Adversarial Synchronization

Anton Lipin and Mikhail Volkov

February 25, 2026

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the finite set of states;
- Σ is the finite alphabet of inputs;
- $\delta \colon Q \times \Sigma \to Q$ is the transition function.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta \colon Q \times \Sigma \to Q$ is the transition function.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the totally defined transition function.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

The function δ specifies how inputs act on states: if $\mathscr{A}$ is in a state $q \in Q$ and receives an input $a \in \Sigma$, the next state of $\mathscr{A}$ is $\delta(q, a)$.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

The function δ specifies how inputs act on states: if $\mathscr{A}$ is in a state $q \in Q$ and receives an input $a \in \Sigma$, the next state of $\mathscr{A}$ is $\delta(q, a)$.

We do not need specifying initial and final states here.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

The function δ specifies how inputs act on states: if $\mathscr{A}$ is in a state $q \in Q$ and receives an input $a \in \Sigma$, the next state of $\mathscr{A}$ is $\delta(q, a)$.

We do not need specifying initial and final states here.

Σ^* stands for the set of all words over Σ, that is, finite sequences of inputs including the empty word. The function δ extends to a function $Q \times \Sigma^* \to Q$

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

The function δ specifies how inputs act on states: if $\mathscr{A}$ is in a state $q \in Q$ and receives an input $a \in \Sigma$, the next state of $\mathscr{A}$ is $\delta(q, a)$.

We do not need specifying initial and final states here.

Σ^* stands for the set of all words over Σ, that is, finite sequences of inputs including the empty word. The function δ extends to a function $Q \times \Sigma^* \to Q$: we set $\delta(q, a_1 a_2) = \delta((\delta(q, a_1), a_2)$, etc.

Definitions and terminology

We consider complete deterministic finite automata (DFAs):

$$\mathscr{A} = \langle Q, \Sigma, \delta \rangle.$$

- Q is the set of states;
- Σ is the alphabet of inputs;
- $\delta\colon Q \times \Sigma \to Q$ is the transition function.

The function δ specifies how inputs act on states: if $\mathscr{A}$ is in a state $q \in Q$ and receives an input $a \in \Sigma$, the next state of $\mathscr{A}$ is $\delta(q, a)$.

We do not need specifying initial and final states here.

Σ^* stands for the set of all words over Σ, that is, finite sequences of inputs including the empty word. The function δ extends to a function $Q \times \Sigma^* \to Q$: we set $\delta(q, a_1 a_2) = \delta((\delta(q, a_1), a_2)$, etc.

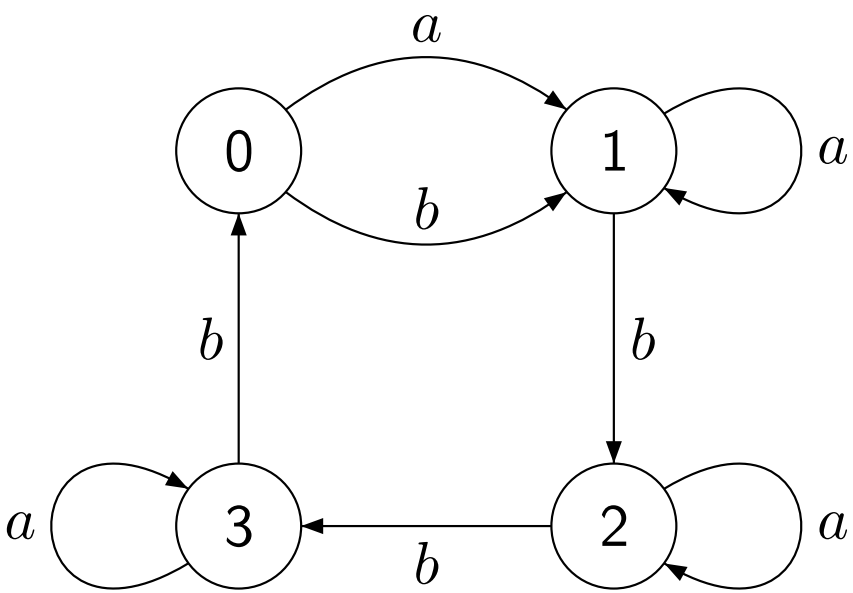
To lighten notation we use qw for $\delta(q, w)$ and $\langle Q, \Sigma \rangle$ for $\langle Q, \Sigma, \delta \rangle$.

Visualization

We use a standard representation of DFAs as labeled digraphs.

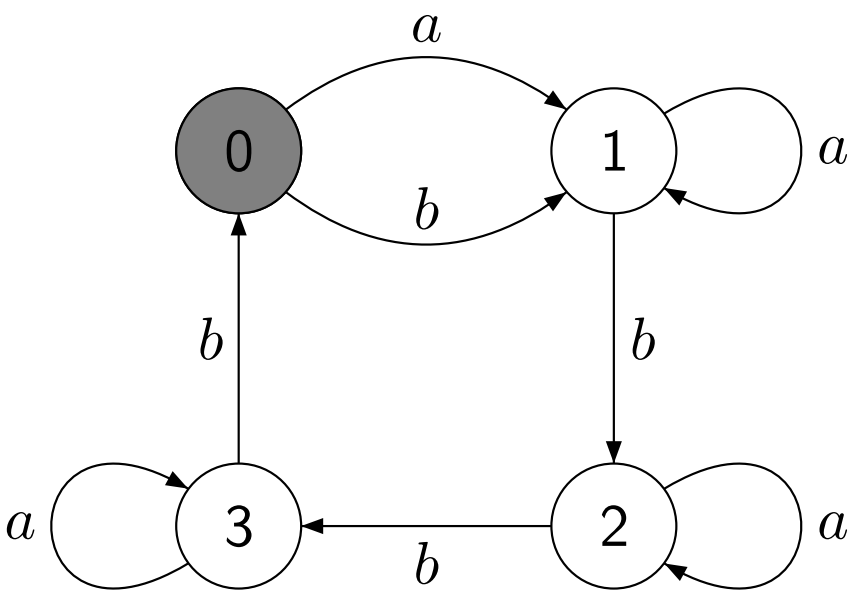
Visualization

We use a standard representation of DFAs as labeled digraphs.
A DFA $\langle Q, \Sigma \rangle$ is represented as a labeled digraph with vertex set Q and edges of the form $q \xrightarrow{a} qa$ for all $q \in Q$ and $a \in \Sigma$.



Visualization

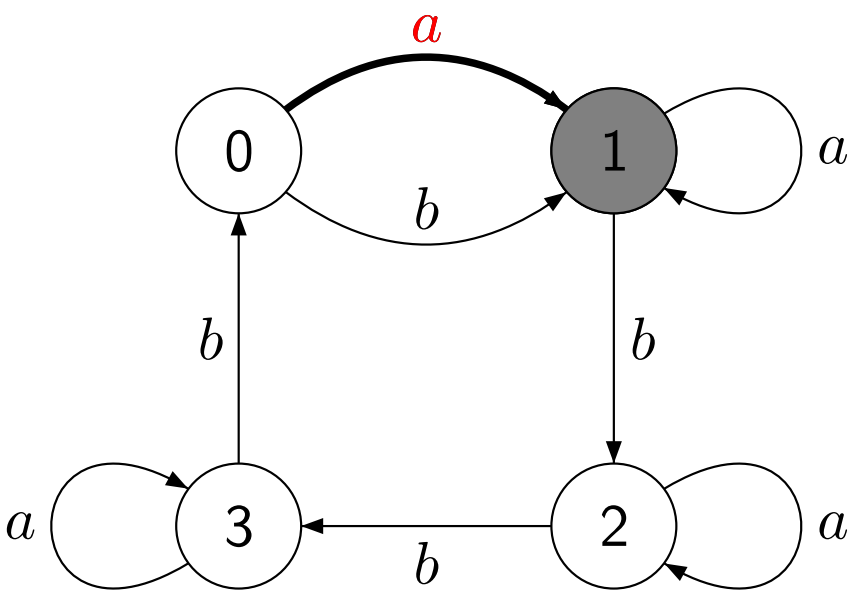
We use a standard representation of DFAs as labeled digraphs.
A DFA $\langle Q, \Sigma \rangle$ is represented as a labeled digraph with vertex set Q and edges of the form $q \xrightarrow{a} qa$ for all $q \in Q$ and $a \in \Sigma$.



The fact that the automaton is in certain state can be visualized by putting a token on this state; say, now the automaton is in state 0.

Visualization

We use a standard representation of DFAs as labeled digraphs.
A DFA $\langle Q, \Sigma \rangle$ is represented as a labeled digraph with vertex set Q and edges of the form $q \xrightarrow{a} qa$ for all $q \in Q$ and $a \in \Sigma$.



The action of a makes the token slide to state 1.

Synchronizing automata

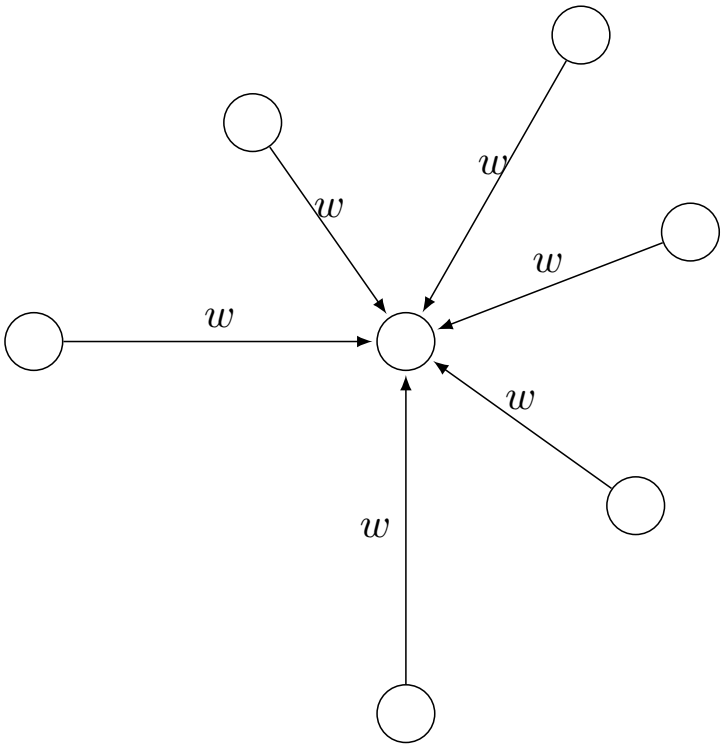
An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state q no matter at which state it started: $pw = q$ for all $p \in Q$.

Synchronizing automata

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state q no matter at which state it started: $pw = q$ for all $p \in Q$. Any such word w is a reset word for $\mathscr{A}$.

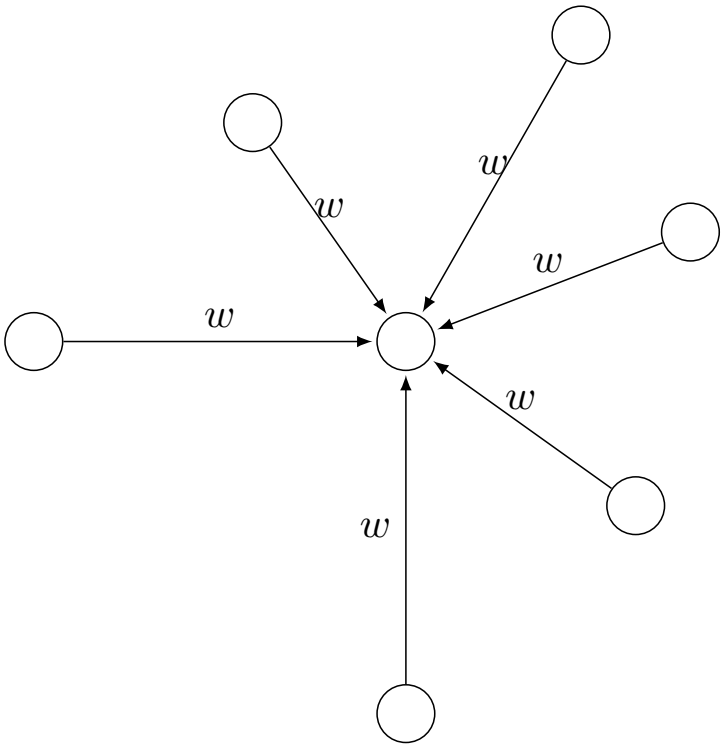
Synchronizing automata

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state q no matter at which state it started: $pw = q$ for all $p \in Q$. Any such word w is a reset word for $\mathscr{A}$.



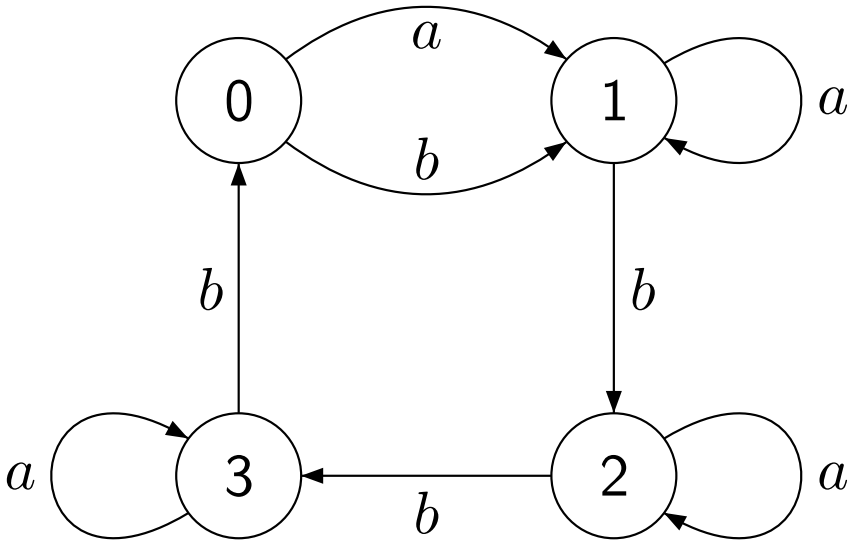
Synchronizing automata

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is synchronizing if there is a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state q no matter at which state it started: $pw = q$ for all $p \in Q$. Any such word w is a reset word for $\mathscr{A}$.

