A Quest for Short Identities

Which questions does automata theory ask algebra over and over again (but gets no answers so far)?

Mikhail Volkov

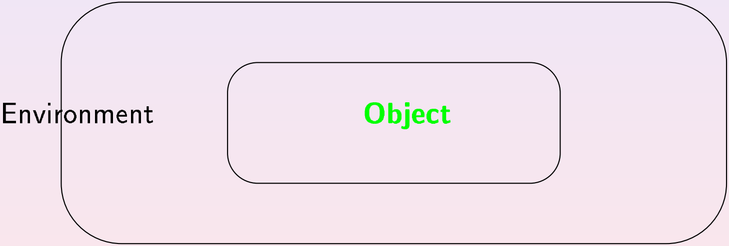
Ural Federal University, Ekaterinburg, Russia

AAA84, Dresden, June 8, 2012

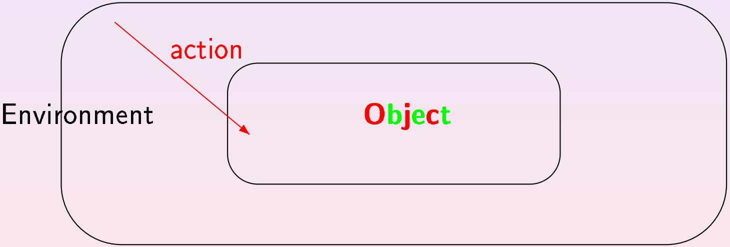
Finite Automata

A finite automaton is a very simple but extremely productive concept that captures the idea of an object interacting with an environment.

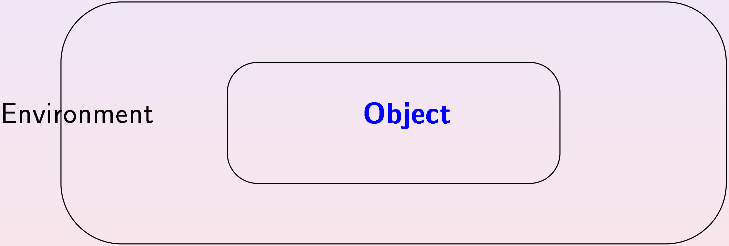
A finite automaton is a very simple but extremely productive concept that captures the idea of an object interacting with an environment.



A finite automaton is a very simple but extremely productive concept that captures the idea of an object interacting with an environment.



A finite automaton is a very simple but extremely productive concept that captures the idea of an object interacting with an environment.



This notion originates in the seminal work by Alan Turing ("On Computable Numbers, With an Application to the Entscheidungsproblem", Proc. London Math. Soc., Ser. 2, 42 (1936), 230–265).

"The behavior of the computer at any moment is determined by the symbols which he is observing, and his state of mind at that moment".

Another important source is the work by neurobiologists Warren McCulloch and Walter Pitts ("A Logical Calculus of the Ideas Immanent in Nervous Activity", Bull. Math. Biophys. 5 (1943), 115–133).

Finite Automata

This notion originates in the seminal work by Alan Turing ("On Computable Numbers, With an Application to the Entscheidungsproblem", Proc. London Math. Soc., Ser. 2, 42 (1936), 230–265).

"The behavior of the computer at any moment is determined by the symbols which he is observing, and his state of mind at that moment".

Another important source is the work by neurobiologists Warren McCulloch and Walter Pitts ("A Logical Calculus of the Ideas Immanent in Nervous Activity", Bull. Math. Biophys. 5 (1943), 115–133).

Finite Automata

This notion originates in the seminal work by Alan Turing ("On Computable Numbers, With an Application to the Entscheidungsproblem", Proc. London Math. Soc., Ser. 2, 42 (1936), 230–265).

"The behavior of the computer at any moment is determined by the symbols which he is observing, and his state of mind at that moment".

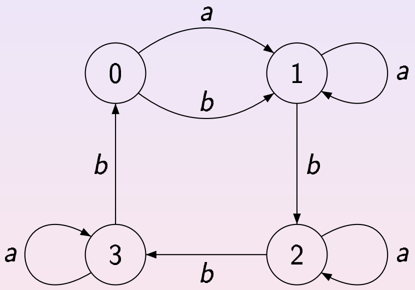
Another important source is the work by neurobiologists Warren McCulloch and Walter Pitts ("A Logical Calculus of the Ideas Immanent in Nervous Activity", Bull. Math. Biophys. 5 (1943), 115–133).

Visualization

Finite automata admit a convenient visual representation.

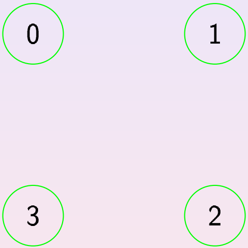
Visualization

Finite automata admit a convenient visual representation.



Visualization

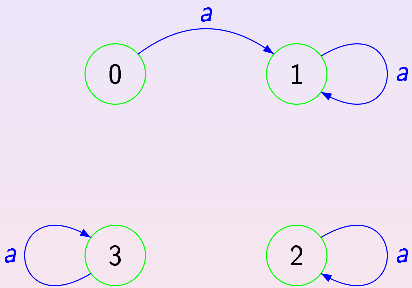
Finite automata admit a convenient visual representation.



Here one sees 4 **states** called 0,1,2,3,

Visualization

Finite automata admit a convenient visual representation.

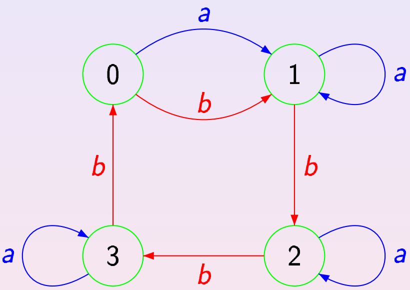


Here one sees 4 **states** called 0,1,2,3, an action called a

AAA84, Dresden, June 8, 2012

Visualization

Finite automata admit a convenient visual representation.



Here one sees 4 **states** called 0,1,2,3, an action called *a* and another action called *b*.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word.
The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.
To simplify notation we write $q \,.\, w$ for $\delta(q, w)$.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word.
The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.
To simplify notation we write $q \, . \, w$ for $\delta(q, w)$.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the finite state set;
- Σ is the input finite alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word.

The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.

To simplify notation we write $q \,.\, w$ for $\delta(q, w)$.

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word.
The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.
To simplify notation we write $q \,.\, w$ for $\delta(q, w)$.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the totally defined transition function.

Σ^* stands for the set of all words over Σ including the empty word.
The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.
To simplify notation we write $q \,.\, w$ for $\delta(q, w)$.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word. The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.

