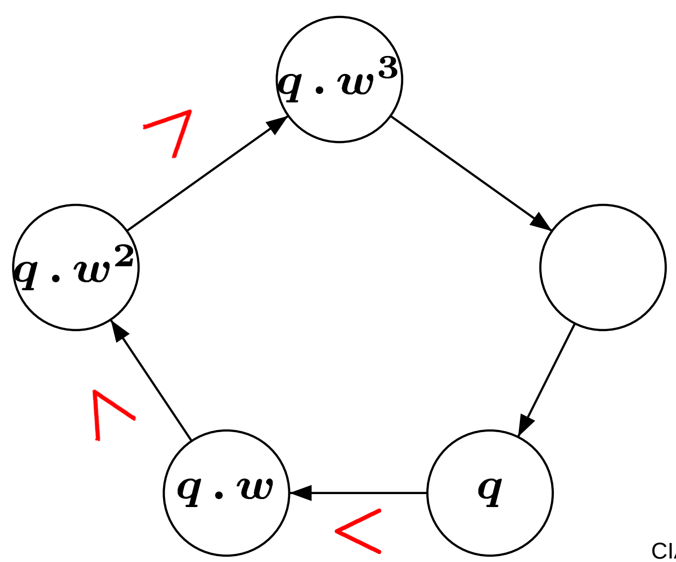


Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).

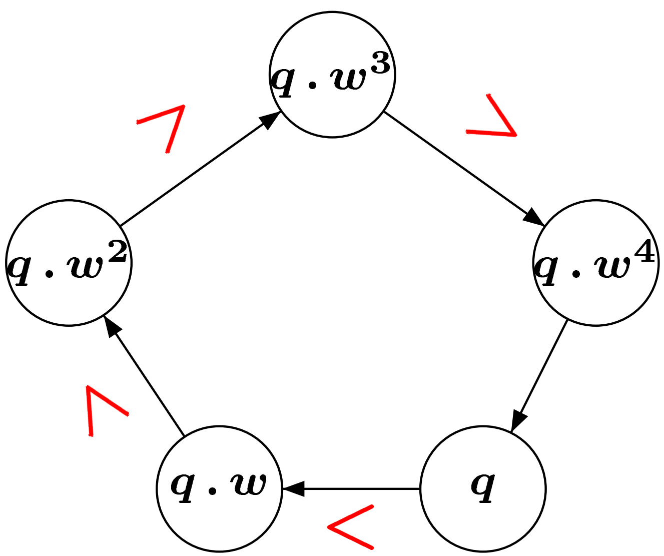


Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).

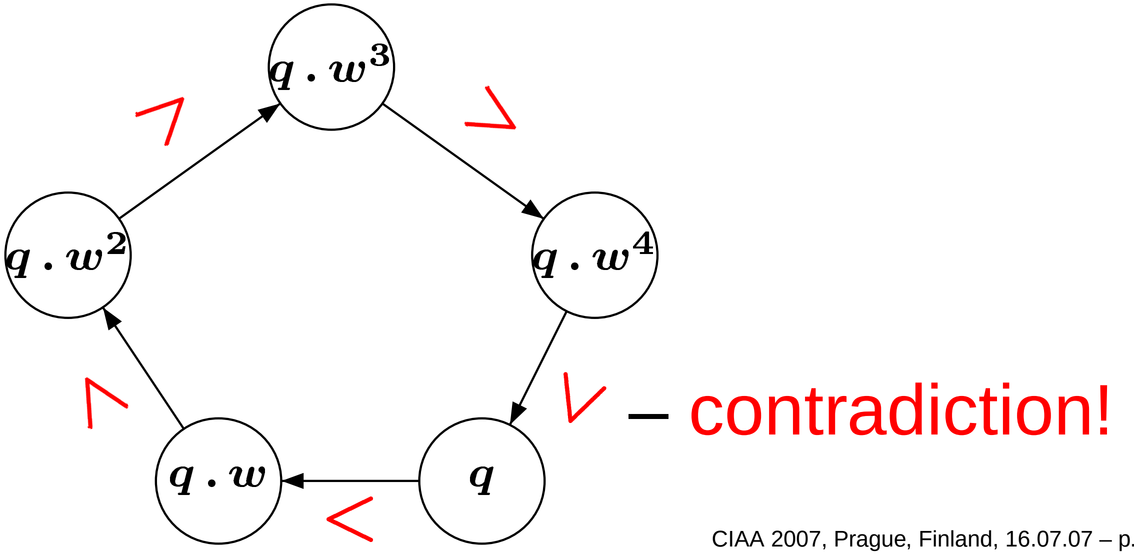


Monotonicity

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is *monotonic* if Q admits a linear order $\leq$ such that, for $a \in \Sigma$, the transformation $\delta(\sqcup, a)$ of Q preserves $\leq$:

$$p \leq q \Rightarrow \delta(p, a) \leq \delta(q, a).$$

Monotonic automata are aperiodic (known and easy).



Monotonicity

For monotonic automata the synchronization problem is easy: Ananichev and $\sim$ (2004) observed that every monotonic synchronizing automaton with n states has a reset word of length $\leq n-1$ and the bound is tight.

Monotonicity

For monotonic automata the synchronization problem is easy: Ananichev and $\sim$ (2004) observed that every monotonic synchronizing automaton with n states has a reset word of length $\leq n-1$ and the bound is tight.

A much more difficult case of so-called 0-monotonic automata was analyzed by Ananichev in 2005.

Monotonicity

For monotonic automata the synchronization problem is easy: Ananichev and $\sim$ (2004) observed that every monotonic synchronizing automaton with n states has a reset word of length $\leq n-1$ and the bound is tight.

A much more difficult case of so-called 0-monotonic automata was analyzed by Ananichev in 2005. A DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is 0-*monotonic* if it has a unique *sink* $\boldsymbol{s} \in \boldsymbol{Q}$ and $\boldsymbol{Q} \setminus \{\boldsymbol{s}\}$ admits a linear order $\leq$ preserved by the restrictions of the transformations $\boldsymbol{\delta}(\sqcup, \boldsymbol{a})$ to $\boldsymbol{Q} \setminus \{\boldsymbol{s}\}$.