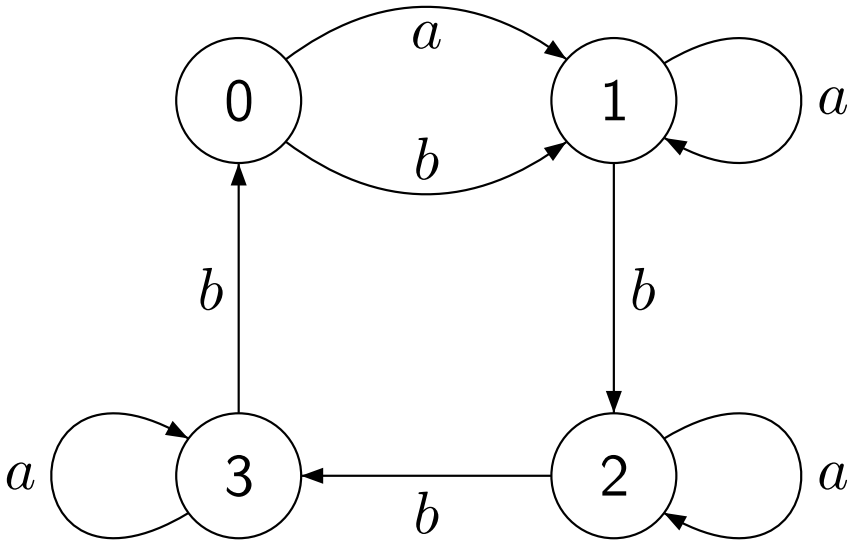


The minimum length of reset words is the reset threshold of $\mathscr{A}$.

Example

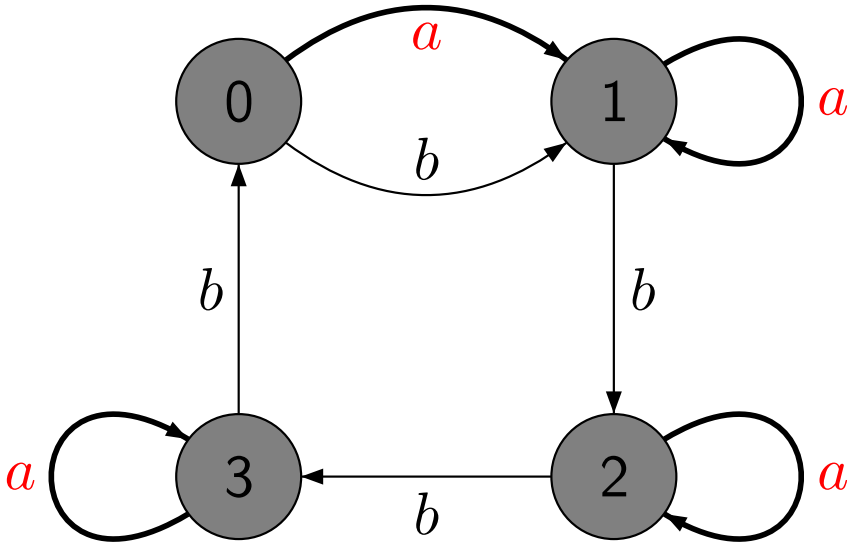


Example



A reset word is $abbbabbba$: it brings any state to the state 1.

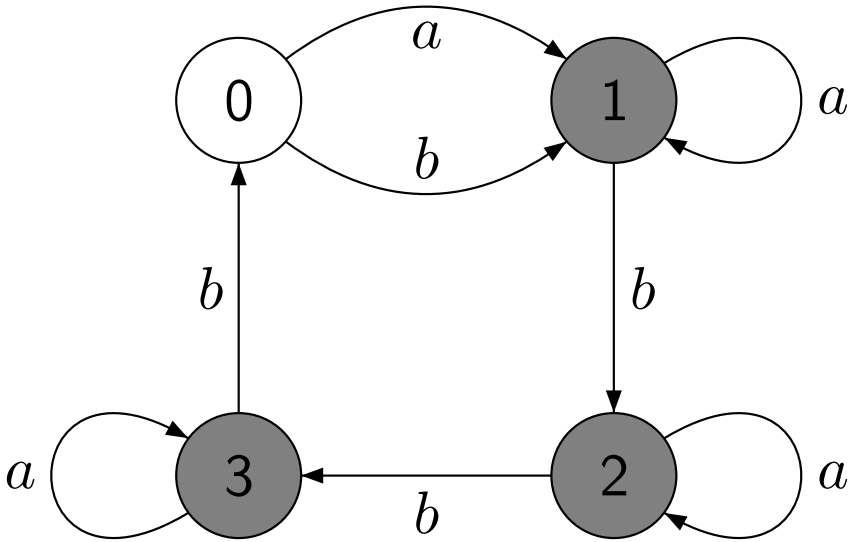
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

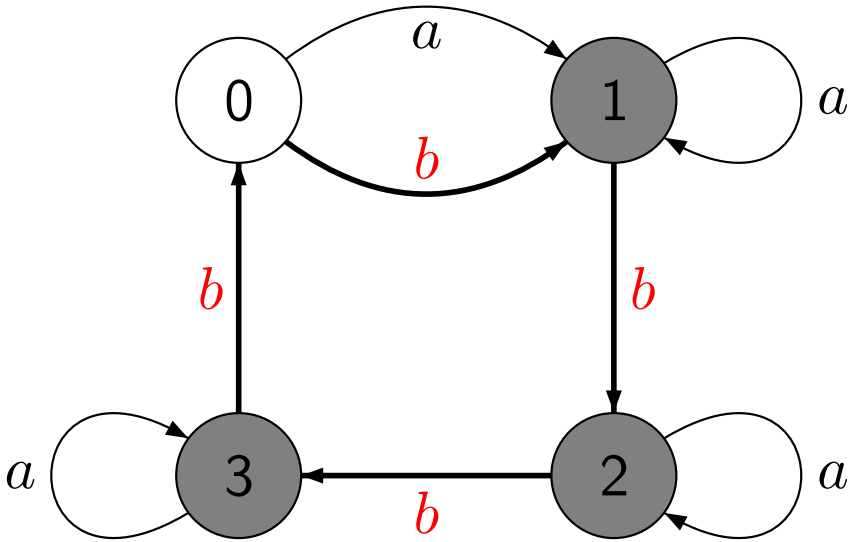
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

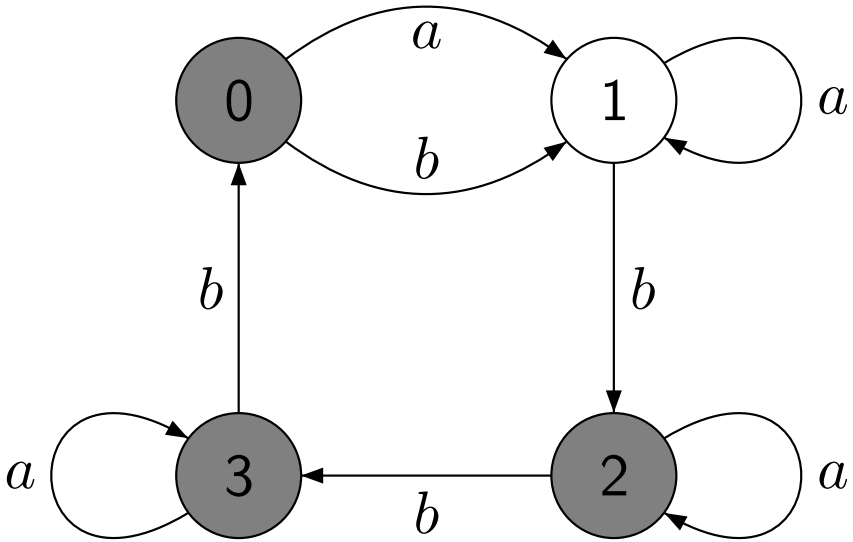
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

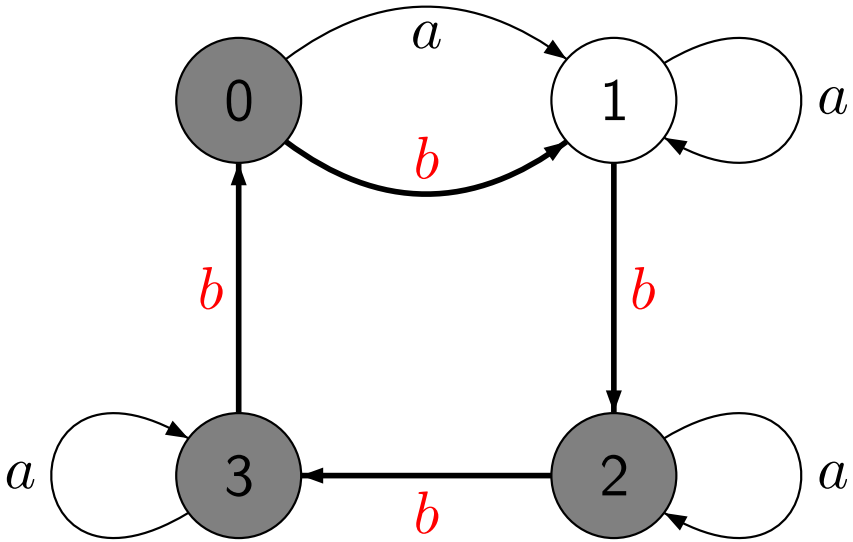
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

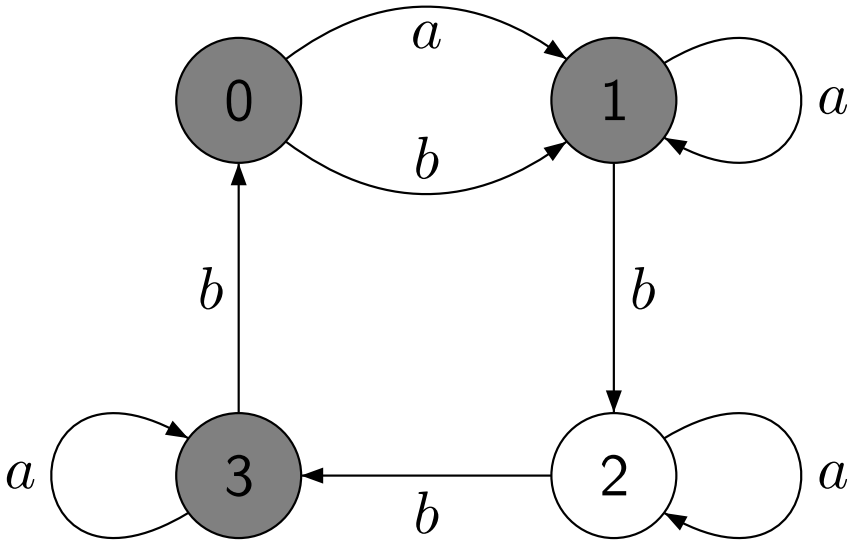
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

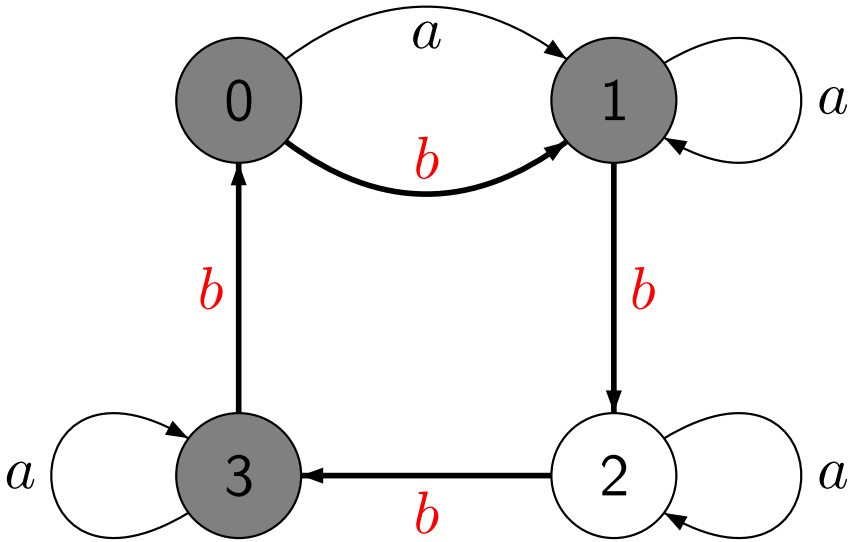
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

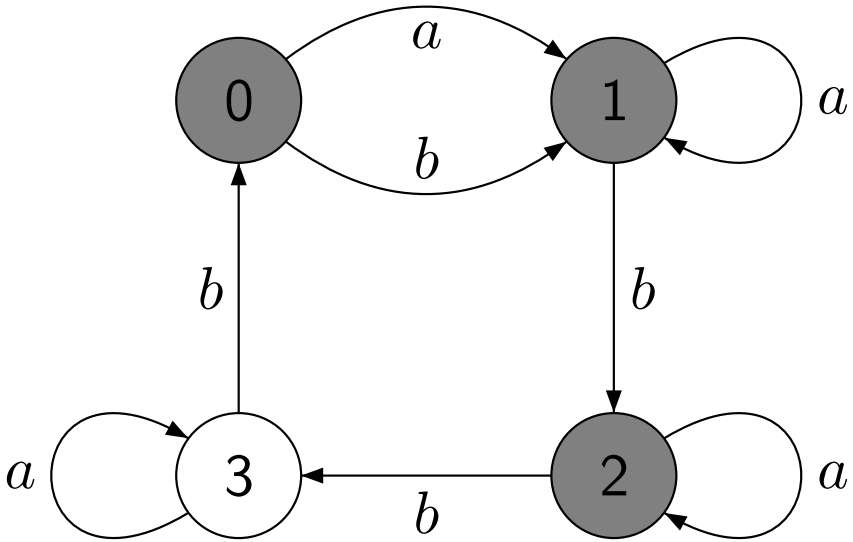
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

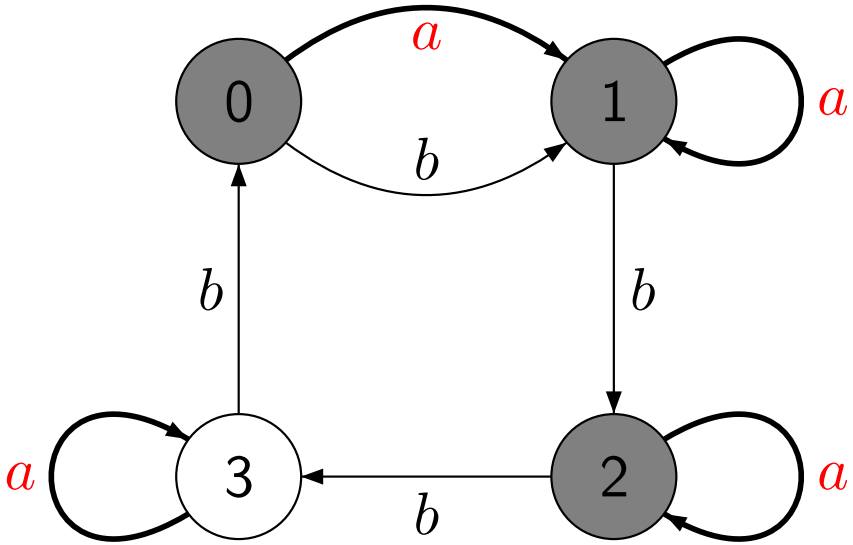
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