To simplify notation we write $q\,.\,w$ for $\delta(q, w)$.

Terminology and Notation

We consider complete deterministic finite automata (DFAs):

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

Here

- Q is the state set;
- Σ is the input alphabet;
- $\delta : Q \times \Sigma \to Q$ is the transition function.

Σ^* stands for the set of all words over Σ including the empty word.
The function δ uniquely extends to a function $Q \times \Sigma^* \to Q$ still denoted by δ.
To simplify notation we write $q\,.\,w$ for $\delta(q, w)$.

Synchronizing Automata

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q the word w was applied: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.

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

Other names:

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

A DFA $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q the word w was applied: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.

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

Other names:

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

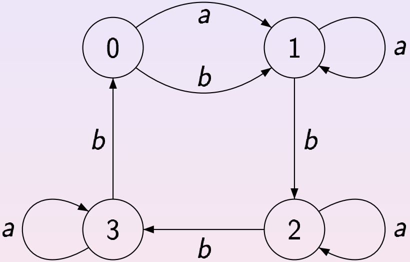
A DFA $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ is called synchronizing if there exists a word $w \in \Sigma^*$ whose action resets $\mathscr{A}$, that is, leaves $\mathscr{A}$ in one particular state no matter at which state in Q the word w was applied: $q \,.\, w = q' \,.\, w$ for all $q, q' \in Q$.

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

Other names:

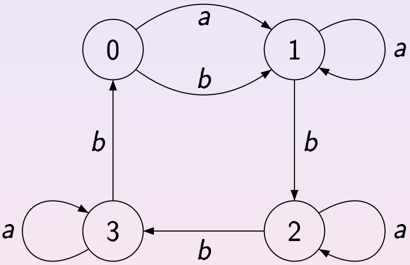
- for automata: directable, cofinal, collapsible, etc;
- for words: directing, recurrent, synchronizing, etc.

An Example



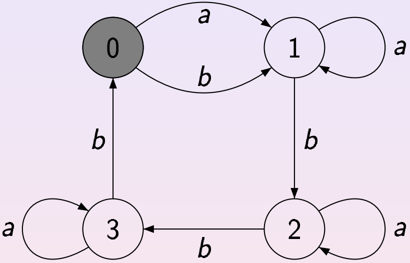
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



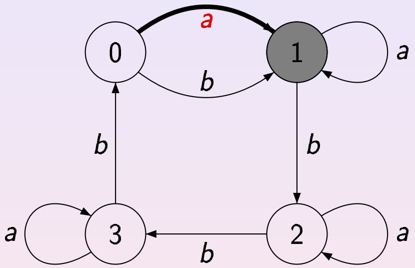
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



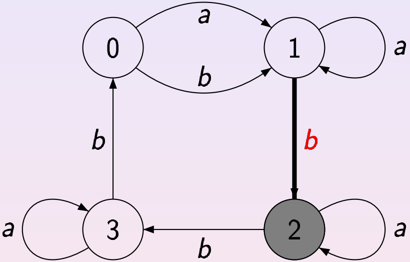
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



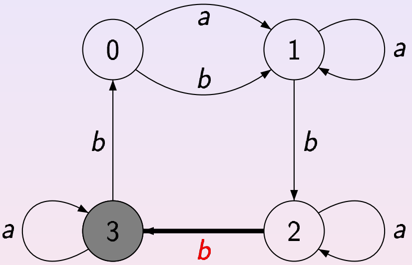
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



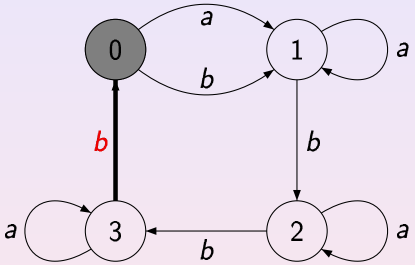
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



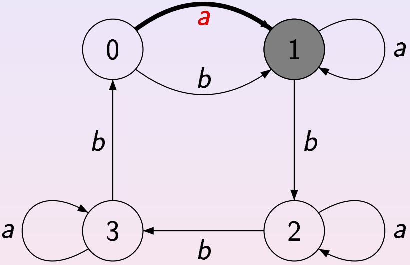
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



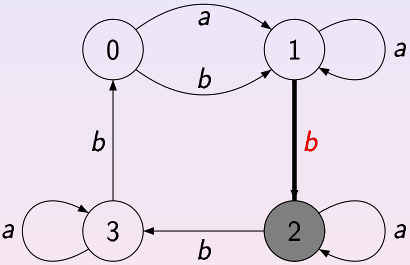
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



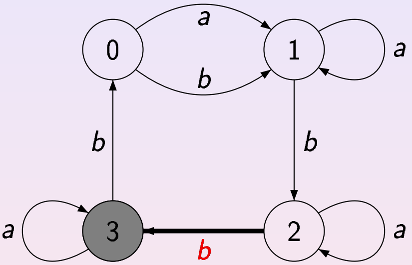
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



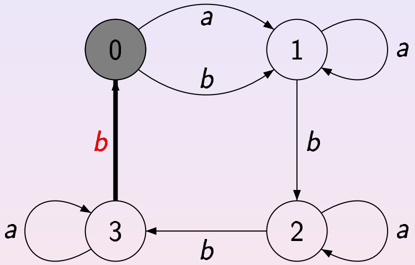
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



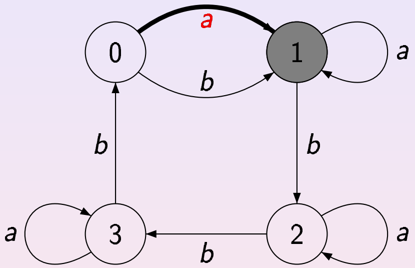
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



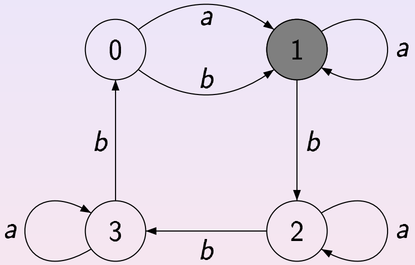
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



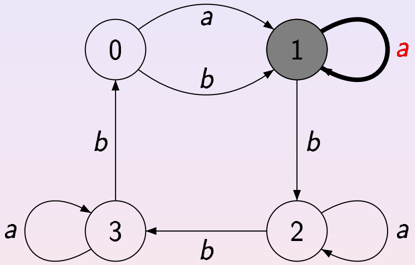
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