Monotonicity

For monotonic automata the synchronization problem is easy: Ananichev and $\sim$ (2004) observed that every monotonic synchronizing automaton with n states has a reset word of length $\leq n-1$ and the bound is tight.

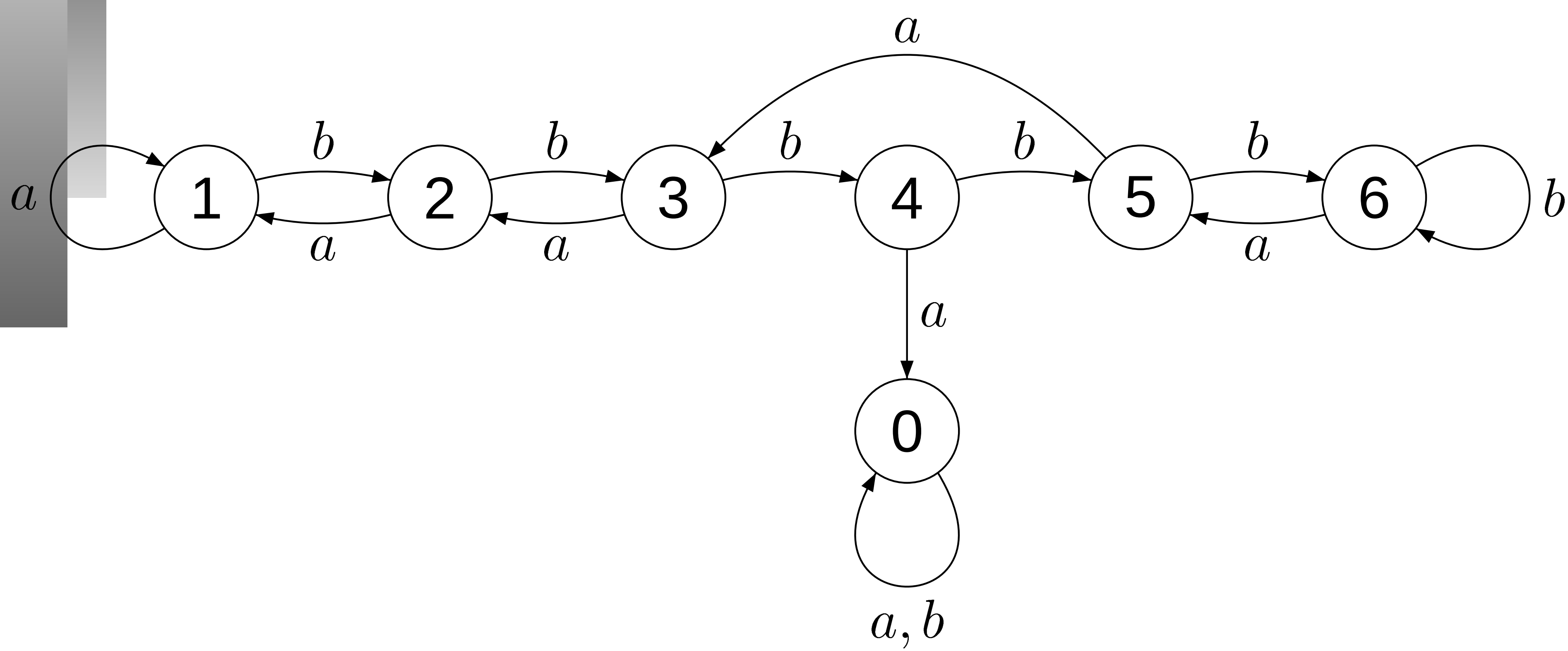
A much more difficult case of so-called 0-monotonic automata was analyzed by Ananichev in 2005. A DFA $\mathscr{A} = \langle \boldsymbol{Q}, \Sigma, \boldsymbol{\delta} \rangle$ is 0-*monotonic* if it has a unique *sink* $\boldsymbol{s} \in \boldsymbol{Q}$ and $\boldsymbol{Q} \setminus \{s\}$ admits a linear order $\leq$ preserved by the restrictions of the transformations $\boldsymbol{\delta}(\sqcup, \boldsymbol{a})$ to $\boldsymbol{Q} \setminus \{s\}$. Clearly, 0-monotonic automata are in a 1-1 correspondence with incomplete monotonic automata.