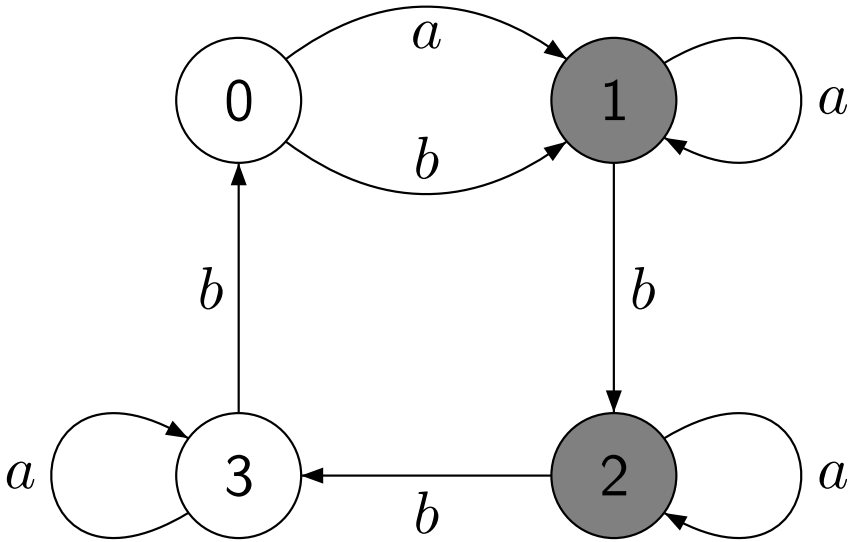
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

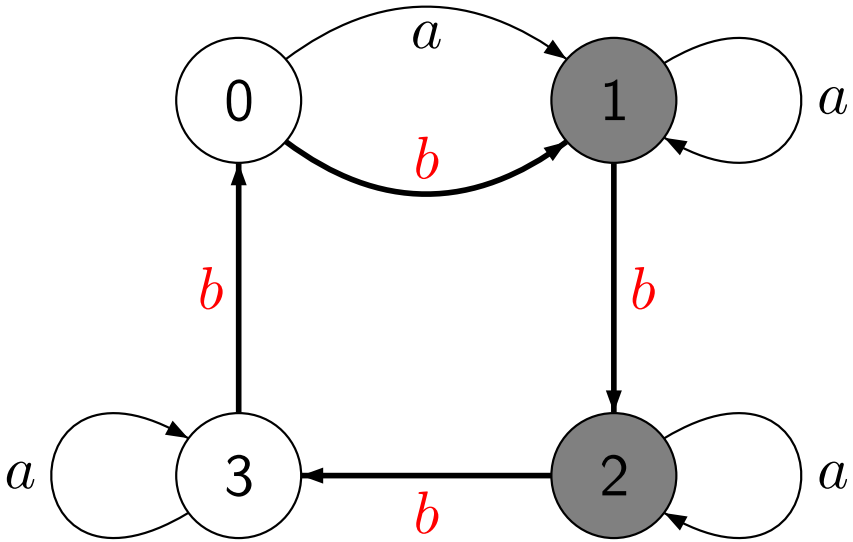
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

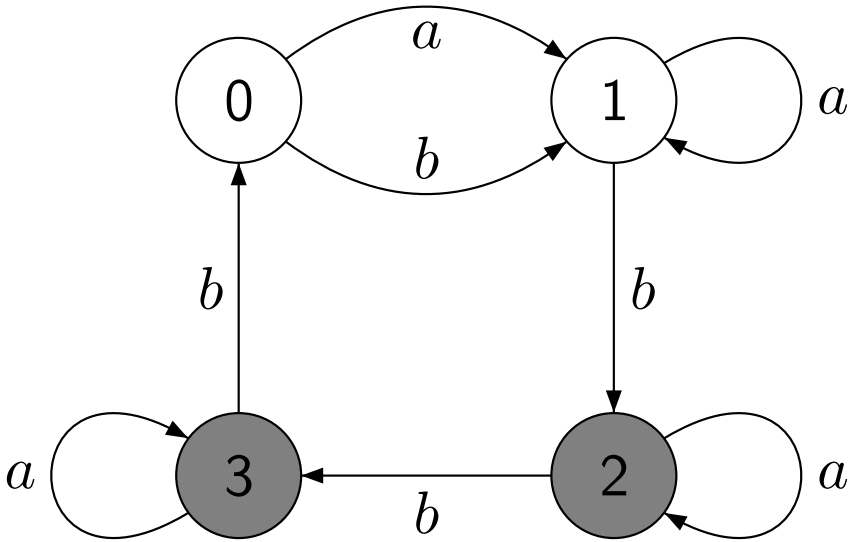
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

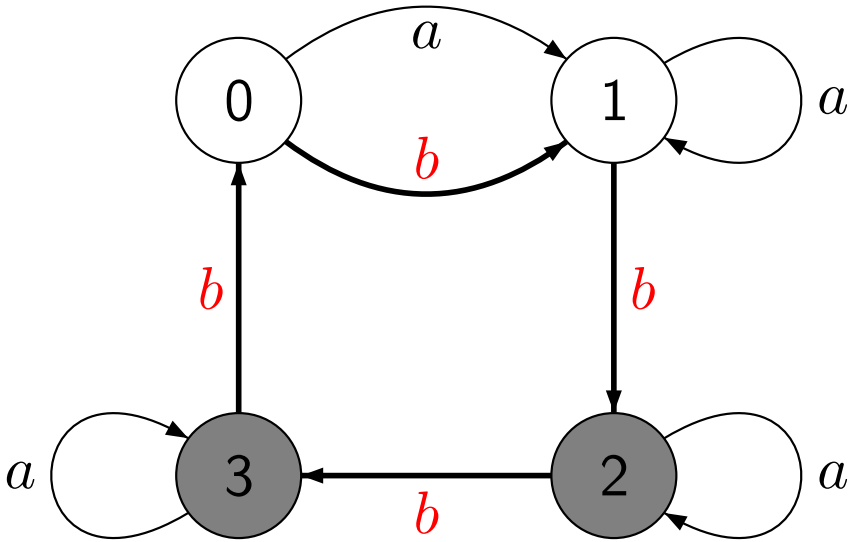
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

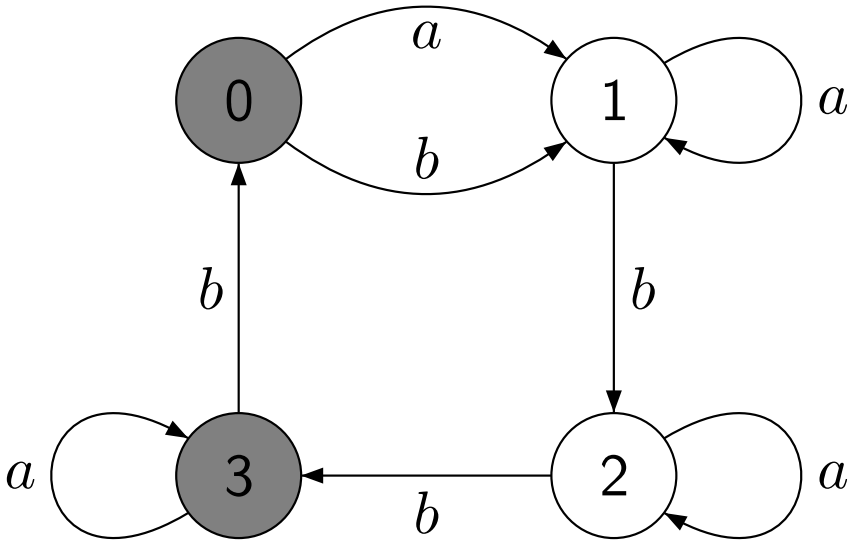
$abbbabbba$

Example



A reset word is $abbbabbba$: it brings any state to the state 1.
$abbbabbba$

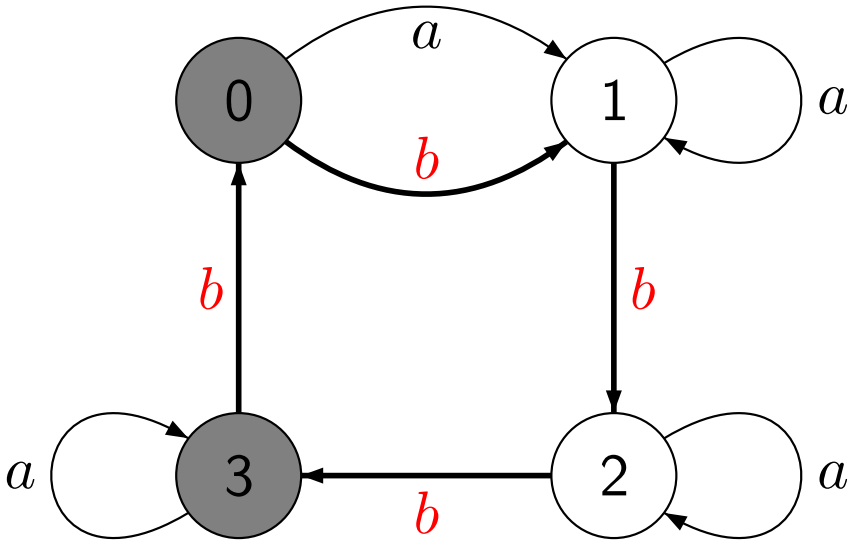
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

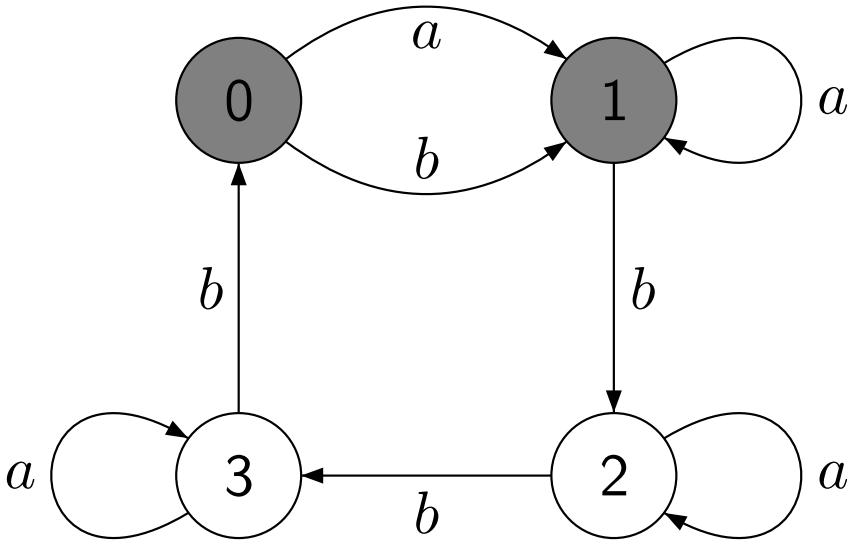
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

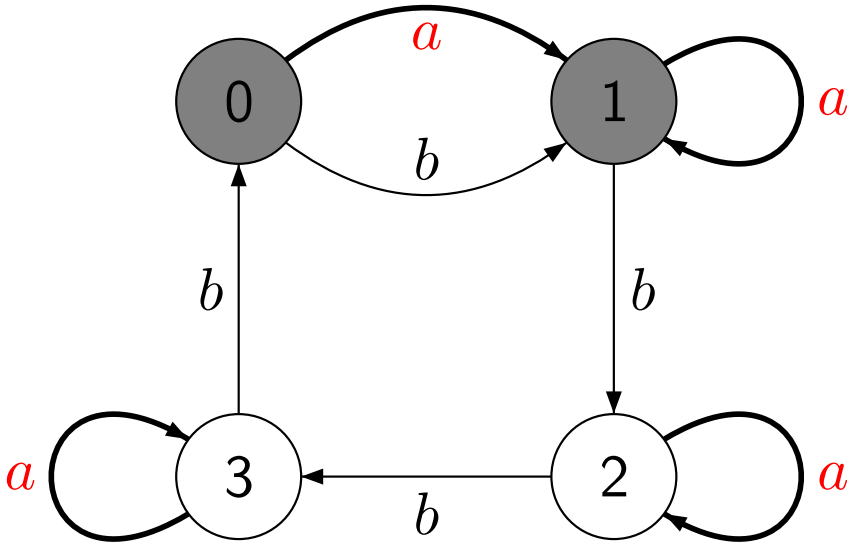
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

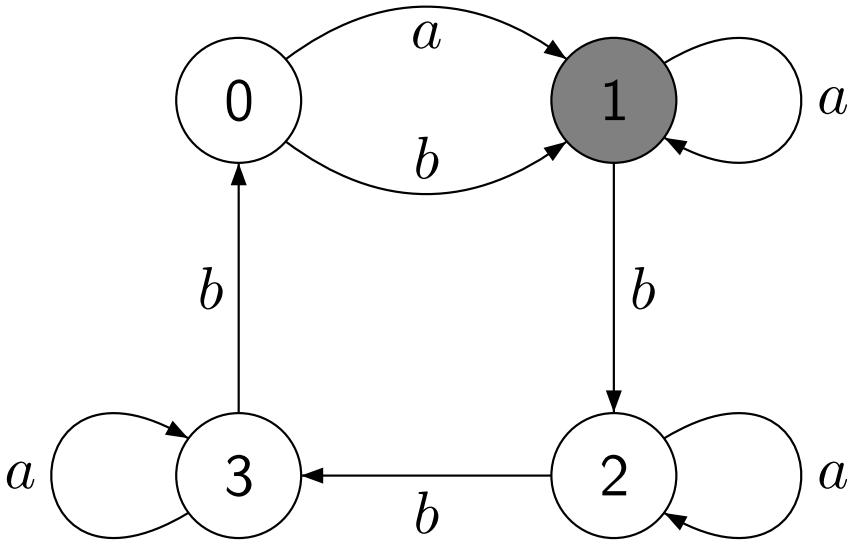
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

$abbbabbba$

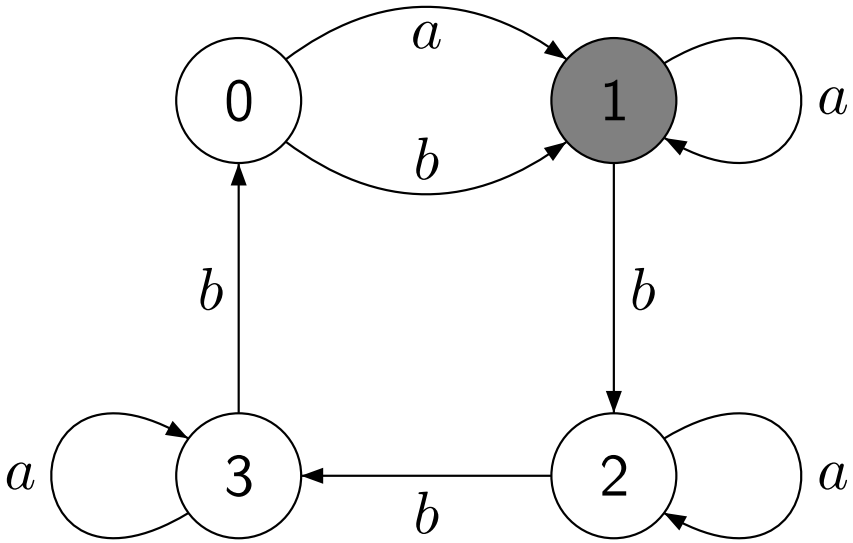
Example



A reset word is $abbbabbba$: it brings any state to the state 1.