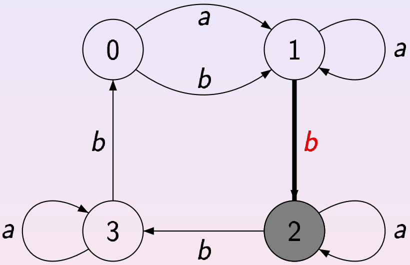
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



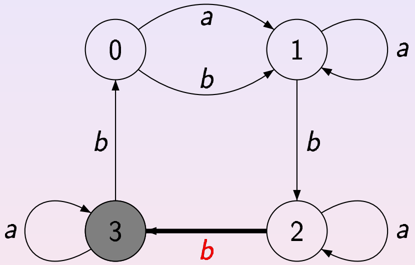
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



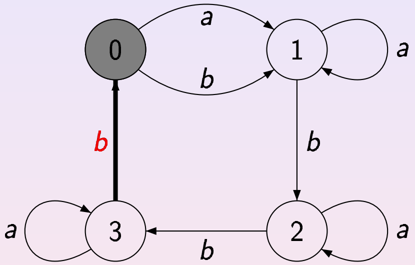
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



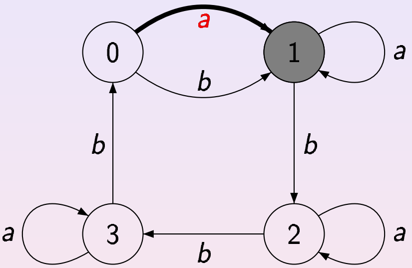
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



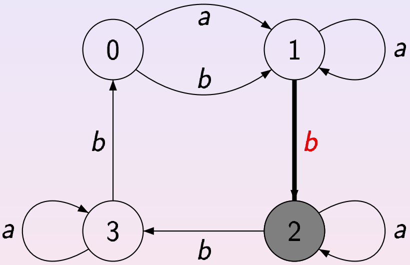
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



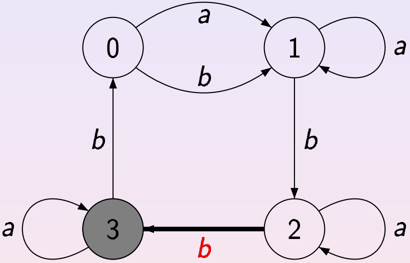
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



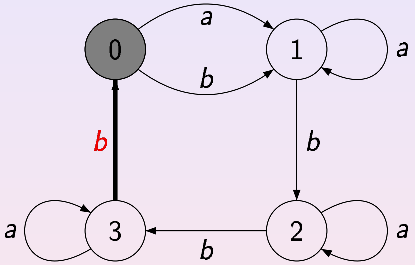
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



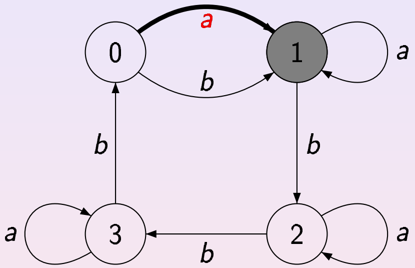
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



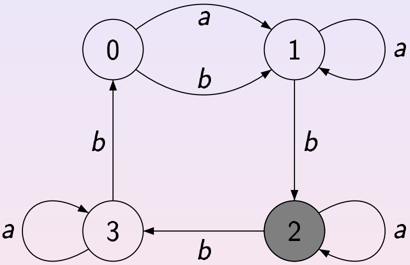
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



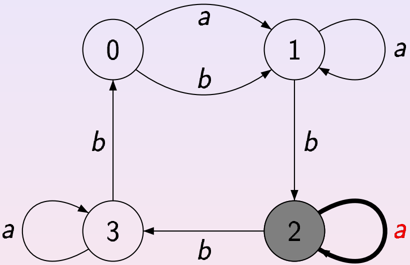
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



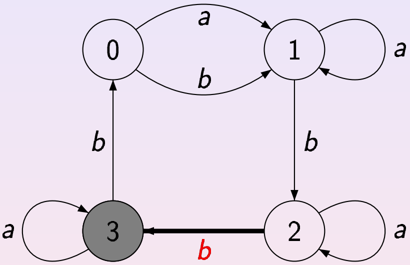
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



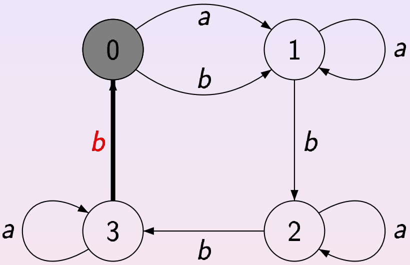
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



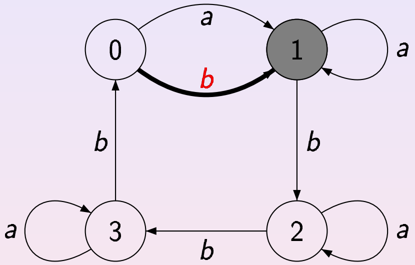
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



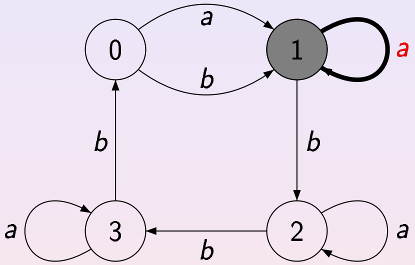
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



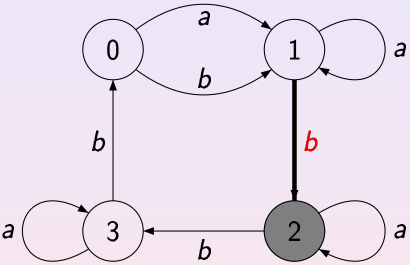
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



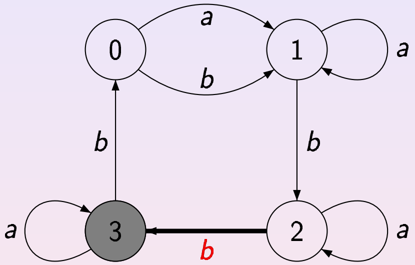
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



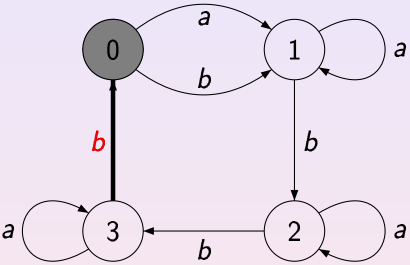
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