Monotonicity

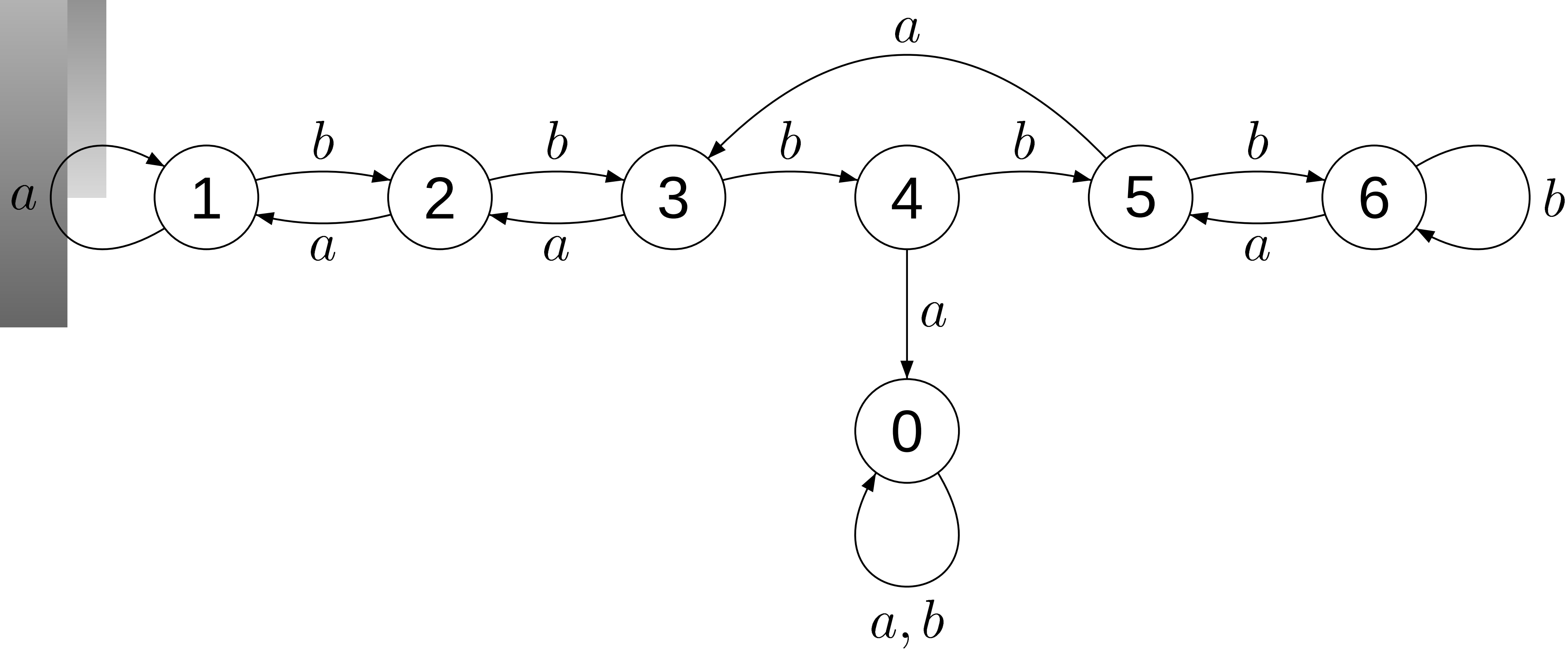
For monotonic automata the synchronization problem is easy: Ananichev and $\sim$ (2004) observed that every monotonic synchronizing automaton with n states has a reset word of length $\leq n-1$ and the bound is tight.

A much more difficult case of so-called 0-monotonic automata was analyzed by Ananichev in 2005. A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is 0-*monotonic* if it has a unique *sink* $s \in Q$ and $Q \setminus \{s\}$ admits a linear order $\leq$ preserved by the restrictions of the transformations $\delta(\sqcup, a)$ to $Q \setminus \{s\}$. Clearly, 0-monotonic automata are in a 1-1 correspondence with incomplete monotonic automata. Again, it is known and easy to check that 0-monotonic automata are aperiodic.

Monotonicity

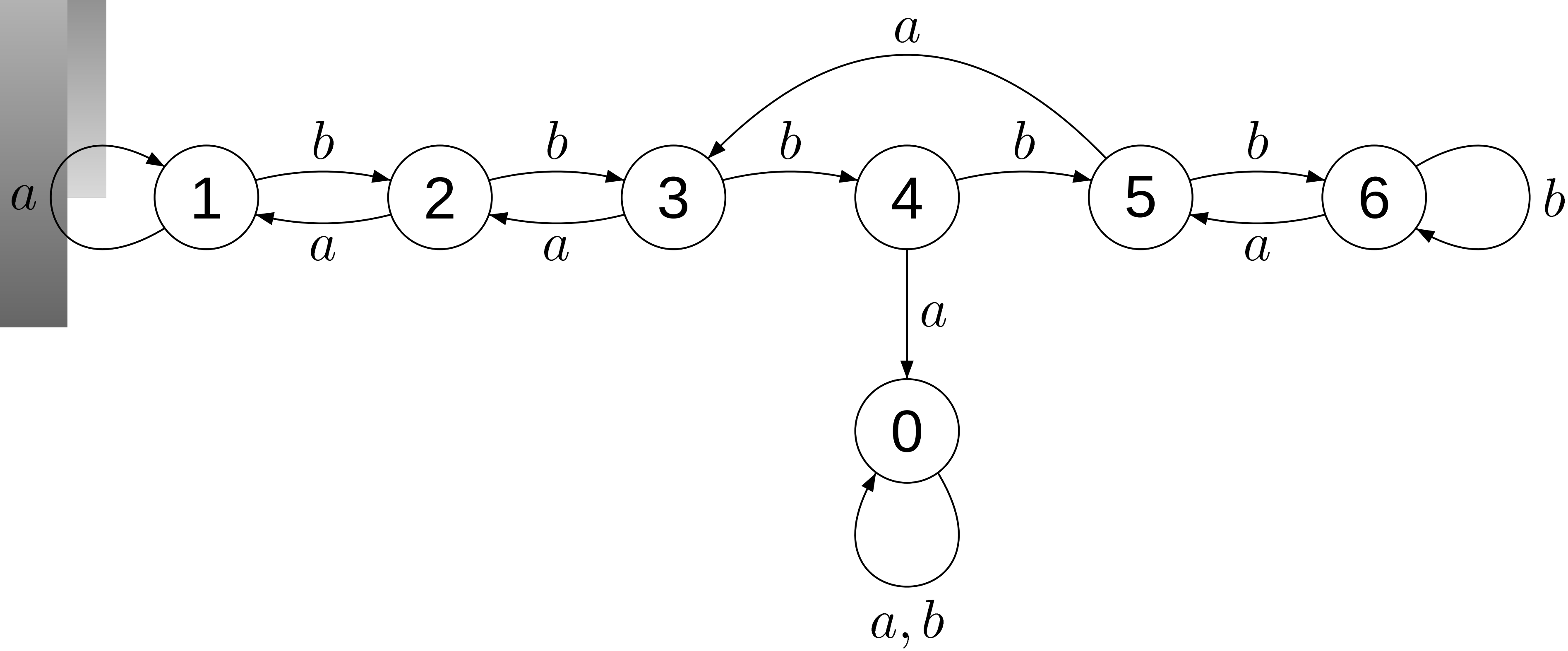


Monotonicity



This 0-monotonic automaton is the first in Ananichev's series that yields the lower bound $\mathrm{SAS}(n) \geq n + \left\lfloor \frac{n}{2} \right\rfloor - 2$.