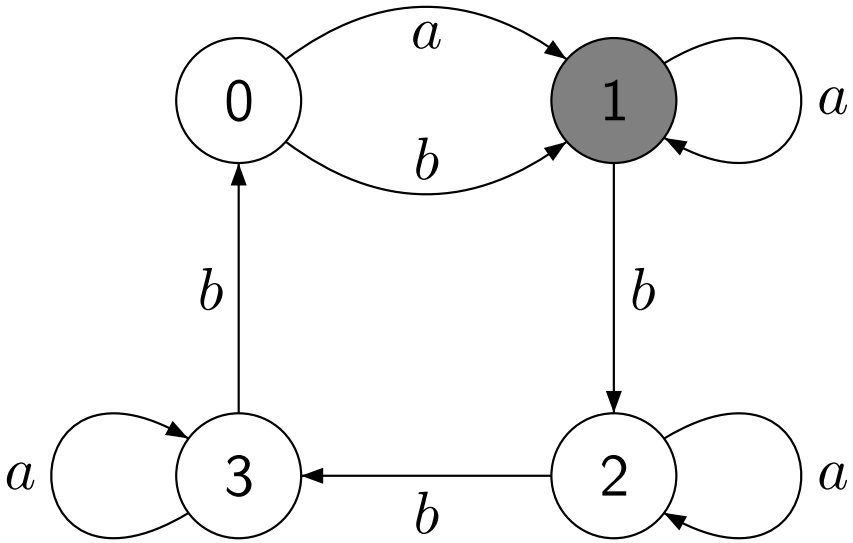
$abbbabbba$

Example



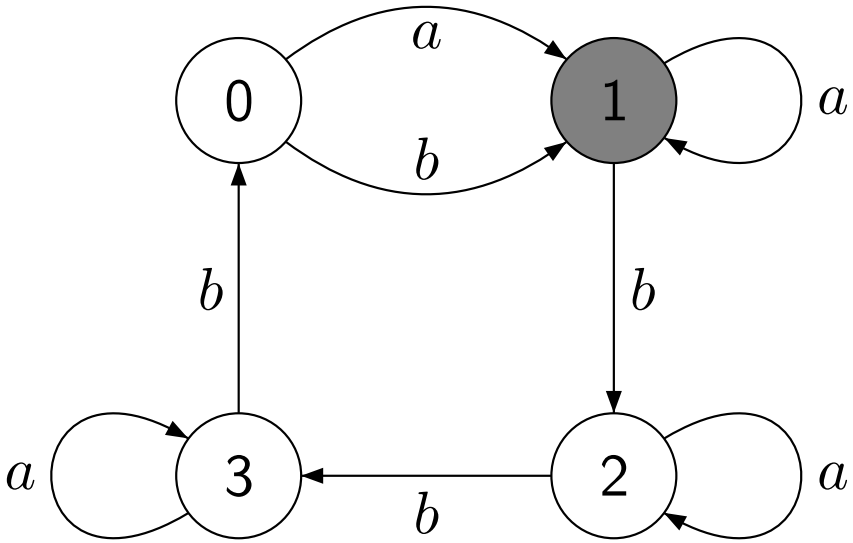
A reset word is $abbbabbba$: it brings any state to the state 1.
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.

Example



A reset word is $abbbabbba$: it brings any state to the state 1.
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.
The automaton belongs to the series $\mathscr{C}_n$ found by Černý in 1964.

Example



A reset word is $abbbabbba$: it brings any state to the state 1.
In fact, this is the reset word of minimum length for the automaton whence the reset threshold of the automaton is 9.
The automaton belongs to the series $\mathscr{C}_n$ found by Černý in 1964.
For each $n > 1$, the automaton $\mathscr{C}_n$ has n states, 2 input letters and reset threshold $(n-1)^2$.

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

$$0a = 1, \quad ma = m \text{ for } 0 < m < n, \quad mb = m + 1 \pmod{n}.$$

Černý series

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

$$0a = 1, \quad ma = m \text{ for } 0 < m < n, \quad mb = m + 1 \pmod{n}.$$

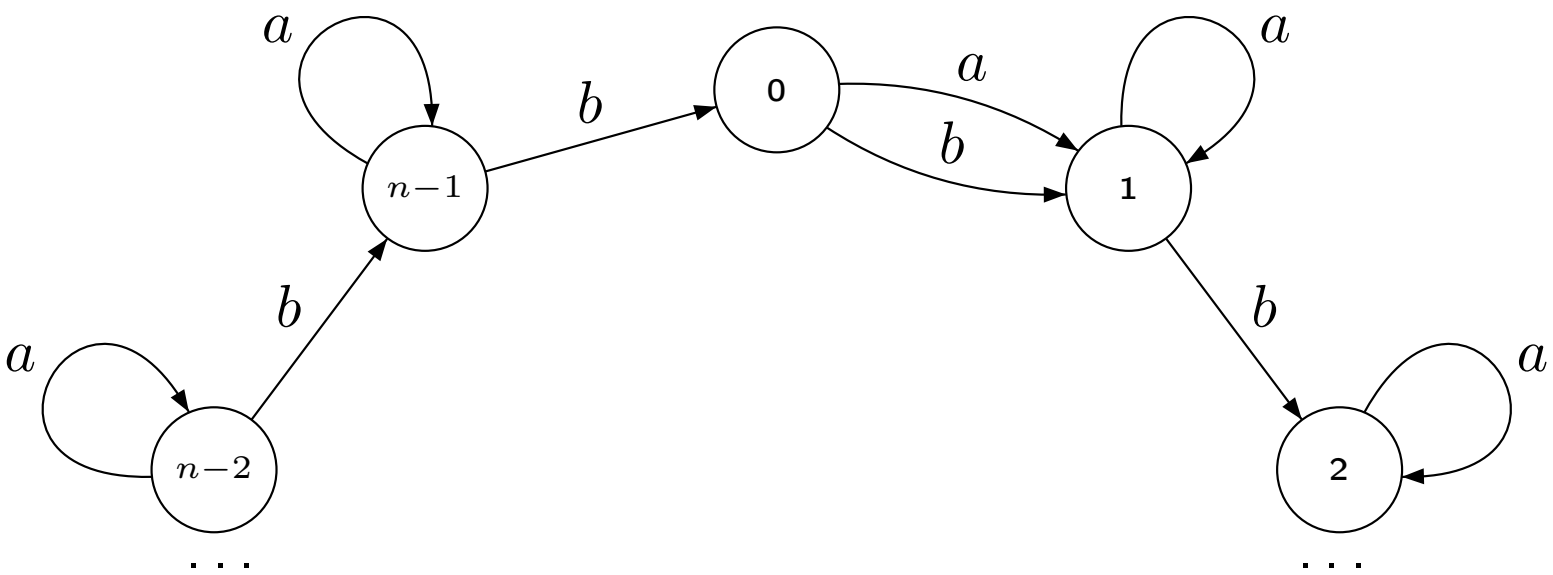
The automaton in the previous slide is $\mathscr{C}_4$.

Černý series

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

$$0a = 1, \quad ma = m \text{ for } 0 < m < n, \quad mb = m + 1 \pmod{n}.$$

The automaton in the previous slide is $\mathscr{C}_4$.
Here is a generic automaton from the Černý series:

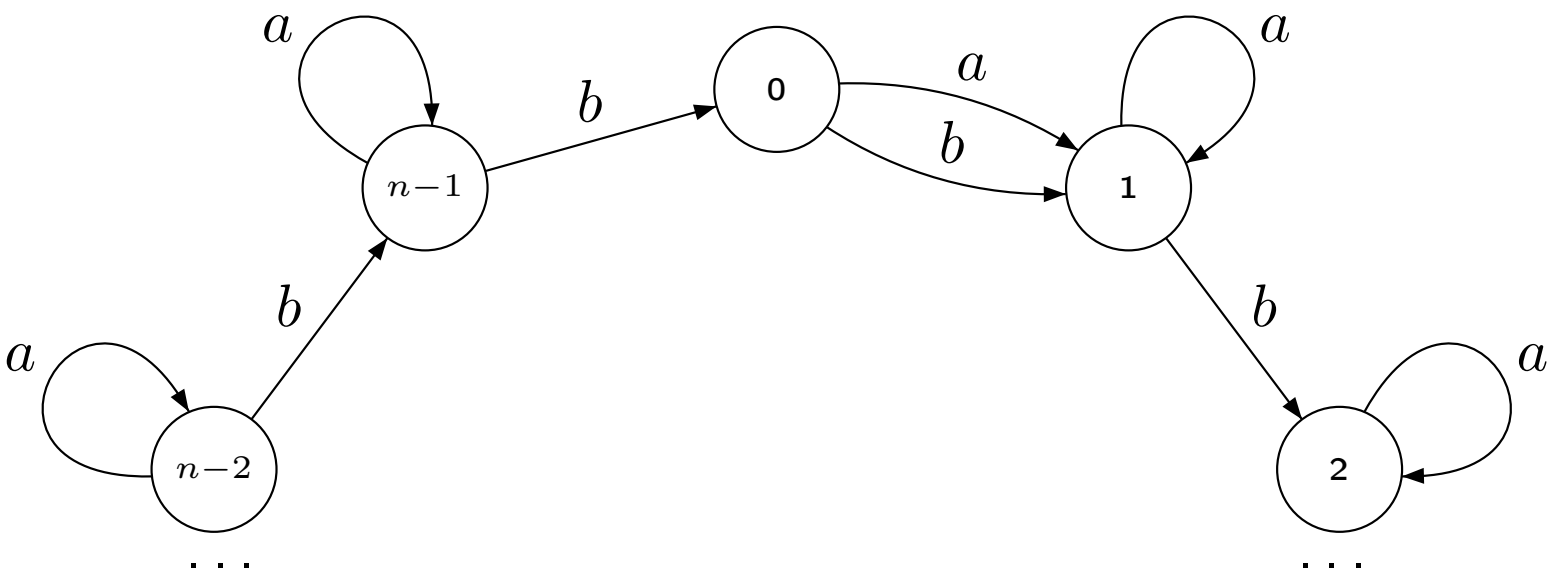


Černý series

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

$$0a = 1, \quad ma = m \text{ for } 0 < m < n, \quad mb = m + 1 \pmod{n}.$$

The automaton in the previous slide is $\mathscr{C}_4$.
Here is a generic automaton from the Černý series:



Černý has proved that the shortest reset word for $\mathscr{C}_n$ is $(ab^{n-1})^{n-2}a$ of length $n(n-2)+1 = (n-1)^2$.

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, yields the inequality $C(n) \geq (n-1)^2$.

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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.

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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.

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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. Up to 2018, everything we knew about the conjecture in general could be summarized in one line:

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

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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. Up to 2018, everything we knew about the conjecture in general could be summarized in one line:

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

A small improvement on this bound has been found by Marek Szykuła (published in STACS 2018): his bound is still cubic in n but improves the coefficient $\frac{1}{6} = 0.1666\ldots$ at n^3

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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. Up to 2018, everything we knew about the conjecture in general could be summarized in one line:

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

A small improvement on this bound has been found by Marek Szykuła (published in STACS 2018): his bound is still cubic in n but improves the coefficient $\frac{1}{6} = 0.1666\ldots$ at n^3 by $\frac{125}{511104} \approx 0.000245$ so that it becomes ≈ 0.1664.

Černý conjecture

Define the Černý function $C(n)$ as the maximum reset threshold of all synchronizing automata with n states. The above property of the series $\{\mathscr{C}_n\}$, 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. Up to 2018, everything we knew about the conjecture in general could be summarized in one line:

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

A small improvement on this bound has been found by Marek Szykuła (published in STACS 2018): his bound is still cubic in n but improves the coefficient $\frac{1}{6} = 0.1666\ldots$ at n^3 by $\frac{125}{511104} \approx 0.000245$ so that it becomes ≈ 0.1664.
In 2019 Yaroslav Shitov improved it to ≈ 0.1654.

Teaser

Synchronizing automata serve as transparent and natural models of error-resistant systems in numerous applications, including coding theory, robotics, and the testing of reactive systems.

Teaser

Synchronizing automata serve as transparent and natural models of error-resistant systems in numerous applications, including coding theory, robotics, and the testing of reactive systems. They also reveal fascinating connections with symbolic dynamics, substitution systems, permutation groups, and other areas of math.

Teaser

Synchronizing automata serve as transparent and natural models of error-resistant systems in numerous applications, including coding theory, robotics, and the testing of reactive systems. They also reveal fascinating connections with symbolic dynamics, substitution systems, permutation groups, and other areas of math.

For an introduction to the field and an overview of its state-of-the-art, see "Černý's conjecture and the Road Colouring Problem" (Chapter 15 of the "Handbook of Automata Theory", 2021) by Jarkko Kari and MV, "Synchronization of finite automata" (Russian Mathematical Surveys, 2022) and the living survey "List of results on the Černý Conjecture and reset thresholds for synchronizing automata" (arXiv:2508.15655v4) by MV.

Teaser

Synchronizing automata serve as transparent and natural models of error-resistant systems in numerous applications, including coding theory, robotics, and the testing of reactive systems. They also reveal fascinating connections with symbolic dynamics, substitution systems, permutation groups, and other areas of math.