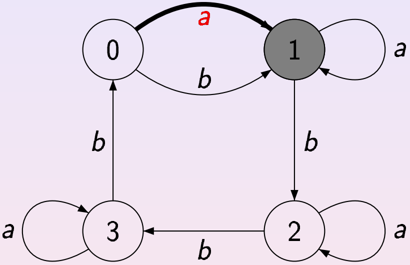
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



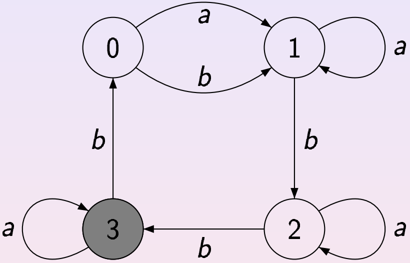
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



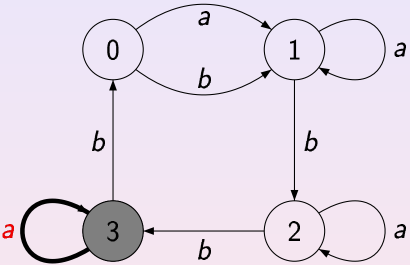
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



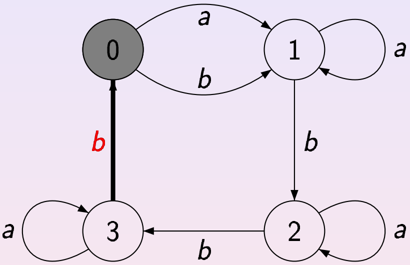
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



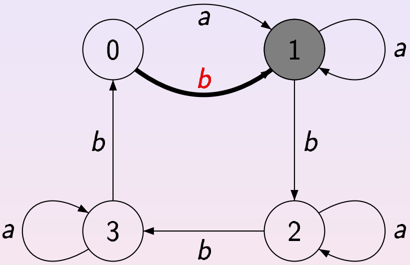
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



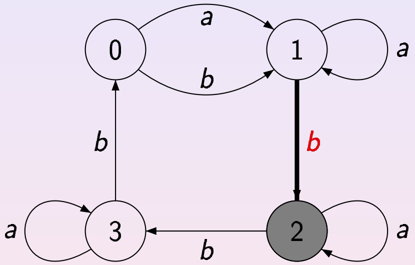
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



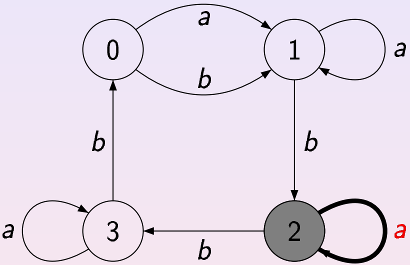
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



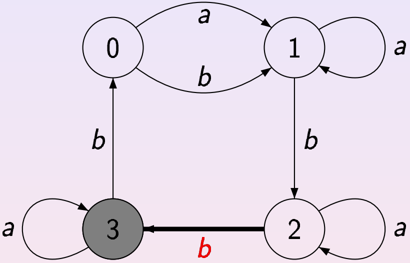
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



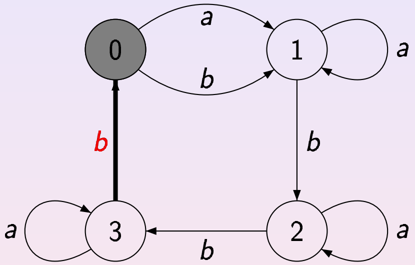
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



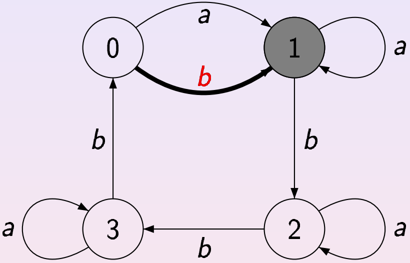
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



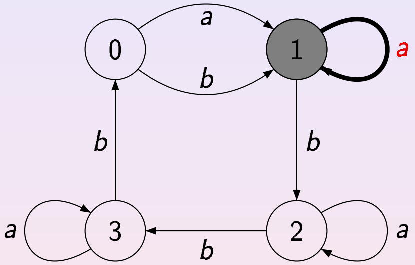
A reset word is *abbbabbba*: applying it at any state brings the automaton to the state 1.

An Example



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

An Example



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

Černý's Paper

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

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

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

Černý's Paper

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

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

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

Černý's Paper

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

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

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

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

- The notion was very natural by itself and fitted fairly well in what was considered as the mainstream of automata theory in the early 1960s.
- It also naturally arises in the framework of variable-length codes (such as Huffman codes), see, e.g., S. Even, “Test for synchronizability of finite automata and variable length codes”, IEEE Trans. Inform. Theory, 10 (1964), 185-189.
- Černý’s paper published in Slovak language remained unknown in the English-speaking world for quite a long time.

A Frequently Discovered Notion

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

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

• It also naturally arises in the framework of variable-length codes (such as Huffman codes), see, e.g., S. Even, "Test for synchronizability of finite automata and variable length codes", IEEE Trans. Inform. Theory, 10 (1964), 185-189.

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

A Frequently Discovered Notion

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

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

• It also naturally arises in the framework of variable-length codes (such as Huffman codes), see, e.g., S. Even, “Test for synchronizability of finite automata and variable length codes”, IEEE Trans. Inform. Theory, 10 (1964), 185-189.

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

A Frequently Discovered Notion

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

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

• It also naturally arises in the framework of variable-length codes (such as Huffman codes), see, e.g., S. Even, “Test for synchronizability of finite automata and variable length codes”, IEEE Trans. Inform. Theory, 10 (1964), 185-189.

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

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

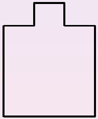
Suppose that one of the parts of a certain device has the following shape:

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

Re-inventing by Engineers

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

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

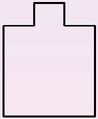


Such parts arrive at manufacturing sites in boxes and they need to be sorted and oriented before assembly.

Re-inventing by Engineers

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

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



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

Re-inventing by Engineers

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

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

Re-inventing by Engineers

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

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

Re-inventing by Engineers

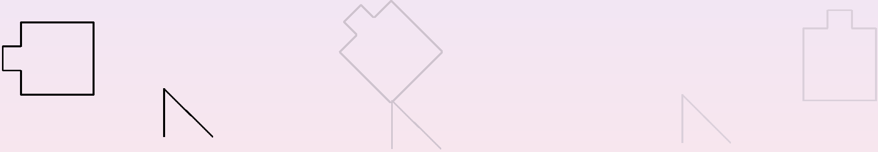
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.