Monotonicity



This 0-monotonic automaton is the first in Ananichev's series that yields the lower bound $\mathrm{SAS}(n) \geq n + \left\lfloor \frac{n}{2} \right\rfloor - 2$.

It has 7 states and its shortest reset word is a^4b^3a of length $7 + \left\lfloor \frac{7}{2} \right\rfloor - 2 = 8$.

Generalized Monotonicity

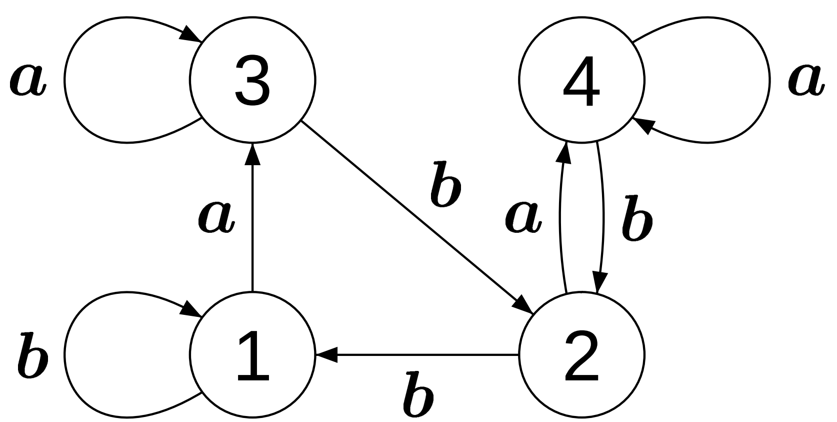
A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is a *stable* if $(\boldsymbol{p}, \boldsymbol{q}) \in \boldsymbol{\rho}$ implies $\big(\boldsymbol{\delta}(\boldsymbol{p}, \boldsymbol{a}), \boldsymbol{\delta}(\boldsymbol{q}, \boldsymbol{a})\big) \in \boldsymbol{\rho}$ for all $\boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{Q}$ and $\boldsymbol{a} \in \boldsymbol{\Sigma}$.

Generalized Monotonicity

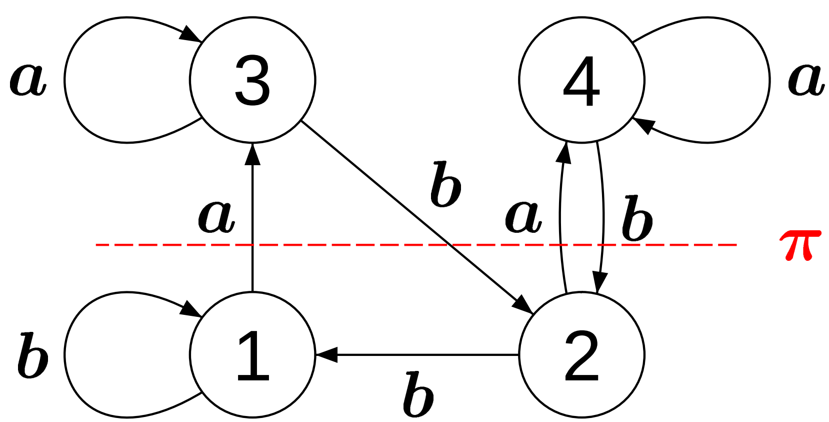
A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle \boldsymbol{Q}, \boldsymbol{\Sigma}, \boldsymbol{\delta} \rangle$ is a *stable* if $(\boldsymbol{p}, \boldsymbol{q}) \in \boldsymbol{\rho}$ implies $\big(\boldsymbol{\delta}(\boldsymbol{p}, \boldsymbol{a}), \boldsymbol{\delta}(\boldsymbol{q}, \boldsymbol{a})\big) \in \boldsymbol{\rho}$ for all $\boldsymbol{p}, \boldsymbol{q} \in \boldsymbol{Q}$ and $\boldsymbol{a} \in \boldsymbol{\Sigma}$.
A *congruence* is a stable equivalence. For $\boldsymbol{\pi}$ being a congruence, $[\boldsymbol{q}]_{\boldsymbol{\pi}}$ is the $\boldsymbol{\pi}$-class containing the state $\boldsymbol{q}$.

Generalized Monotonicity

A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is a *stable* if $(p, q) \in \rho$ implies $\big(\delta(p, a), \delta(q, a)\big) \in \rho$ for all $p, q \in Q$ and $a \in \Sigma$.
A *congruence* is a stable equivalence. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.

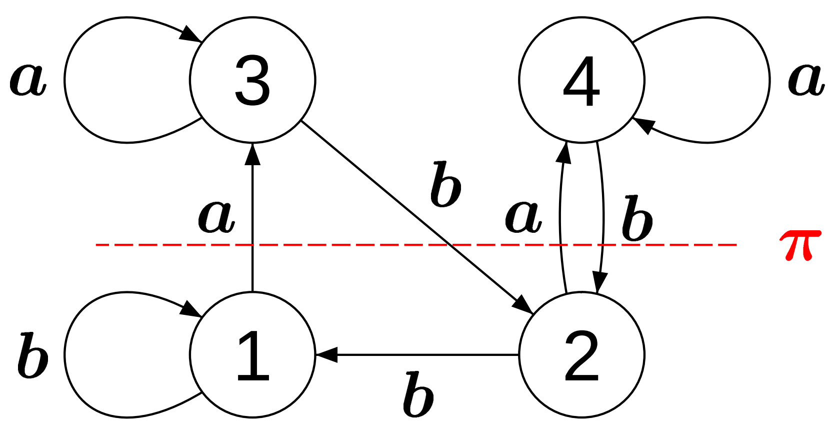


A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is a *stable* if $(p, q) \in \rho$ implies $(\delta(p, a), \delta(q, a)) \in \rho$ for all $p, q \in Q$ and $a \in \Sigma$.
A *congruence* is a stable equivalence. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.