For an introduction to the field and an overview of its state-of-the-art, see "Černý's conjecture and the Road Colouring Problem" (Chapter 15 of the "Handbook of Automata Theory", 2021) by Jarkko Kari and MV, "Synchronization of finite automata" (Russian Mathematical Surveys, 2022) and the living survey "List of results on the Černý Conjecture and reset thresholds for synchronizing automata" (arXiv:2508.15655v4) by MV.

The MathSciNet reviewer of the above chapter complained, "The only topic that I missed is this survey is the one related to synchronization games".

Teaser

Synchronizing automata serve as transparent and natural models of error-resistant systems in numerous applications, including coding theory, robotics, and the testing of reactive systems. They also reveal fascinating connections with symbolic dynamics, substitution systems, permutation groups, and other areas of math.

For an introduction to the field and an overview of its state-of-the-art, see "Černý's conjecture and the Road Colouring Problem" (Chapter 15 of the "Handbook of Automata Theory", 2021) by Jarkko Kari and MV, "Synchronization of finite automata" (Russian Mathematical Surveys, 2022) and the living survey "List of results on the Černý Conjecture and reset thresholds for synchronizing automata" (arXiv:2508.15655v4) by MV.

The MathSciNet reviewer of the above chapter complained, "The only topic that I missed is this survey is the one related to synchronization games".
Well, let's speak about synchronization games.

Synchronization games

In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory.

Synchronization games

In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory. Our motivation came from a game-theoretical approach to software testing proposed by a group at Microsoft Research in 2006.

Synchronization games

In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory. Our motivation came from a game-theoretical approach to software testing proposed by a group at Microsoft Research in 2006.

Synchronizing automata model regaining control over a device whose current state is unknown. We aimed to model situations in which the task is complicated by the presence of an adversary who actively resists our efforts.

Synchronization games

In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory. Our motivation came from a game-theoretical approach to software testing proposed by a group at Microsoft Research in 2006.

Synchronizing automata model regaining control over a device whose current state is unknown. We aimed to model situations in which the task is complicated by the presence of an adversary who actively resists our efforts.

We introduced and studied synchronization game, in which two players, Alice (Synchronizer) and Bob (Desynchronizer), take turns choosing letters from the input alphabet of an automaton $\mathscr{A}$.

Synchronization games

In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory. Our motivation came from a game-theoretical approach to software testing proposed by a group at Microsoft Research in 2006.

Synchronizing automata model regaining control over a device whose current state is unknown. We aimed to model situations in which the task is complicated by the presence of an adversary who actively resists our efforts.

We introduced and studied synchronization game, in which two players, Alice (Synchronizer) and Bob (Desynchronizer), take turns choosing letters from the input alphabet of an automaton $\mathscr{A}$. Alice who wants to synchronize $\mathscr{A}$ wins when the sequence of chosen letters forms a reset word for $\mathscr{A}$.

Synchronization games

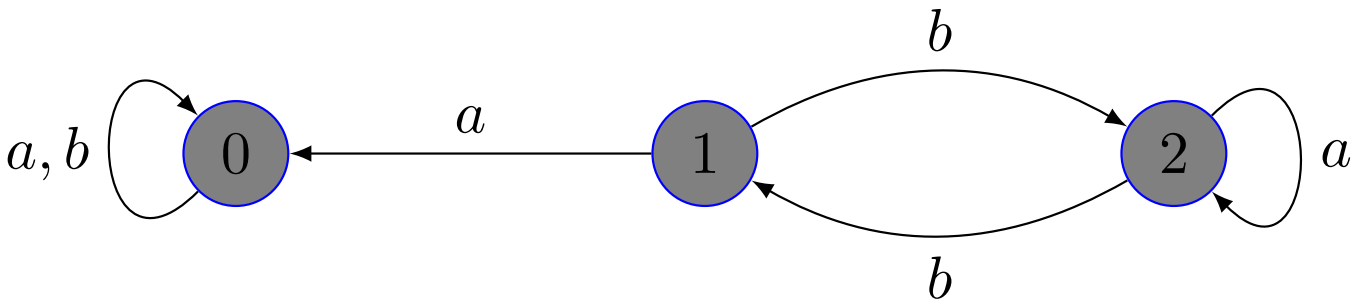
In a 2012 paper, Fëdor Fominykh and MV initiated the study of synchronizing automata through the lens of game theory. Our motivation came from a game-theoretical approach to software testing proposed by a group at Microsoft Research in 2006.

Synchronizing automata model regaining control over a device whose current state is unknown. We aimed to model situations in which the task is complicated by the presence of an adversary who actively resists our efforts.

We introduced and studied synchronization game, in which two players, Alice (Synchronizer) and Bob (Desynchronizer), take turns choosing letters from the input alphabet of an automaton $\mathscr{A}$. Alice who wants to synchronize $\mathscr{A}$ wins when the sequence of chosen letters forms a reset word for $\mathscr{A}$. Bob, on the other hand, strives to prevent synchronization or, if synchronization is unavoidable, to delay it as long as possible.

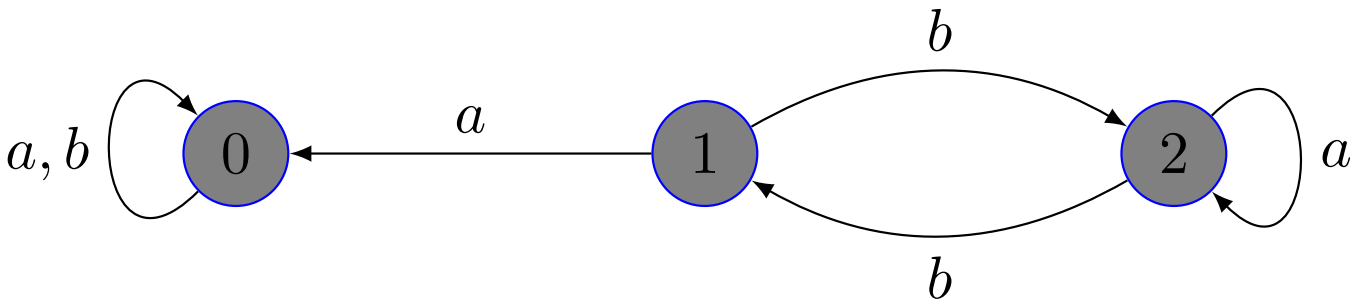
Synchronization game example

To visualize the game on $\mathscr{A} = \langle Q, \Sigma \rangle$, imagine that at the start, each state in Q holds a token.



Synchronization game example

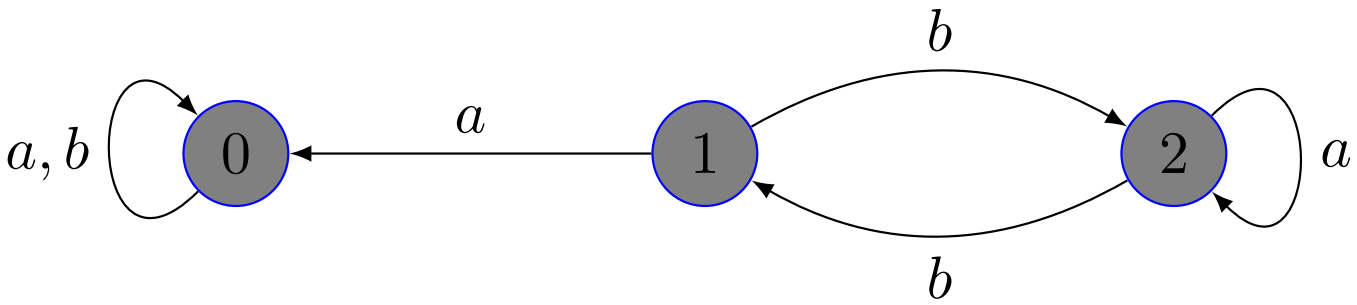
To visualize the game on $\mathscr{A} = \langle Q, \Sigma \rangle$, imagine that at the start, each state in Q holds a token.



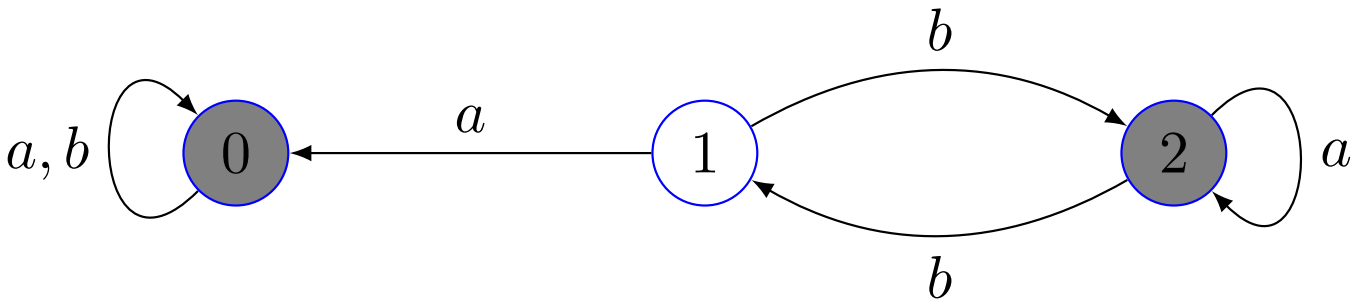
If Alice chooses the input a on her first move, the tokens in 0 and 2 remain due to the loops.

Synchronization game example

To visualize the game on $\mathscr{A} = \langle Q, \Sigma \rangle$, imagine that at the start, each state in Q holds a token.

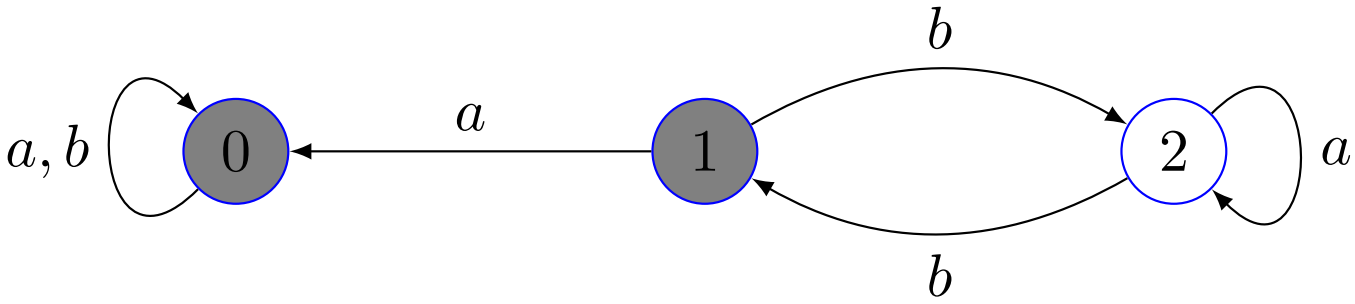


If Alice chooses the input a on her first move, the tokens in 0 and 2 remain due to the loops. The token from 1 moves to 0. Hence, the position after Alice's first move looks as follows:



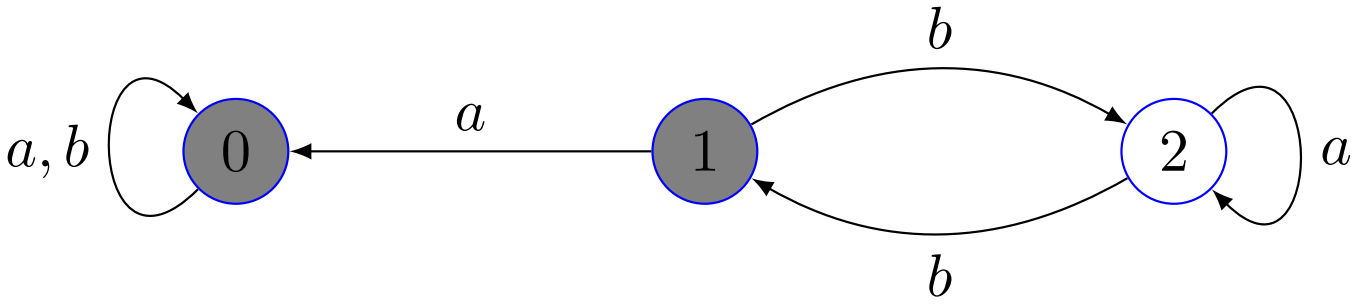
Synchronization game example, continued

If Bob responds by choosing the input b, the token in 0 remains while the token from 2 moves to 1. Here is the position after Bob's response:

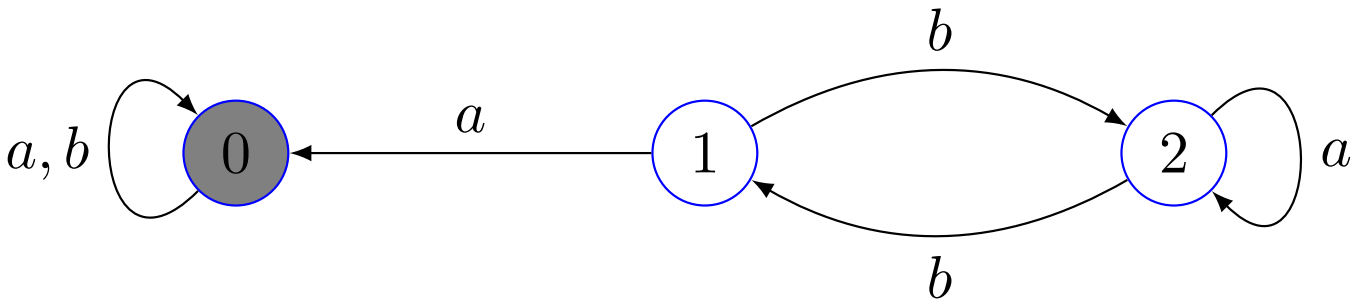


Synchronization game example, continued

If Bob responds by choosing the input b, the token in 0 remains while the token from 2 moves to 1. Here is the position after Bob's response:



Now choosing a, Alice wins because the token from 1 moves to 0, and we get the position with all tokens in one state:

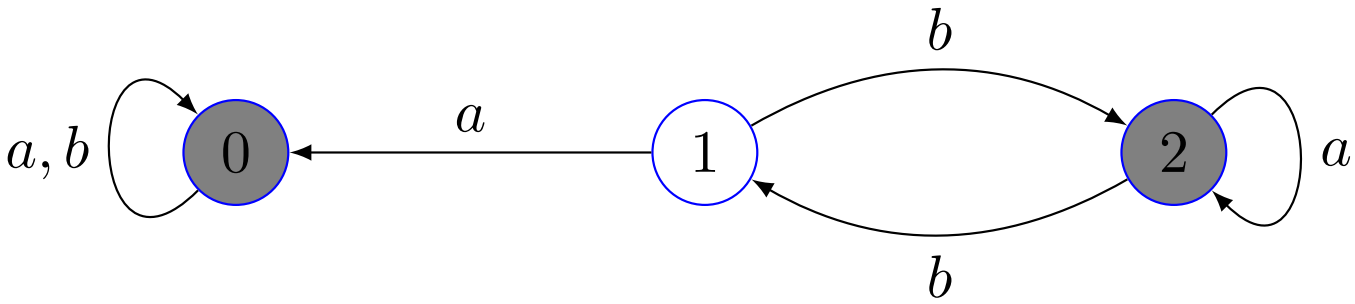


Synchronization games, discussion

Notice that Alice won the game described above only because of Bob's unfortunate response.