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

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

Re-inventing by Engineers

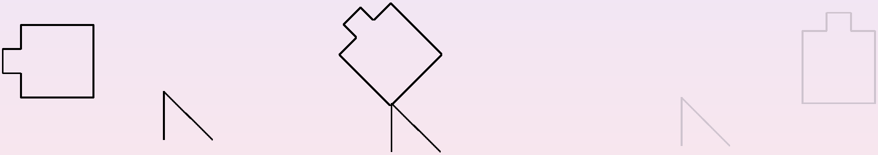
We put parts to be oriented on a conveyer belt which takes them to the assembly point and let the stream of the parts encounter a series of passive obstacles of two types (*high* and *low*) placed along the belt.
A high obstacle is high enough so that any part on the belt encounters this obstacle by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



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

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

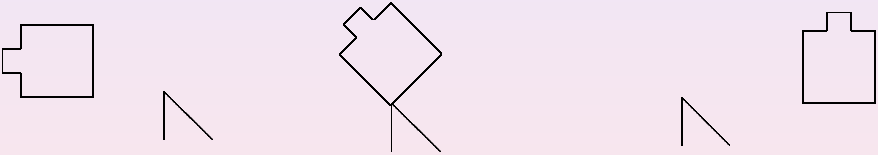
A high obstacle is high enough so that any part on the belt encounters this obstacle by its rightmost low angle.



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

Re-inventing by Engineers

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

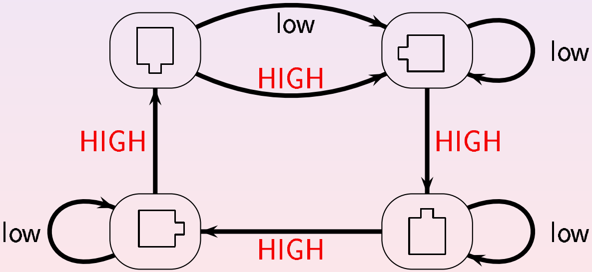


Being curried by the belt, the part then is forced to turn 90° clockwise.

Re-inventing by Engineers

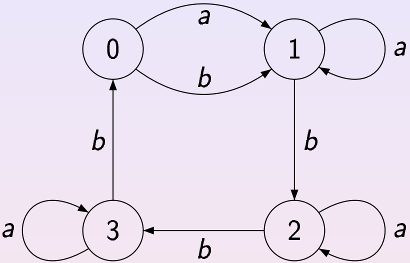
A low obstacle has the same effect whenever the part is in the "bump-down" orientation; otherwise it does not touch the part which therefore passes by without changing the orientation.

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



Re-inventing by Engineers

We met this picture a few slides ago:



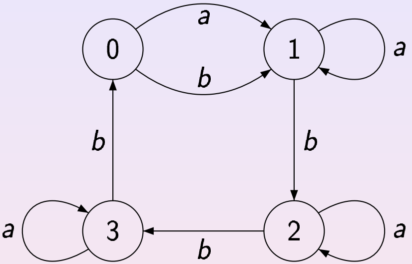
– this was our example of a synchronizing automaton, and we saw that *abbbabbba* is a reset sequence of actions. Hence the series of obstacles

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

yields the desired orienter.

Re-inventing by Engineers

We met this picture a few slides ago:



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

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

yields the desired orienter.

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

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

Re-inventing by Dynamics Theorists

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

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

Re-inventing by Dynamics Theorists

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

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

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

Re-inventing by Dynamics Theorists

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

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

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

Re-inventing by Dynamics Theorists

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

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

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

Re-inventing by Dynamics Theorists

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

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

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

Re-inventing by Dynamics Theorists

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

$$\begin{array}{ccccccccc}
0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 122020\mathbf{1}112202011 \\
1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 122020\mathbf{1}120111212 \\
2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 201112\mathbf{1}212201220
\end{array}$$

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

Re-inventing by Dynamics Theorists

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

$$\begin{array}{ccccccccc} 0 & \mapsto & 11 & \mapsto & 1212 & \mapsto & 12201220 & \mapsto & 122020\mathbf{1}112202011 \\ 1 & \mapsto & 12 & \mapsto & 1220 & \mapsto & 12202011 & \mapsto & 122020\mathbf{1}120111212 \\ 2 & \mapsto & 20 & \mapsto & 2011 & \mapsto & 20111212 & \mapsto & 201112\mathbf{1}212201220 \end{array}$$

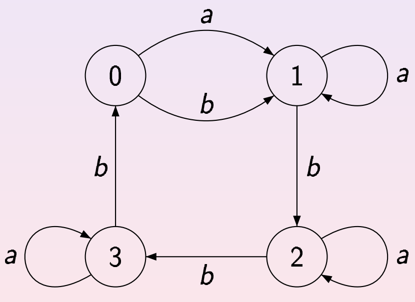
Thus, τ satisfies the coincidence condition (with $k = 4$, $m = 7$). The coincidence condition completely characterizes the constant length substitutions that give rise to dynamical systems measure-theoretically isomorphic to a translation on a compact Abelian group (Dekking, 1978).

Re-inventing by Dynamics Theorists

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

Re-inventing by Dynamics Theorists

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



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

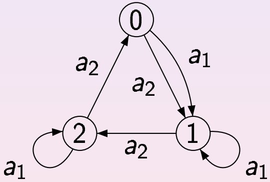
Re-inventing by Dynamics Theorists

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

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

Re-inventing by Dynamics Theorists

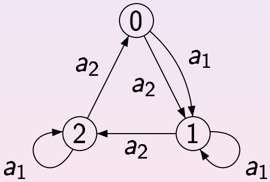
Conversely, each substitution $\sigma : X \to X^+$ such that all words $\sigma(x)$, $x \in X$, have the same length ℓ gives rise to a DFA for which X is the state set and which has ℓ input letters $a_1, \dots, a_\ell$ acting on X as follows: $x \,.\, a_i$ is the symbol in the i-th position of the word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



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

Re-inventing by Dynamics Theorists

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



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

Černý Conjecture

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

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

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

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

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

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

Černý Conjecture

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

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

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

Černý Conjecture

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

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

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

Černý Conjecture

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

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

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

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

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

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

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

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

Černý Conjecture

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

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

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

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

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x . w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x . u = x . v$ (homotypical identities) or $x . u = y . v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x . w = y . w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x . u = y . v$ say, then substituting y for x we get $y . u = y . v$ whence $x . u = y . u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x\,.\,w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x\,.\,u = x\,.\,v$ (homotypical identities) or $x\,.\,u = y\,.\,v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x\,.\,w = y\,.\,w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x\,.\,u = y\,.\,v$ say, then substituting y for x we get $y\,.\,u = y\,.\,v$ whence $x\,.\,u = y\,.\,u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

An Algebraic Viewpoint

One may treat DFAs as unary algebras since each letter of the input alphabet defines a unary operation on the state set.