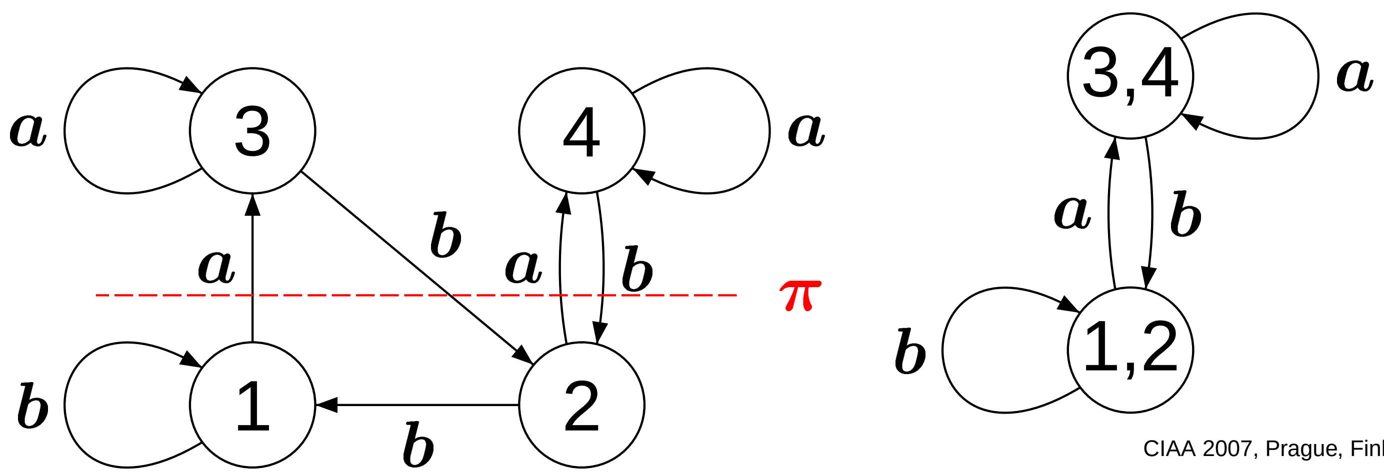
Generalized Monotonicity

A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is a *stable* if $(p, q) \in \rho$ implies $(\delta(p, a), \delta(q, a)) \in \rho$ for all $p, q \in Q$ and $a \in \Sigma$.
A *congruence* is a stable equivalence. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.
The *quotient* $\mathscr{A}/\pi$ is the DFA $\langle Q/\pi, \Sigma, \delta_\pi \rangle$ where $Q/\pi = \{[q]_\pi \mid q \in Q\}$ and the function δ_π is defined by the rule $\delta_\pi([q]_\pi, a) = [\delta(q, a)]_\pi$.



Generalized Monotonicity

A binary relation ρ on the state set Q of a DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is a *stable* if $(p, q) \in \rho$ implies $(\delta(p, a), \delta(q, a)) \in \rho$ for all $p, q \in Q$ and $a \in \Sigma$.
A *congruence* is a stable equivalence. For π being a congruence, $[q]_\pi$ is the π-class containing the state q.
The *quotient* $\mathscr{A}/\pi$ is the DFA $\langle Q/\pi, \Sigma, \delta_\pi \rangle$ where $Q/\pi = \{[q]_\pi \mid q \in Q\}$ and the function δ_π is defined by the rule $\delta_\pi([q]_\pi, a) = [\delta(q, a)]_\pi$.



Generalized Monotonicity

We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\boldsymbol{\rho_0} \subset \boldsymbol{\rho_1} \subset \cdots \subset \boldsymbol{\rho_\ell}$, satisfying the following conditions:

Generalized Monotonicity

We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\boldsymbol{\rho_0} \subset \boldsymbol{\rho_1} \subset \cdots \subset \boldsymbol{\rho_\ell}$, satisfying the following conditions:

- $\boldsymbol{\rho_0}$ is the equality;

Generalized Monotonicity

We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\boldsymbol{\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell}$, satisfying the following conditions:

- $\boldsymbol{\rho_0}$ is the equality;
- for each $\boldsymbol{i = 1, \ldots, \ell}$, the congruence $\boldsymbol{\pi_{i-1}}$ generated by $\boldsymbol{\rho_{i-1}}$ is contained in $\boldsymbol{\rho_i}$ and the relation $\boldsymbol{\rho_i / \pi_{i-1}}$ is a linear order on each $\boldsymbol{\pi_i / \pi_{i-1}}$-class;

Generalized Monotonicity

We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\boldsymbol{\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell}$, satisfying the following conditions:
- $\boldsymbol{\rho_0}$ is the equality;
- for each $\boldsymbol{i = 1, \ldots, \ell}$, the congruence $\boldsymbol{\pi_{i-1}}$ generated by $\boldsymbol{\rho_{i-1}}$ is contained in $\boldsymbol{\rho_i}$ and the relation $\boldsymbol{\rho_i/\pi_{i-1}}$ is a linear order on each $\boldsymbol{\pi_i/\pi_{i-1}}$-class;
- $\boldsymbol{\pi_\ell}$ is the universal relation.

Generalized Monotonicity

We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$, satisfying the following conditions:

- ρ_0 is the equality;
- for each $i = 1, \ldots, \ell$, the congruence π_{i-1} generated by ρ_{i-1} is contained in ρ_i and the relation ρ_i/π_{i-1} is a linear order on each π_i/π_{i-1}-class;
- π_ℓ is the universal relation.

Monotonic automata are precisely generalized monotonic automata of level 1.