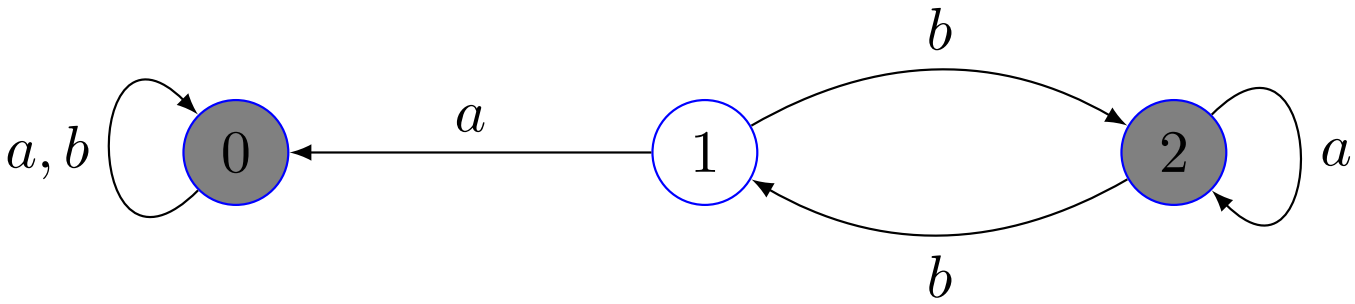
Synchronization games, discussion

Notice that Alice won the game described above only because of Bob's unfortunate response. In fact, Bob has a winning strategy in the synchronization game on this automaton: if Bob repeats Alice's moves, that is, chooses the same letter Alice chose on her previous move, he can maintain two tokens in different states forever:



Synchronization games, discussion

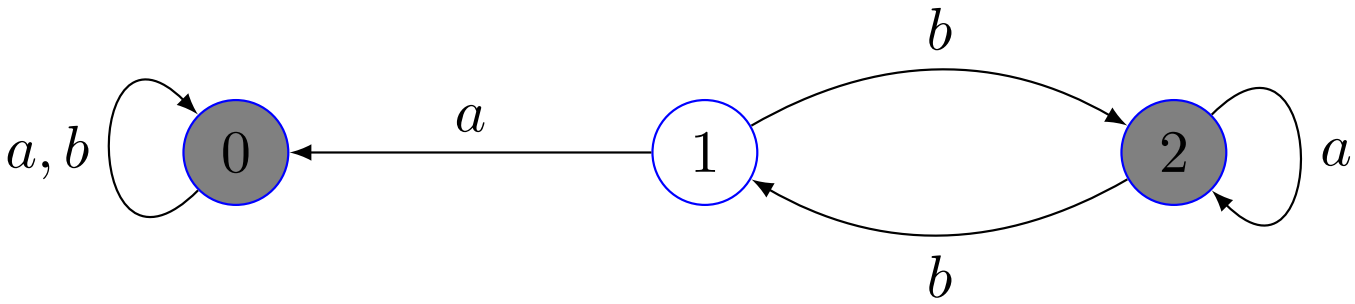
Notice that Alice won the game described above only because of Bob's unfortunate response. In fact, Bob has a winning strategy in the synchronization game on this automaton: if Bob repeats Alice's moves, that is, chooses the same letter Alice chose on her previous move, he can maintain two tokens in different states forever:



If both players play optimally, the outcome of the game depends solely on the automaton.

Synchronization games, discussion

Notice that Alice won the game described above only because of Bob's unfortunate response. In fact, Bob has a winning strategy in the synchronization game on this automaton: if Bob repeats Alice's moves, that is, chooses the same letter Alice chose on her previous move, he can maintain two tokens in different states forever:



If both players play optimally, the outcome of the game depends solely on the automaton. Our 2012 paper and its follow-ups addressed the problem of classifying automata into those for which Alice has a winning strategy and those for which Bob does.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely. The goals of Alice and Bob remain as they were before (synchronize/prevent synchronization).

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely. The goals of Alice and Bob remain as they were before (synchronize/prevent synchronization).

Modified rules appear more favorable to Bob and indeed they are.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely. The goals of Alice and Bob remain as they were before (synchronize/prevent synchronization).

Modified rules appear more favorable to Bob and indeed they are. It is easy to construct examples of automata on which Bob wins the modified game while Alice wins under "old" rules.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely. The goals of Alice and Bob remain as they were before (synchronize/prevent synchronization).

Modified rules appear more favorable to Bob and indeed they are. It is easy to construct examples of automata on which Bob wins the modified game while Alice wins under "old" rules.

For brevity, let A-automata be automata that admit a winning strategy for Alice under the modified rules.

Adversarial synchronization

The drawback of the above game is that it misses its original aim of modelling synchronization against an adversary's resistance. Indeed, nature or an adversary does not necessarily act in the same rhythm as us, nor are they required to respond to each of our actions with exactly one obstacle or counteraction.

So, let's modify the rules such that Bob is allowed to make an arbitrary finite sequence of moves in response to each of Alice's moves, or even to skip a move entirely. The goals of Alice and Bob remain as they were before (synchronize/prevent synchronization).

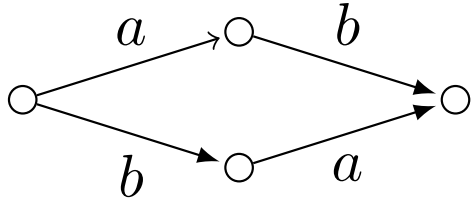
Modified rules appear more favorable to Bob and indeed they are. It is easy to construct examples of automata on which Bob wins the modified game while Alice wins under "old" rules.

For brevity, let A-automata be automata that admit a winning strategy for Alice under the modified rules. Do A-automata exist?

A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

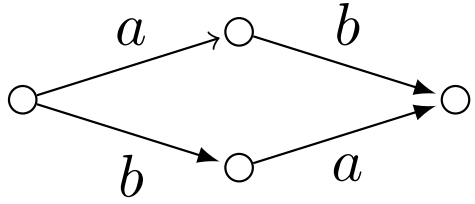
$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:



A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:

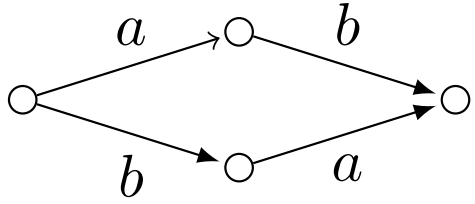


Any commutative synchronizing automaton is an A-automaton.

A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:



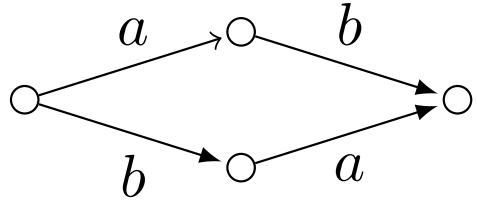
Any commutative synchronizing automaton is an A-automaton.

If $w = a_1 a_2 \cdots a_k$ is a reset word for $\mathscr{A}$, Alice wins by choosing the input a_i on her i-th move, regardless of Bob's responses.

A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:



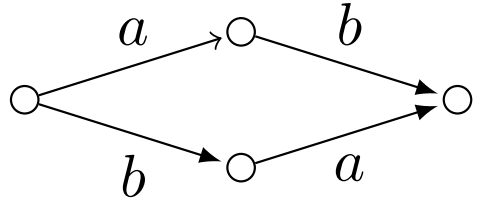
Any commutative synchronizing automaton is an A-automaton.

If $w = a_1 a_2 \cdots a_k$ is a reset word for $\mathscr{A}$, Alice wins by choosing the input a_i on her i-th move, regardless of Bob's responses. This results in a word of the form $a_1 B_1 a_2 B_2 \cdots B_{k-1} a_k$ where B_i is the word chosen by Bob on his i-th move.

A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:



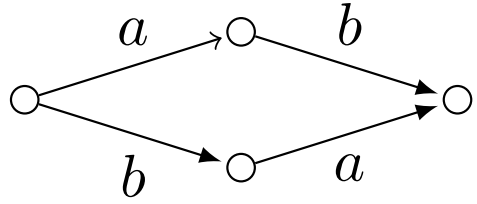
Any commutative synchronizing automaton is an A-automaton.

If $w = a_1 a_2 \cdots a_k$ is a reset word for $\mathscr{A}$, Alice wins by choosing the input a_i on her i-th move, regardless of Bob's responses. This results in a word of the form $a_1 B_1 a_2 B_2 \cdots B_{k-1} a_k$ where B_i is the word chosen by Bob on his i-th move. Due to commutativity, it acts the same as the word $w B_1 B_2 \cdots B_{k-1}$ which is a reset word.

A-Automata: Simple example

An automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ is said to be commutative if

$qab = qba$ for all $a, b \in \Sigma$ and $q \in Q$:



Any commutative synchronizing automaton is an A-automaton.

If $w = a_1 a_2 \cdots a_k$ is a reset word for $\mathscr{A}$, Alice wins by choosing the input a_i on her i-th move, regardless of Bob's responses. This results in a word of the form $a_1 B_1 a_2 B_2 \cdots B_{k-1} a_k$ where B_i is the word chosen by Bob on his i-th move. Due to commutativity, it acts the same as the word $w B_1 B_2 \cdots B_{k-1}$ which is a reset word.

In fact, the class of A-automata is much more extensive than that of commutative synchronizing automata and includes many types of synchronizing automata studied in the literature.

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata.

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

The problem remains meaningful (and often highly non-trivial) when restricted to synchronizing automata from various subclasses.

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

The problem remains meaningful (and often highly non-trivial) when restricted to synchronizing automata from various subclasses.

What about A-automata?

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

The problem remains meaningful (and often highly non-trivial) when restricted to synchronizing automata from various subclasses.

What about A-automata? A-automata appear to be more amenable to synchronization, as they can be synchronized even when an adversary tries to prevent it.

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

The problem remains meaningful (and often highly non-trivial) when restricted to synchronizing automata from various subclasses.

What about A-automata? A-automata appear to be more amenable to synchronization, as they can be synchronized even when an adversary tries to prevent it. Consequently, it is natural to expect that in the absence of resistance, they should be quickly synchronizable—that is, they admit short reset words.

Reset threshold of A-automata

Recall: a major open problem about synchronizing automata is to upper bound the reset threshold for all such automata. (Conjectured bound: $(n-1)^2$, where n is the number of states.)

The problem remains meaningful (and often highly non-trivial) when restricted to synchronizing automata from various subclasses.

What about A-automata? A-automata appear to be more amenable to synchronization, as they can be synchronized even when an adversary tries to prevent it. Consequently, it is natural to expect that in the absence of resistance, they should be quickly synchronizable—that is, they admit short reset words.

Our main result confirms this expectation:

Theorem

The reset threshold of every A-automaton is strictly less than the number of its states.

Proof idea

The proof is quite involved; see arXiv:2601.18362 for details.

Proof idea

The proof is quite involved; see arXiv:2601.18362 for details.

In a rough approximation, the idea of the proof is to assign to each position in the synchronization game on an A-automaton $\langle Q, \Sigma \rangle$ the value of a certain parameter with the following properties:

- Alice always has a move that decreases the current value (in an appropriate sense), while no move of Bob can increase it;
- the total number of values of this parameter is less than $|Q|$.

Proof idea

The proof is quite involved; see arXiv:2601.18362 for details.

In a rough approximation, the idea of the proof is to assign to each position in the synchronization game on an A-automaton $\langle Q, \Sigma \rangle$ the value of a certain parameter with the following properties:

- Alice always has a move that decreases the current value (in an appropriate sense), while no move of Bob can increase it;
- the total number of values of this parameter is less than $|Q|$.

We will now discuss an application of the main result.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

$T(\mathscr{A})$ can alternatively be defined as the monoid of all maps $\tau_w : Q \to Q$ defined by $q \mapsto qw$ where $w \in \Sigma^*$.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

$T(\mathscr{A})$ can alternatively be defined as the monoid of all maps $\tau_w : Q \to Q$ defined by $q \mapsto qw$ where $w \in \Sigma^*$.

Synchronization is actually a property of transition monoids: $\mathscr{A}$ is synchronizing iff $T(\mathscr{A})$ contains a constant transformation.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

$T(\mathscr{A})$ can alternatively be defined as the monoid of all maps $\tau_w : Q \to Q$ defined by $q \mapsto qw$ where $w \in \Sigma^*$.

Synchronization is actually a property of transition monoids: $\mathscr{A}$ is synchronizing iff $T(\mathscr{A})$ contains a constant transformation.

Being an A-automaton is not a property of transition monoids.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

$T(\mathscr{A})$ can alternatively be defined as the monoid of all maps $\tau_w : Q \to Q$ defined by $q \mapsto qw$ where $w \in \Sigma^*$.

Synchronization is actually a property of transition monoids: $\mathscr{A}$ is synchronizing iff $T(\mathscr{A})$ contains a constant transformation.

Being an A-automaton is not a property of transition monoids. Every synchronizing automaton $\mathscr{A}$ can be converted into an A-automaton $\mathscr{A}'$ by adding a reset word w of $\mathscr{A}$ as an extra input of $\mathscr{A}'$. This does not change the transition monoid since the map τ_w is already contained in $T(\mathscr{A})$.

Transition monoids and synchronization

For any automaton $\mathscr{A} = \langle Q, \Sigma \rangle$ and each input $a \in \Sigma$, the map $\tau_a : Q \to Q$ defined by $q \mapsto qa$ is a transformation on the set Q.

The transition monoid $T(\mathscr{A})$ of $\mathscr{A}$ is the submonoid of the monoid of all transformations on Q generated by the set $\{\tau_a \mid a \in \Sigma\}$.