Terms in the language of such unary algebras are expressions of the form $x \,.\, w$, where x is a variable and w is a word over an alphabet Σ. Identities of unary algebras can be of the form either $x \,.\, u = x \,.\, v$ (homotypical identities) or $x \,.\, u = y \,.\, v$ with $x \neq y$ (heterotypical identities).

Clearly, if $\mathscr{A}$ is a synchronizing automaton and w is its reset word, then $\mathscr{A}$ satisfies the identity $x \,.\, w = y \,.\, w$. Conversely, if $\mathscr{A}$ satisfies a heterotypical identity, $x \,.\, u = y \,.\, v$ say, then substituting y for x we get $y \,.\, u = y \,.\, v$ whence $x \,.\, u = y \,.\, u$. We conclude that u is a reset word for $\mathscr{A}$.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications . . .

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers – this reformulation is well known and it does not solve the problem and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications – e.g., the question of whether a given finite unary algebra satisfies an identity of a given length is NP-complete. . .

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications ...

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Algebraic Reformulation

Thus, synchronizing automata = unary algebras satisfying heterotypical identities.

The Černý conjecture is thus just the claim that if a unary algebra on an n-element base set satisfies a heterotypical identity, then the algebra satisfies such an identity with one of the terms involved of length at most $(n-1)^2$.

Disclaimers and applications . . .

One real thing to remember: the Černý conjecture is a question about short identities in a certain algebra.

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$.

How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al's* "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for a new use.

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$. (We know the state set and the input alphabet but we have no idea about the transition function.)

How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al*'s "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$. How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al*'s "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for a new use.

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$. How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al*'s "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for a new use.

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$. How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al*'s "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for a new use.

Black-Box Version

Consider now a black-box synchronizing automaton $\mathscr{A} = \langle Q, \Sigma, \mathbf{?} \rangle$. How to synchronize such an automaton?

Motivation: real computational devices are composites made from many finite automata, each a with relatively small number of states. We need an input signal which would simultaneously reset all those automata and which could be generated without analyzing the structure of each particular component of the device.

In particular, Yacob Benenson *et al*'s "*soup of automata*", see "Programmable and autonomous computing machine made of biomolecules", Nature 414 (2001), 430–434; "DNA molecule provides a computing machine with both data and fuel", Proc. National Acad. Sci. USA 100 (2003), 2191–2196, is a solution containing 3×10^{12} DNA-based automata per μl that work in parallel on different inputs (DNA strands). One has to feed the automata with a common reset word in order to get them ready for a new use.

Algebraic Applications

Universal reset words also admit various algebraic applications.

Reinhard Pöschel *et al* ("Identities in full transformation semigroups", Algebra Universalis 31 (1994), 580–588) used them to find identities separating the full transformation semigroup $\mathbb{T}_n$ from its proper subsemigroups;

Jorge Almeida and ∼ ("Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety", J. Algebra and its Applications 2 (2003), 137–163) applied them to construct idempotents in the least ideal of the free profinite semigroup.

Jorge Almeida, ∼, and Svetlana Goldberg ("Complexity of the identity checking problem in finite semigroups", J. Math. Sciences 158 (2009), 605–614) used them to relate the term equivalence problem for a given finite semigroup S to the analogous problem for the maximal subgroups of S.

Algebraic Applications

Universal reset words also admit various algebraic applications.

Reinhard Pöschel *et al* ("Identities in full transformation semigroups", Algebra Universalis 31 (1994), 580–588) used them to find identities separating the full transformation semigroup $\mathbb{T}_n$ from its proper subsemigroups;

Jorge Almeida and ∼ ("Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety", J. Algebra and its Applications 2 (2003), 137–163) applied them to construct idempotents in the least ideal of the free profinite semigroup.

Jorge Almeida, ∼, and Svetlana Goldberg ("Complexity of the identity checking problem in finite semigroups", J. Math. Sciences 158 (2009), 605–614) used them to relate the term equivalence problem for a given finite semigroup S to the analogous problem for the maximal subgroups of S.

Algebraic Applications

Universal reset words also admit various algebraic applications.

Reinhard Pöschel *et al* ("Identities in full transformation semigroups", Algebra Universalis 31 (1994), 580–588) used them to find identities separating the full transformation semigroup $\mathbb{T}_n$ from its proper subsemigroups;

Jorge Almeida and ∼ ("Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety", J. Algebra and its Applications 2 (2003), 137–163) applied them to construct idempotents in the least ideal of the free profinite semigroup.

Jorge Almeida, ∼, and Svetlana Goldberg ("Complexity of the identity checking problem in finite semigroups", J. Math. Sciences 158 (2009), 605–614) used them to relate the term equivalence problem for a given finite semigroup S to the analogous problem for the maximal subgroups of S.

Algebraic Applications

Universal reset words also admit various algebraic applications.

Reinhard Pöschel *et al* ("Identities in full transformation semigroups", Algebra Universalis 31 (1994), 580–588) used them to find identities separating the full transformation semigroup $\mathbb{T}_n$ from its proper subsemigroups;

Jorge Almeida and $\sim$ ("Profinite identities for finite semigroups whose subgroups belong to a given pseudovariety", J. Algebra and its Applications 2 (2003), 137–163) applied them to construct idempotents in the least ideal of the free profinite semigroup.

Jorge Almeida, $\sim$, and Svetlana Goldberg ("Complexity of the identity checking problem in finite semigroups", J. Math. Sciences 158 (2009), 605–614) used them to relate the term equivalence problem for a given finite semigroup S to the analogous problem for the maximal subgroups of S.

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

Constructing Universal Reset Words

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

Constructing Universal Reset Words

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

Constructing Universal Reset Words

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

Constructing Universal Reset Words

How to construct a universal reset word?