Generalized Monotonicity

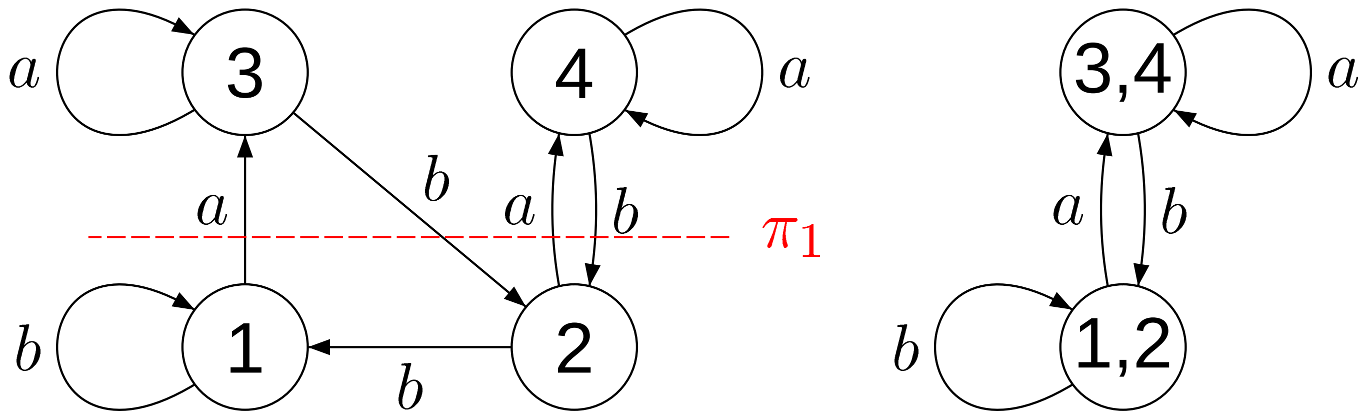
We call a DFA $\mathscr{A}$ *generalized monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$, satisfying the following conditions:

- ρ_0 is the equality;
- for each $i = 1, \ldots, \ell$, the congruence π_{i-1} generated by ρ_{i-1} is contained in ρ_i and the relation ρ_i/π_{i-1} is a linear order on each π_i/π_{i-1}-class;
- π_ℓ is the universal relation.

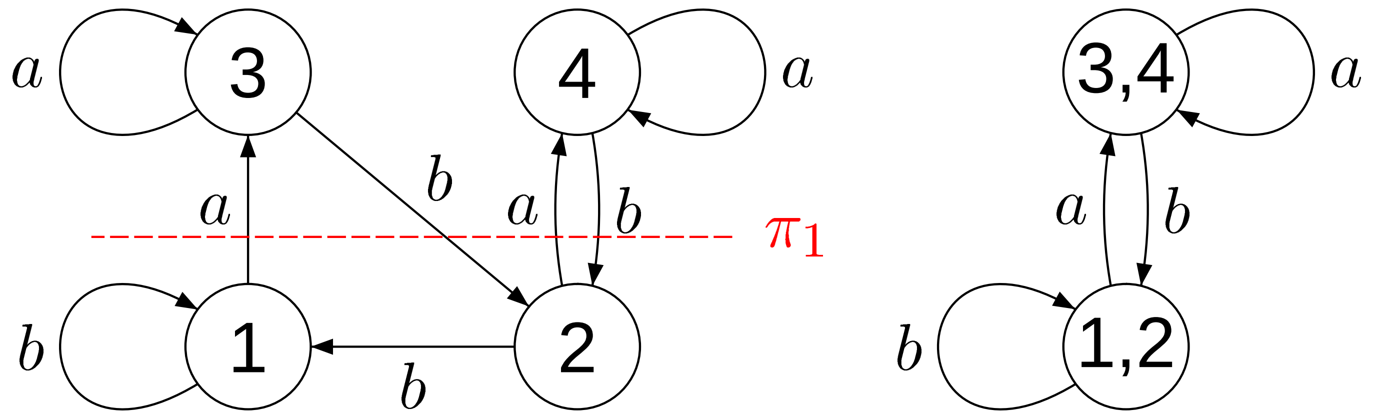
Monotonic automata are precisely generalized monotonic automata of level 1.

The automaton in the previous example is a generalized monotonic automaton of level 2.