$T(\mathscr{A})$ can alternatively be defined as the monoid of all maps $\tau_w : Q \to Q$ defined by $q \mapsto qw$ where $w \in \Sigma^*$.

Synchronization is actually a property of transition monoids: $\mathscr{A}$ is synchronizing iff $T(\mathscr{A})$ contains a constant transformation.

Being an A-automaton is not a property of transition monoids. Every synchronizing automaton $\mathscr{A}$ can be converted into an A-automaton $\mathscr{A}'$ by adding a reset word w of $\mathscr{A}$ as an extra input of $\mathscr{A}'$. This does not change the transition monoid since the map τ_w is already contained in $T(\mathscr{A})$.

Still, certain properties of transition monoids (e.g. commutativity) may force a synchronizing automaton to be an A-automaton.

DS-automata

The set $\mathbf{DS}$ of all finite monoids with regular $\mathfrak{D}$-classes being subsemigroups is a variety playing a distinguished role in the algebraic theory of regular languages and computational complexity.

DS-automata

The set $\mathbf{DS}$ of all finite monoids with regular $\mathfrak{D}$-classes being subsemigroups is a variety playing a distinguished role in the algebraic theory of regular languages and computational complexity.

Recall: Green's relations on a monoid M

$$a \mathfrak{R} b \iff \exists s,t \in M : a = bs \wedge b = at\,;$$
$$a \mathfrak{L} b \iff \exists s,t \in M : a = sb \wedge b = ta\,;$$

$\mathfrak{D} := \mathfrak{R} \circ \mathfrak{L} = \mathfrak{L} \circ \mathfrak{R}$ is the least equivalence containing $\mathfrak{R}$ and $\mathfrak{L}$.

DS-automata

The set $\mathbf{DS}$ of all finite monoids with regular $\mathfrak{D}$-classes being subsemigroups is a variety playing a distinguished role in the algebraic theory of regular languages and computational complexity.

Recall: Green's relations on a monoid M

$$a \,\mathfrak{R}\, b \iff \exists s,t \in M : a = bs \wedge b = at\,;$$
$$a \,\mathfrak{L}\, b \iff \exists s,t \in M : a = sb \wedge b = ta\,;$$

$\mathfrak{D} := \mathfrak{R} \circ \mathfrak{L} = \mathfrak{L} \circ \mathfrak{R}$ is the least equivalence containing $\mathfrak{R}$ and $\mathfrak{L}$.
An element a of a semigroup S is regular if $asa = a$ for some $s \in S$.
A $\mathfrak{D}$-class D is called regular if it contains a regular element.
Then, every element of D is known to be regular (Green, 1951).

DS-automata

The set $\mathbf{DS}$ of all finite monoids with regular $\mathfrak{D}$-classes being subsemigroups is a variety playing a distinguished role in the algebraic theory of regular languages and computational complexity.

Recall: Green's relations on a monoid M

$$a \,\mathfrak{R}\, b \iff \exists s,t \in M : a = bs \wedge b = at\,;$$
$$a \,\mathfrak{L}\, b \iff \exists s,t \in M : a = sb \wedge b = ta\,;$$

$\mathfrak{D} := \mathfrak{R} \circ \mathfrak{L} = \mathfrak{L} \circ \mathfrak{R}$ is the least equivalence containing $\mathfrak{R}$ and $\mathfrak{L}$.
An element a of a semigroup S is regular if $asa = a$ for some $s \in S$.
A $\mathfrak{D}$-class D is called regular if it contains a regular element.
Then, every element of D is known to be regular (Green, 1951).

A DFA whose transition monoid lies in $\mathbf{DS}$ is a DS-automaton.

Proposition (Fernau, Haase, Hoffmann, MV, arXiv:2512.11007)

Each synchronizing DS-automaton is an A-automaton.

Application

Corollary

The reset threshold of every synchronizing DS-automaton is strictly less than the number of its states.

Application

Corollary

The reset threshold of every synchronizing DS-automaton is strictly less than the number of its states.

This result was first obtained by Almeida and Steinberg [Matrix mortality and the Černý–Pin conjecture. In: DLT 2009, LNCS, vol. 5583, pp. 67–80]. Their proof was based on the classical theory of linear representations of semigroups (the Munn–Ponizovsky theory).

Application

Corollary

The reset threshold of every synchronizing DS-automaton is strictly less than the number of its states.

This result was first obtained by Almeida and Steinberg [Matrix mortality and the Černý–Pin conjecture. In: DLT 2009, LNCS, vol. 5583, pp. 67–80]. Their proof was based on the classical theory of linear representations of semigroups (the Munn–Ponizovsky theory). Our proof is quite different, as it is purely combinatorial.

Application

Corollary

The reset threshold of every synchronizing DS-automaton is strictly less than the number of its states.

This result was first obtained by Almeida and Steinberg [Matrix mortality and the Černý–Pin conjecture. In: DLT 2009, LNCS, vol. 5583, pp. 67–80]. Their proof was based on the classical theory of linear representations of semigroups (the Munn–Ponizovsky theory). Our proof is quite different, as it is purely combinatorial.

An A-automaton need not be a DS-automaton.

Corollary

The reset threshold of every synchronizing DS-automaton is strictly less than the number of its states.

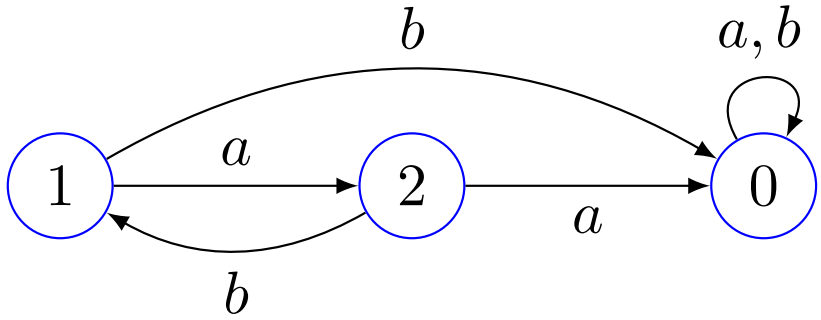
This result was first obtained by Almeida and Steinberg [Matrix mortality and the Černý–Pin conjecture. In: DLT 2009, LNCS, vol. 5583, pp. 67–80]. Their proof was based on the classical theory of linear representations of semigroups (the Munn–Ponizovsky theory). Our proof is quite different, as it is purely combinatorial.

An A-automaton need not be a DS-automaton. It is known that $\mathbf{DS}$ excludes the Brandt monoid B_2^1 consisting of the following six matrices, with standard matrix multiplication:

$$\begin{pmatrix}1&0\\0&0\end{pmatrix}, \begin{pmatrix}0&1\\0&0\end{pmatrix}, \begin{pmatrix}0&0\\1&0\end{pmatrix}, \begin{pmatrix}0&0\\0&1\end{pmatrix}, \begin{pmatrix}0&0\\0&0\end{pmatrix}, \begin{pmatrix}1&0\\0&1\end{pmatrix}.$$

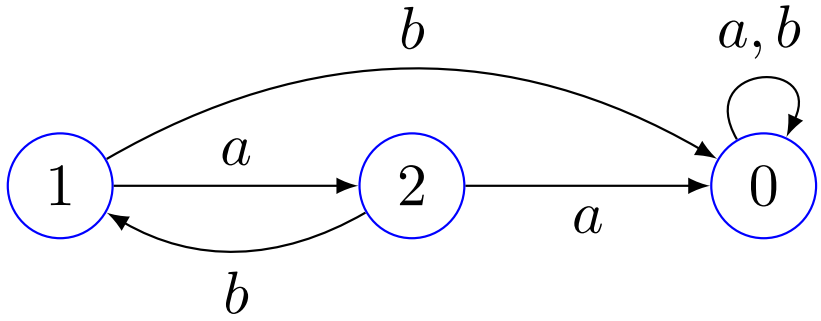
An A-automaton beyond $\mathbf{DS}$

B_2^1 is the transition monoid of

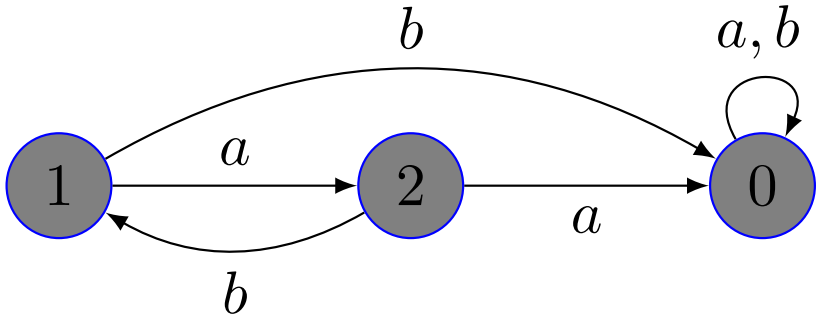


An A-automaton beyond $\mathbf{DS}$

B_2^1 is the transition monoid of

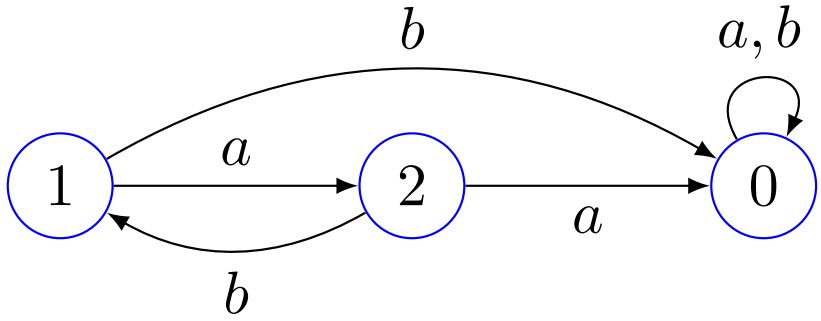


Alice wins on this automaton.

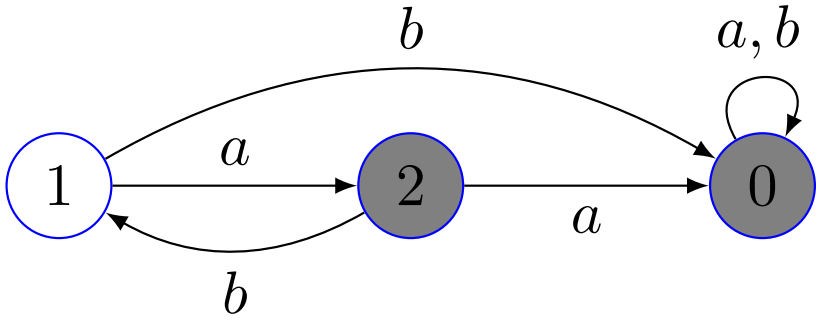


An A-automaton beyond $\mathbf{DS}$

B_2^1 is the transition monoid of



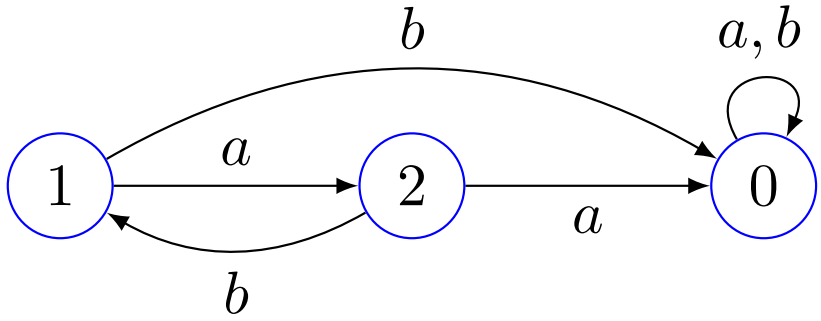
Alice wins on this automaton.



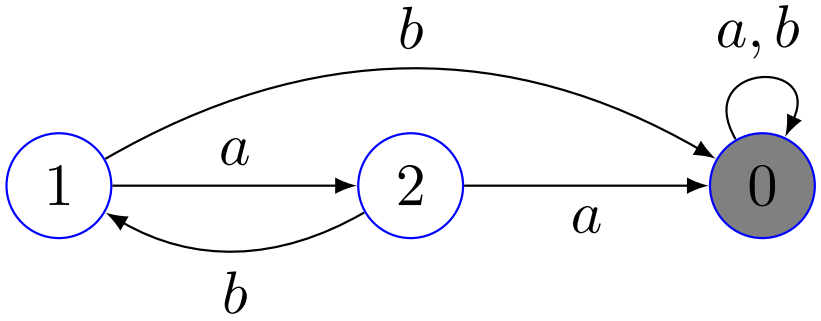
She can start by choosing a.

An A-automaton beyond DS

B_2^1 is the transition monoid of



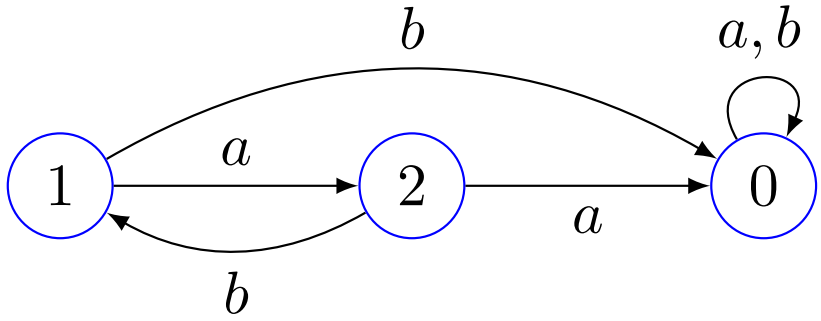
Alice wins on this automaton.



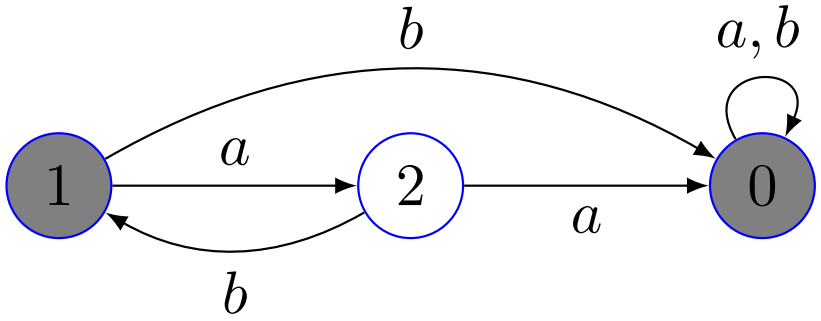
She can start by choosing a. If Bob responds by skipping his turn or by choosing a word that ends with a, Alice wins by choosing a.