A brute force method relies on the fact that a synchronizing automaton with n states has a reset word of length at most $\frac{n^3-n}{6}$ (Pin-Frankl, 1983). Therefore, given a finite alphabet Σ, one can concatenate all words over Σ of length up to $\frac{n^3-n}{6}$ and get a word that resets all synchronizing automata with n states and input alphabet Σ. This idea is due to Masami Ito and Jürgen Duske "On cofinal and definite automata", Acta Cybernetica 6 (1983), 181–189.

If the Černý conjecture holds true, it suffices to concatenate all words over Σ of length up to $(n-1)^2$. An accurate concatenation (based on the DeBruijn graph) yields a universal reset word of length $|\Sigma|^{(n-1)^2} + n^2 - 2n$.

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and ∼ (“Words guaranteeing minimum image”, Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1} + n - 2$.

Define the *universal Černý function* $UC(t, n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1} + n - 2 \leq UC(t, n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and $\sim$ ("Words guaranteeing minimum image", Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1} + n - 2$.

Define the *universal Černý function* $UC(t, n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1} + n - 2 \leq UC(t, n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and ∼ ("Words guaranteeing minimum image", Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1} + n - 2$.

Define the *universal Černý function* $UC(t, n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1} + n - 2 \leq UC(t, n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and ∼ ("Words guaranteeing minimum image", Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1}+n-2$.

Define the *universal Černý function* $UC(t,n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1}+n-2 \leq UC(t,n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and ∼ ("Words guaranteeing minimum image", Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1} + n - 2$.

Define the *universal Černý function* $UC(t,n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1} + n - 2 \leq UC(t,n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Constructing Universal Reset Words

A somewhat surprising fact is that one can do much better: Stuart Margolis, Jean-Éric Pin and ∼ ("Words guaranteeing minimum image", Int. J. Foundations Comp. Sci. 15 (2004), 259–276) found a word that resets all synchronizing automata with n states and input alphabet Σ and has length $O(|\Sigma|^{\frac{n^2-n}{2}})$. This construction does not depend on the Černý conjecture.

On the other hand, it is proved that such a word cannot have length less than $|\Sigma|^{n-1}+n-2$.

Define the *universal Černý function* $UC(t,n)$ as the minimum length of a word that resets all synchronizing automata with n states and t input letters. In terms of this function, our current knowledge can be summarized in the following line:

$$t^{n-1}+n-2 \leq UC(t,n) \leq t^{\frac{n^2-n}{2}} + o(t^{\frac{n^2-n}{2}}).$$

Algebraic Reformulation

Again, it should be clear that universal reset words correspond to heterotypical identities that hold in every unary algebra with the given size of the base set.

Therefore, the problem of evaluating the universal Černý function $UC(t, n)$ is nothing but the problem of finding a heterotypical identity of minimum length which holds in all n-element algebras with t unary operations that satisfy a heterotypical identity.

Again, we see that a problem of automata theory becomes a question about short identities in a certain algebra.

Algebraic Reformulation

Again, it should be clear that universal reset words correspond to heterotypical identities that hold in every unary algebra with the given size of the base set.

Therefore, the problem of evaluating the universal Černý function $UC(t, n)$ is nothing but the problem of finding a heterotypical identity of minimum length which holds in all n-element algebras with t unary operations that satisfy a heterotypical identity.

Again, we see that a problem of automata theory becomes a question about short identities in a certain algebra.

Algebraic Reformulation

Again, it should be clear that universal reset words correspond to heterotypical identities that hold in every unary algebra with the given size of the base set.
Therefore, the problem of evaluating the universal Černý function $UC(t, n)$ is nothing but the problem of finding a heterotypical identity of minimum length which holds in all n-element algebras with t unary operations that satisfy a heterotypical identity.

Again, we see that a problem of automata theory becomes a question about short identities in a certain algebra.

Separating Words by Automata

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA in which we fix an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. We say that $\mathscr{A}$ accepts a word $w \in \Sigma^*$ if $q_0 \, . \, w \in F$, that is, the path starting at q_0 and labeled w ends at a state in F. Otherwise $\mathscr{A}$ rejects w.

For instance, the above automaton accepts the word *aabb* but rejects the word *bbaa*.

Separating Words by Automata

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA in which we fix an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. We say that $\mathscr{A}$ accepts a word $w \in \Sigma^*$ if $q_0 \,.\, w \in F$, that is, the path starting at q_0 and labeled w ends at a state in F. Otherwise $\mathscr{A}$ rejects w.

For instance, the above automaton accepts the word *aabb* but rejects the word *bbaa*.

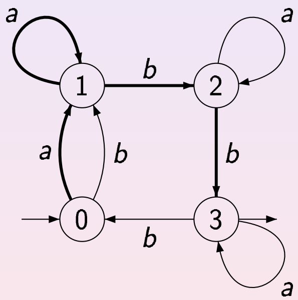
Separating Words by Automata

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA in which we fix an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. We say that $\mathscr{A}$ accepts a word $w \in \Sigma^*$ if $q_0 \, . \, w \in F$, that is, the path starting at q_0 and labeled w ends at a state in F. Otherwise $\mathscr{A}$ rejects w.

For instance, the above automaton accepts the word *aabb* but rejects the word *bbaa*.

Separating Words by Automata

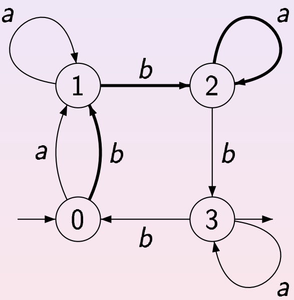
Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA in which we fix an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. We say that $\mathscr{A}$ accepts a word $w \in \Sigma^*$ if $q_0 \, . \, w \in F$, that is, the path starting at q_0 and labeled w ends at a state in F. Otherwise $\mathscr{A}$ rejects w.



For instance, the above automaton accepts the word *aabb* but rejects the word *bbaa*.

Separating Words by Automata

Let $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ be a DFA in which we fix an initial state $q_0 \in Q$ and a set of final states $F \subseteq Q$. We say that $\mathscr{A}$ accepts a word $w \in \Sigma^*$ if $q_0 \,.\, w \in F$, that is, the path starting at q_0 and labeled w ends at a state in F. Otherwise $\mathscr{A}$ rejects w.



For instance, the above automaton accepts the word *aabb* but rejects the word *bbaa*.

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\operatorname{sep}(u, v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\operatorname{sep}(u, v) = \operatorname{sep}(v, u)$.

Let $S(n) = \max \operatorname{sep}(u, v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\mathrm{sep}(u, v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\mathrm{sep}(u, v) = \mathrm{sep}(v, u)$.