Generalized Monotonicity

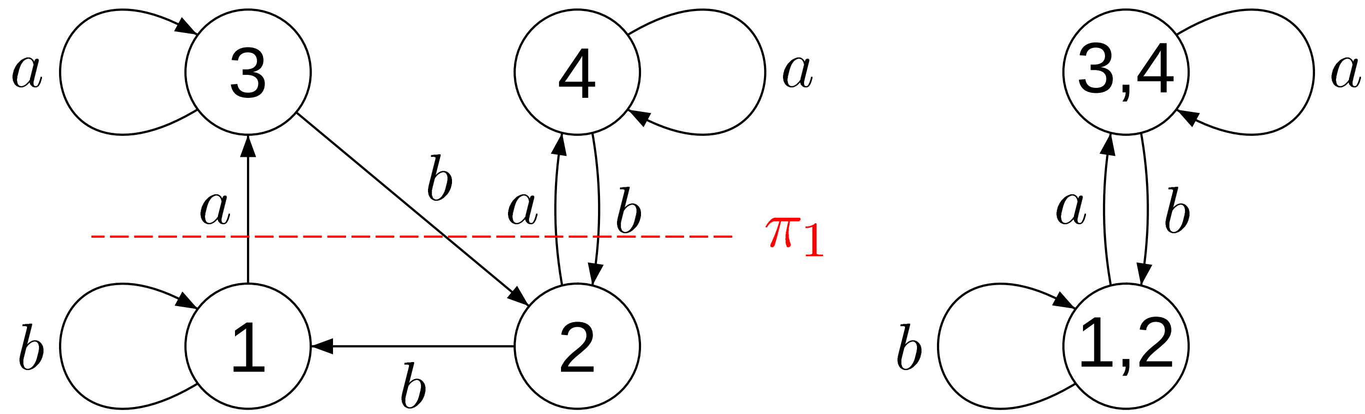


Generalized Monotonicity



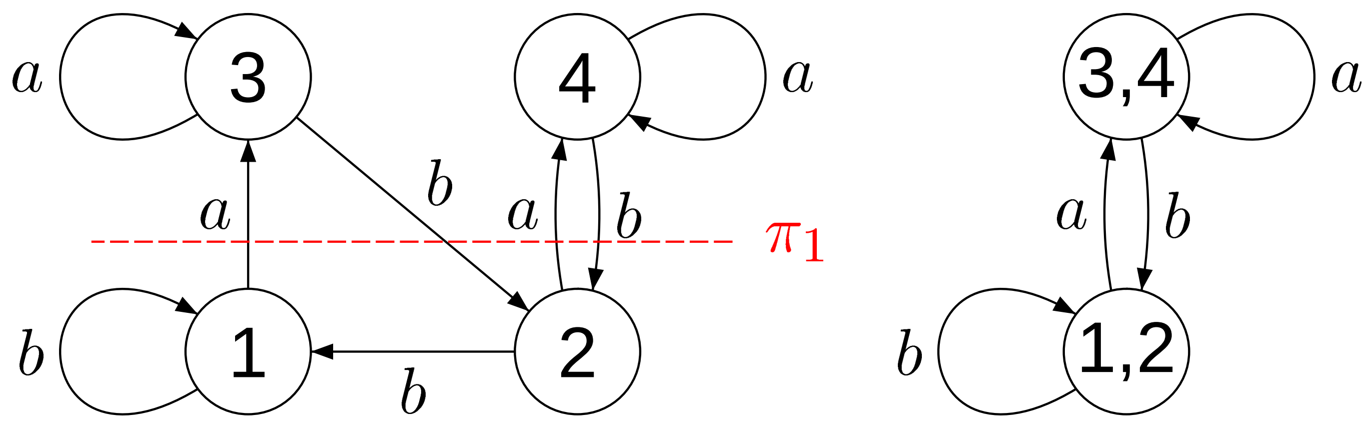
Endowing Q with the order $\leq_1$ such that $1 <_1 2$ and $3 <_1 4$, we get a linear order on each π_1-class.

Generalized Monotonicity



Endowing Q with the order $\leq_1$ such that $1 <_1 2$ and $3 <_1 4$, we get a linear order on each π_1-class. If we order Q/π_1 by letting $\{1, 2\} <_2 \{3, 4\}$, the quotient automaton becomes monotonic.

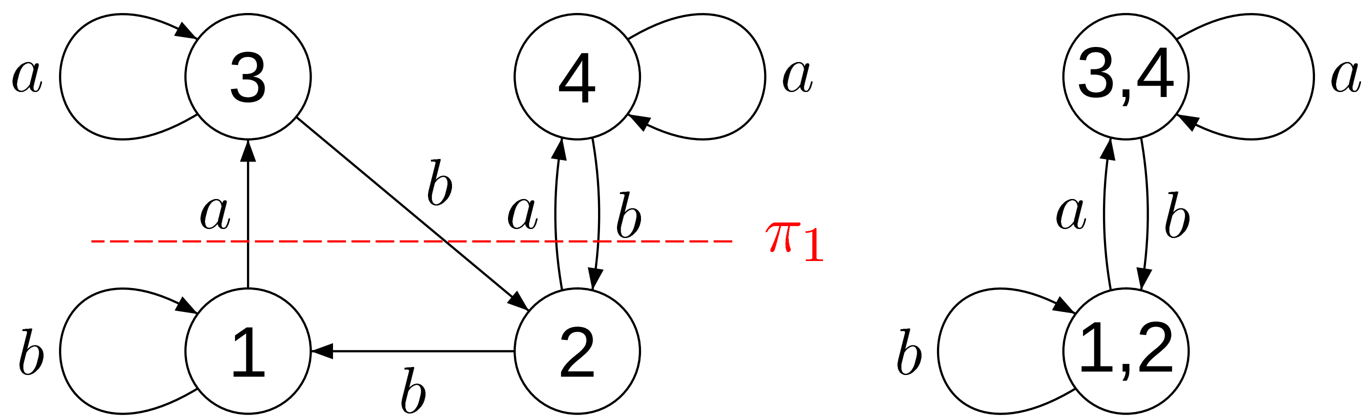
Generalized Monotonicity



Endowing Q with the order $\leq_1$ such that $1 <_1 2$ and $3 <_1 4$, we get a linear order on each π_1-class. If we order Q/π_1 by letting $\{1,2\} <_2 \{3,4\}$, the quotient automaton becomes monotonic.

It can be shown that the automaton is not monotonic.

Generalized Monotonicity



Endowing Q with the order $\leq_1$ such that $1 <_1 2$ and $3 <_1 4$, we get a linear order on each π_1-class. If we order Q/π_1 by letting $\{1,2\} <_2 \{3,4\}$, the quotient automaton becomes monotonic.

It can be shown that the automaton is not monotonic. Moreover, it cannot be emulated by any monotonic automaton.

Generalized Monotonicity

1. The hierarchy of generalized monotonic automata is strict: there are automata of each level $\ell = 1, 2, \ldots$.

Generalized Monotonicity

1. The hierarchy of generalized monotonic automata is strict: there are automata of each level $\ell = 1, 2, \ldots$.
2. Every generalized monotonic automaton is aperiodic.

Generalized Monotonicity

1. The hierarchy of generalized monotonic automata is strict: there are automata of each level $\ell = 1, 2, \ldots$.
2. Every generalized monotonic automaton is aperiodic.
3. Every star-free language can be recognized by a generalized monotonic automaton.

Generalized Monotonicity

1. The hierarchy of generalized monotonic automata is strict: there are automata of each level $\ell = 1, 2, \ldots$.
2. Every generalized monotonic automaton is aperiodic.
3. Every star-free language can be recognized by a generalized monotonic automaton.

However, generalized monotonic automata are not representative for the class $\mathbf{Ap}$ from the synchronization point of view: Ananichev and $\sim$ (2005) proved that every generalized monotonic synchronizing automaton with n states has a reset word of length $\leq n - 1$.

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.
We call a DFA $\mathscr{A}$ *weakly monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$ satisfying the following conditions:

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.
We call a DFA $\mathscr{A}$ *weakly monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$ satisfying the following conditions:
- ρ_0 is the equality relation;

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.
We call a DFA $\mathscr{A}$ *weakly monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$ satisfying the following conditions:
- ρ_0 is the equality relation;
- for each $i = 1, \ldots, \ell$, the congruence π_{i-1} generated by ρ_{i-1} is contained in ρ_i and the relation ρ_i/π_{i-1} is a (partial) order on Q/π_{i-1};

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.
We call a DFA $\mathscr{A}$ *weakly monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$ satisfying the following conditions:
• ρ_0 is the equality relation;
• for each $i = 1, \ldots, \ell$, the congruence π_{i-1} generated by ρ_{i-1} is contained in ρ_i and the relation ρ_i/π_{i-1} is a (partial) order on Q/π_{i-1};
• π_ℓ is the universal relation.