An A-automaton beyond $\mathbf{DS}$

B_2^1 is the transition monoid of



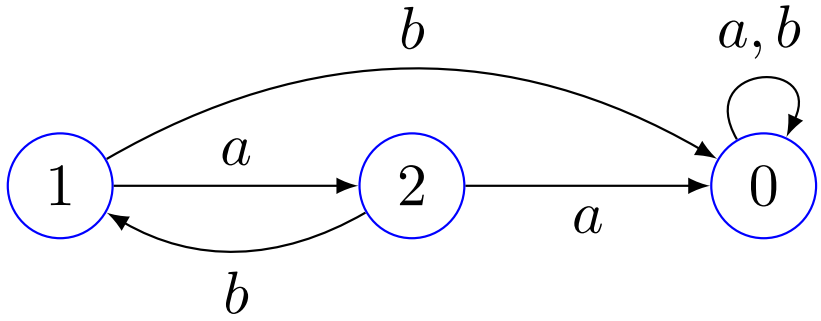
Alice wins on this automaton.



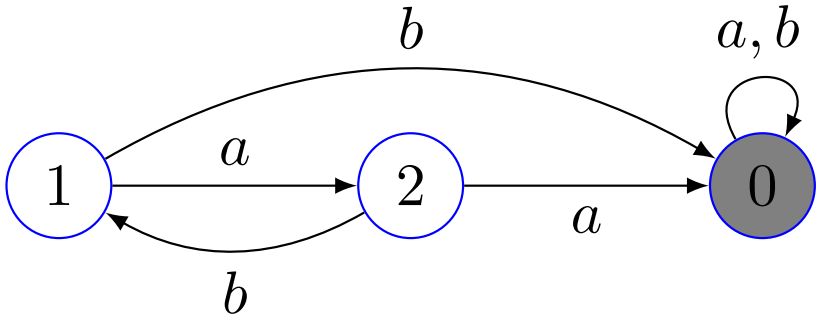
She can start by choosing a. If Bob responds by skipping his turn or by choosing a word that ends with a, Alice wins by choosing a. If Bob responds by choosing some word that ends with b

An A-automaton beyond $\mathbf{DS}$

B_2^1 is the transition monoid of



Alice wins on this automaton.



She can start by choosing a. If Bob responds by skipping his turn or by choosing a word that ends with a, Alice wins by choosing a. If Bob responds by choosing some word that ends with b, then Alice wins by choosing b.

Restricting Bob's capacity

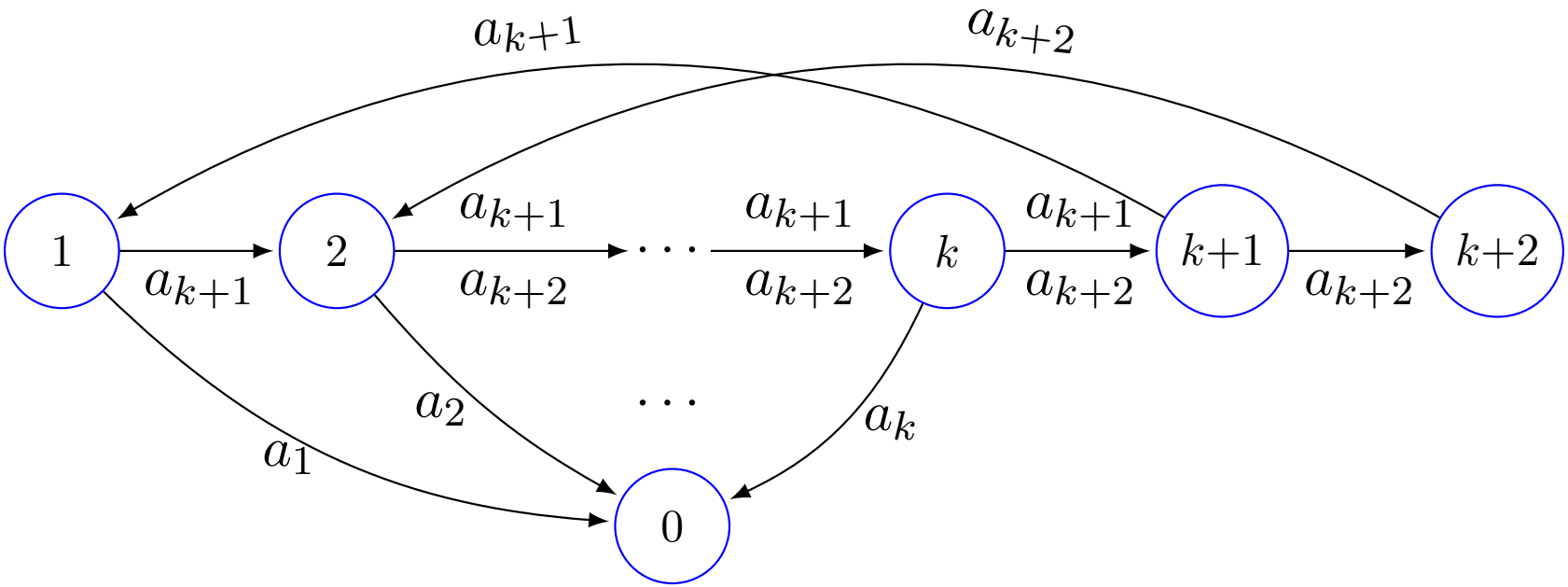
Modify the rules such that Bob is allowed to make less than $k \in \mathbb{Z}_+$ moves in response to each of Alice's moves.

Restricting Bob's capacity

Modify the rules such that Bob is allowed to make less than $k \in \mathbb{Z}_+$ moves in response to each of Alice's moves.
Here, for each k, we have a series of DFAs, on which Alice can win such a modified game, with reset thresholds being a quadratic function of the number of states, with leading coefficient $\frac{1}{2k}$.

Restricting Bob's capacity

Modify the rules such that Bob is allowed to make less than $k \in \mathbb{Z}_+$ moves in response to each of Alice's moves.
Here, for each k, we have a series of DFAs, on which Alice can win such a modified game, with reset thresholds being a quadratic function of the number of states, with leading coefficient $\frac{1}{2k}$.



Enhancing Alice's capacity

What happens if Alice, like Bob, is allowed to play words rather than single letter? If Alice may play arbitrary words, the game trivializes as Alice instantly wins on any synchronizing automaton by spelling out a reset word. The variant in which Alice's moves restricted to words of length at most $m \in \mathbb{Z}_+$ is more interesting.

Enhancing Alice's capacity

What happens if Alice, like Bob, is allowed to play words rather than single letter? If Alice may play arbitrary words, the game trivializes as Alice instantly wins on any synchronizing automaton by spelling out a reset word. The variant in which Alice's moves restricted to words of length at most $m \in \mathbb{Z}_+$ is more interesting.

Proposition

The reset threshold of an n-state DFA on which Alice wins by playing words of length at most m does not exceed $m(n-2)+1$.

Enhancing Alice's capacity

What happens if Alice, like Bob, is allowed to play words rather than single letter? If Alice may play arbitrary words, the game trivializes as Alice instantly wins on any synchronizing automaton by spelling out a reset word. The variant in which Alice's moves restricted to words of length at most $m \in \mathbb{Z}_+$ is more interesting.

Proposition

The reset threshold of an n-state DFA on which Alice wins by playing words of length at most m does not exceed $m(n-2)+1$.

We know that the bound is tight for $m = 1$ and $m = 2$.

Enhancing Alice's capacity

What happens if Alice, like Bob, is allowed to play words rather than single letter? If Alice may play arbitrary words, the game trivializes as Alice instantly wins on any synchronizing automaton by spelling out a reset word. The variant in which Alice's moves restricted to words of length at most $m \in \mathbb{Z}_+$ is more interesting.

Proposition

The reset threshold of an n-state DFA on which Alice wins by playing words of length at most m does not exceed $m(n-2)+1$.

We know that the bound is tight for $m = 1$ and $m = 2$.

Corollary

The Černý conjecture holds

Enhancing Alice's capacity

What happens if Alice, like Bob, is allowed to play words rather than single letter? If Alice may play arbitrary words, the game trivializes as Alice instantly wins on any synchronizing automaton by spelling out a reset word. The variant in which Alice's moves restricted to words of length at most $m \in \mathbb{Z}_+$ is more interesting.

Proposition

The reset threshold of an n-state DFA on which Alice wins by playing words of length at most m does not exceed $m(n-2)+1$.

We know that the bound is tight for $m = 1$ and $m = 2$.

Corollary

The Černý conjecture holds for n-state DFAs on which Alice wins by playing words of length at most n.

Algorithmic issues

Theorem

There exists an algorithm that, given a DFA $\mathscr{A} = \langle Q, \Sigma) \rangle$, decides the winner in the synchronization game on $\mathscr{A}$ in $O(|Q|^2 \cdot |\Sigma|)$ time.

Algorithmic issues

Theorem

There exists an algorithm that, given a DFA $\mathscr{A} = \langle Q, \Sigma) \rangle$, decides the winner in the synchronization game on $\mathscr{A}$ in $O(|Q|^2 \cdot |\Sigma|)$ time.

The proof relies on the well-known construction of the 2-subset automaton. Given a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$, define the action of the letters $a \in \Sigma$ on the set of all 2-element subsets of Q, augmented with a fresh symbol $\mathbf{s}$, by:

$$\mathbf{s}a := \mathbf{s} \quad \text{and} \quad \{q, q'\}a := \begin{cases} \{qa,\, q'a\} & \text{if } qa \neq q'a, \\ \mathbf{s} & \text{otherwise.} \end{cases}$$

This defines a DFA $\mathscr{A}^{[2]}$, in which $\mathbf{s}$ is a sink.

Algorithmic issues

Theorem

There exists an algorithm that, given a DFA $\mathscr{A} = \langle Q, \Sigma) \rangle$, decides the winner in the synchronization game on $\mathscr{A}$ in $O(|Q|^2 \cdot |\Sigma|)$ time.

The proof relies on the well-known construction of the 2-subset automaton. Given a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$, define the action of the letters $a \in \Sigma$ on the set of all 2-element subsets of Q, augmented with a fresh symbol $\mathbf{s}$, by:

$$\mathbf{s}a := \mathbf{s} \quad \text{and} \quad \{q, q'\}a := \begin{cases} \{qa, q'a\} & \text{if } qa \neq q'a, \\ \mathbf{s} & \text{otherwise.} \end{cases}$$

This defines a DFA $\mathscr{A}^{[2]}$, in which $\mathbf{s}$ is a sink.

The DFAs $\mathscr{A}$ and $\mathscr{A}^{[2]}$ have the same reset words. From this, it is easy to see that the DFAs are equivalent with respect to adversarial synchronization.

Algorithmic issues, continued

A-automata with a sink admit a transparent characterization:

Proposition

A DFA $\langle Q, \Sigma \rangle$ with a sink s is an A-automaton iff for every $q \neq \mathrm{s}$ there exists a letter $a \in \Sigma$ such that the states q and qa belong to distinct strongly connected component of $\langle Q, \Sigma \rangle$.

Algorithmic issues, continued

A-automata with a sink admit a transparent characterization:

Proposition

A DFA $\langle Q, \Sigma \rangle$ with a sink $\mathbf{s}$ is an A-automaton iff for every $q \neq \mathbf{s}$ there exists a letter $a \in \Sigma$ such that the states q and qa belong to distinct strongly connected component of $\langle Q, \Sigma \rangle$.

This readily gives an $O(|Q| \cdot |\Sigma|)$ time algorithm that, given a DFA $\langle Q, \Sigma \rangle$ with a sink, decides who wins the game on $\langle Q, \Sigma \rangle$.

Algorithmic issues, continued

A-automata with a sink admit a transparent characterization:

Proposition

A DFA $\langle Q, \Sigma \rangle$ with a sink s is an A-automaton iff for every $q \neq \mathrm{s}$ there exists a letter $a \in \Sigma$ such that the states q and qa belong to distinct strongly connected component of $\langle Q, \Sigma \rangle$.

This readily gives an $O(|Q| \cdot |\Sigma|)$ time algorithm that, given a DFA $\langle Q, \Sigma \rangle$ with a sink, decides who wins the game on $\langle Q, \Sigma \rangle$.

Applying this algorithm to the 2-subset automaton of a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ decides the winner in the synchronization game on $\mathscr{A}$ in $O(|Q|^2 \cdot |\Sigma|)$ time.

Algorithmic issues, continued

A-automata with a sink admit a transparent characterization:

Proposition

A DFA $\langle Q, \Sigma \rangle$ with a sink s is an A-automaton iff for every $q \neq \mathrm{s}$ there exists a letter $a \in \Sigma$ such that the states q and qa belong to distinct strongly connected component of $\langle Q, \Sigma \rangle$.

This readily gives an $O(|Q| \cdot |\Sigma|)$ time algorithm that, given a DFA $\langle Q, \Sigma \rangle$ with a sink, decides who wins the game on $\langle Q, \Sigma \rangle$.

Applying this algorithm to the 2-subset automaton of a DFA $\mathscr{A} = \langle Q, \Sigma \rangle$ decides the winner in the synchronization game on $\mathscr{A}$ in $O(|Q|^2 \cdot |\Sigma|)$ time.

For the version with Bob's moves restricted to words of length less than $k \in \mathbb{Z}_+$, a polynomial time algorithm also exists but it and its justification are much more involved.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.
- Adversarial synchronization gives a model in the form of a game.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.
- Adversarial synchronization gives a model in the form of a game.
- Although the rules of the game favor the adversary, we have identified a broad class of robustly synchronizable automata that can be reliably controlled in the face of resistance.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.
- Adversarial synchronization gives a model in the form of a game.
- Although the rules of the game favor the adversary, we have identified a broad class of robustly synchronizable automata that can be reliably controlled in the face of resistance.
- Robustly synchronizable automata admit reset words with length less than the number of states.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.
- Adversarial synchronization gives a model in the form of a game.
- Although the rules of the game favor the adversary, we have identified a broad class of robustly synchronizable automata that can be reliably controlled in the face of resistance.
- Robustly synchronizable automata admit reset words with length less than the number of states.
- Robust synchronizability can be efficiently decided.

Conclusion

- We aim to model reliably controllable systems that are required to perform tasks in the presence of natural or artificial interference.
- Adversarial synchronization gives a model in the form of a game.
- Although the rules of the game favor the adversary, we have identified a broad class of robustly synchronizable automata that can be reliably controlled in the face of resistance.
- Robustly synchronizable automata admit reset words with length less than the number of states.
- Robust synchronizability can be efficiently decided.
- Thank you for your attention!