Yet Another Generalization

Surprisingly, a slight relaxation of the definition of a generalized monotonic automaton gives a much larger class of automata that strictly includes $\mathbf{Ap}$.

We call a DFA $\mathscr{A}$ *weakly monotonic of level* ℓ if it has a strictly increasing chain of stable binary relations $\rho_0 \subset \rho_1 \subset \cdots \subset \rho_\ell$ satisfying the following conditions:

- ρ_0 is the equality relation;
- for each $i = 1, \ldots, \ell$, the congruence π_{i-1} generated by ρ_{i-1} is contained in ρ_i and the relation ρ_i/π_{i-1} is a (partial) order on Q/π_{i-1};
- π_ℓ is the universal relation.

Thus, we just dropped the restriction that the order ρ_i/π_{i-1} is linear on each π_i/π_{i-1}-class.

Yet Another Generalization

Examples:

Yet Another Generalization

Examples:

- every aperiodic automaton is weakly monotonic;

Yet Another Generalization

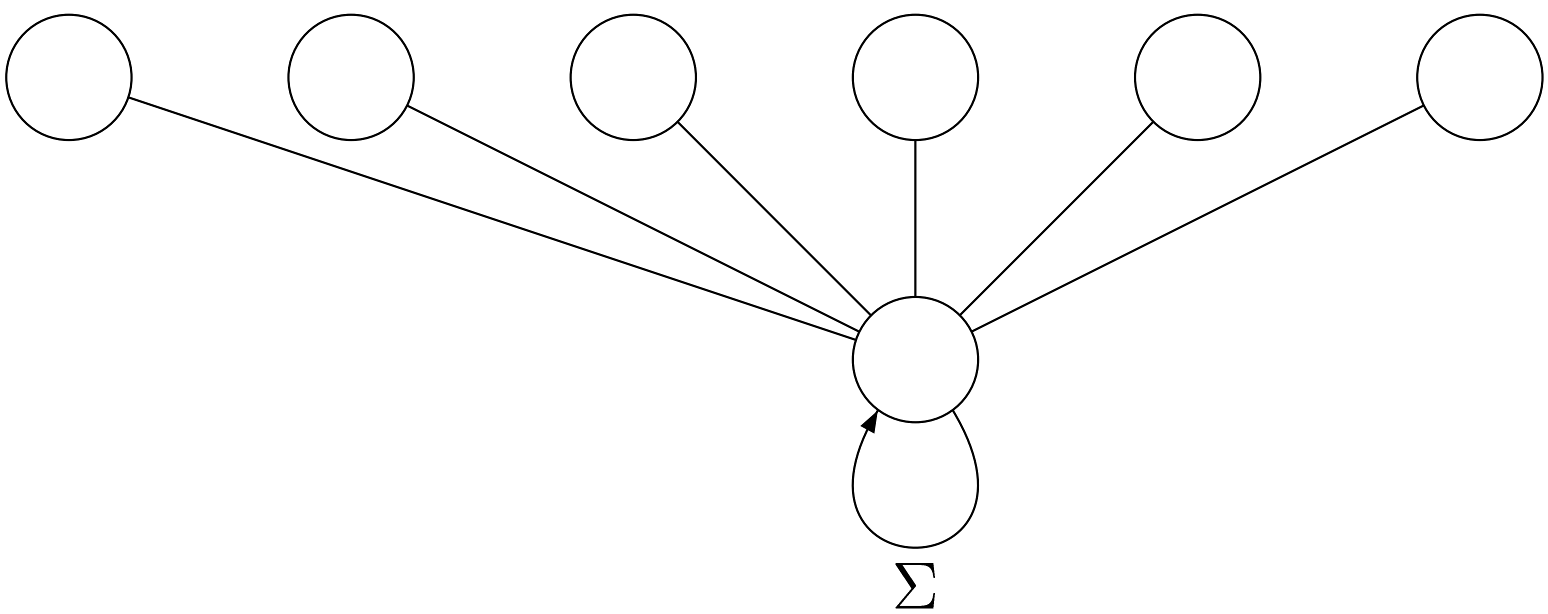
Examples:

- every aperiodic automaton is weakly monotonic;
- every automaton with a unique sink state is weakly monotonic (of level 1).

Yet Another Generalization

Examples:

- every aperiodic automaton is weakly monotonic;
- every automaton with a unique sink state is weakly monotonic (of level 1).



Yet Another Generalization

Main results:

Yet Another Generalization

Main results:

- Every weakly monotonic automaton with a strongly connected underlying digraph is synchronizing.

Yet Another Generalization

Main results:

- Every weakly monotonic automaton with a strongly connected underlying digraph is synchronizing. (A non-trivial generalization of the corresponding result for aperiodic automata.)

Yet Another Generalization

Main results:

- Every weakly monotonic automaton with a strongly connected underlying digraph is synchronizing. (A non-trivial generalization of the corresponding result for aperiodic automata.)
- Every weakly monotonic automaton with a strongly connected underlying digraph and n states has a reset word of length $\leq \left\lfloor \frac{n(n+1)}{6} \right\rfloor$.

Yet Another Generalization

Main results:

- Every weakly monotonic automaton with a strongly connected underlying digraph is synchronizing. (A non-trivial generalization of the corresponding result for aperiodic automata.)
- Every weakly monotonic automaton with a strongly connected underlying digraph and n states has a reset word of length $\leq \left\lfloor \frac{n(n+1)}{6} \right\rfloor$. (This upper bound is new even for the aperiodic case – recall that Trakhtman's bound was 3 times higher, namely, $\frac{n(n-1)}{2}$.)