Let $S(n) = \max \mathrm{sep}(u, v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\text{sep}(u, v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\text{sep}(u, v) = \text{sep}(v, u)$.

Let $S(n) = \max \text{sep}(u, v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\mathrm{sep}(u,v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\mathrm{sep}(u,v) = \mathrm{sep}(v,u)$.

Let $S(n) = \max \mathrm{sep}(u,v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\mathrm{sep}(u,v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\mathrm{sep}(u,v) = \mathrm{sep}(v,u)$.

Let $S(n) = \max \mathrm{sep}(u,v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Separating Words by Automata

We say that a DFA $\mathscr{A}$ separates words u and v if $\mathscr{A}$ accepts one but rejects the other. Given two distinct words u, v we let $\mathrm{sep}(u, v)$ be the number of states in the smallest DFA accepting u and rejecting v. Observe that $\mathrm{sep}(u, v) = \mathrm{sep}(v, u)$.

Let $S(n) = \max \mathrm{sep}(u, v)$ where u and v are distinct words of length at most n. The Separating Words Problem is to determine good upper and lower bounds on $S(n)$. It was introduced by Pawel Goralčik and Vačlav Koubek ("On discerning words by automata", Lect. Notes Comput. Sci. 226 (1986), 116–122), who proved

$$S(n) = o(n).$$

The best upper bound so far is due to Robson ("Separating words with machines and groups", RAIRO Inform. Théor. App., 30 (1996), 81–86), who obtained

$$S(n) = O(n^{2/5}(\log n)^{3/5}).$$

Algebraic Viewpoint

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\mathrm{lcm}(1,2,\dots,k)}.$$

It is known that $\mathrm{lcm}(1, 2, \dots, k)$ grows faster than 2^k so that k is logarithmic in $\mathrm{lcm}(1, 2, \dots, k)$.

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\mathrm{lcm}(1,2,\dots,k)}.$$

It is known that $\mathrm{lcm}(1, 2, \dots, k)$ grows faster than 2^k so that k is logarithmic in $\mathrm{lcm}(1, 2, \dots, k)$.

Algebraic Viewpoint

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\mathrm{lcm}(1,2,\dots,k)}.$$

It is known that $\mathrm{lcm}(1,2,\dots,k)$ grows faster than 2^k so that k is logarithmic in $\mathrm{lcm}(1,2,\dots,k)$.

Algebraic Viewpoint

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\text{lcm}(1,2,\dots,k)}.$$

It is known that $\text{lcm}(1, 2, \dots, k)$ grows faster than 2^k so that k is logarithmic in $\text{lcm}(1, 2, \dots, k)$.

Algebraic Viewpoint

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\mathrm{lcm}(1,2,\dots,k)}.$$

It is known that $\mathrm{lcm}(1, 2, \dots, k)$ grows faster than 2^k so that k is logarithmic in $\mathrm{lcm}(1, 2, \dots, k)$.

Algebraic Viewpoint

Observe that if distinct words u, v are such that the identity $u = v$ holds true in the full transformation semigroup $\mathbb{T}_k$, then u and v cannot be separated by any automaton $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ with at most k states. Indeed, the transition semigroup of $\mathscr{A}$, that is, the semigroup of transformations of the set Q induced by the words in Σ^*, embeds in $\mathbb{T}_k$ whence u and v act the same on Q and are simultaneously accepted or rejected.

Hence short identities in $\mathbb{T}_k$ may be used to produce lower bounds for the Separating Words Problem.

All known lower bounds for $S(n)$ are of magnitude $\Omega(\log n)$. They correspond to the following one-letter identity of $\mathbb{T}_k$:

$$x^{k-1} = x^{k-1+\mathrm{lcm}(1,2,\dots,k)}.$$

It is known that $\mathrm{lcm}(1, 2, \dots, k)$ grows faster than 2^k so that k is logarithmic in $\mathrm{lcm}(1, 2, \dots, k)$.

A similar problem concerns separation of words by permutation automata (DFAs in which each letter acts as a permutation of the state set). Here the best upper bound so far is $O(n^{\frac{1}{2}})$ – this means that every two distinct words of length at most n can be separated by a permutation automaton with $O(n^{\frac{1}{2}})$ states – Robson, loc. cit. For a lower bound, one needs short "positive" identities in the symmetric group $\mathbb{S}_k$.

Again, there is a one-letter identity of length $\mathrm{lcm}(1, 2, \ldots, k)$ which is exponential of k. However, at least for some k there are shorter identities, for instance, the identity

$$xxyxxyyyyy = yyyyyxxyxx$$

of length 11 holds in $\mathbb{S}_4$ while $\mathrm{lcm}(1, 2, 3, 4) = 12$.

A similar problem concerns separation of words by permutation automata (DFAs in which each letter acts as a permutation of the state set). Here the best upper bound so far is $O(n^{\frac{1}{2}})$ – this means that every two distinct words of length at most n can be separated by a permutation automaton with $O(n^{\frac{1}{2}})$ states – Robson, loc. cit. For a lower bound, one needs short "positive" identities in the symmetric group $\mathbb{S}_k$.

Again, there is a one-letter identity of length $\mathrm{lcm}(1, 2, \ldots, k)$ which is exponential of k. However, at least for some k there are shorter identities, for instance, the identity

$$xxyxxyyyyy = yyyyyxxyxx$$

of length 11 holds in $\mathbb{S}_4$ while $\mathrm{lcm}(1, 2, 3, 4) = 12$.

Identities in Symmetric Groups

A similar problem concerns separation of words by permutation automata (DFAs in which each letter acts as a permutation of the state set). Here the best upper bound so far is $O(n^{\frac{1}{2}})$ – this means that every two distinct words of length at most n can be separated by a permutation automaton with $O(n^{\frac{1}{2}})$ states – Robson, loc. cit. For a lower bound, one needs short "positive" identities in the symmetric group $\mathbb{S}_k$.

Again, there is a one-letter identity of length $\mathrm{lcm}(1, 2, \ldots, k)$ which is exponential of k. However, at least for some k there are shorter identities, for instance, the identity

$$xxyxxyyyyy = yyyyyxxyxx$$

of length 11 holds in $\mathbb{S}_4$ while $\mathrm{lcm}(1, 2, 3, 4) = 12$.

Identities in Symmetric Groups

A similar problem concerns separation of words by permutation automata (DFAs in which each letter acts as a permutation of the state set). Here the best upper bound so far is $O(n^{\frac{1}{2}})$ – this means that every two distinct words of length at most n can be separated by a permutation automaton with $O(n^{\frac{1}{2}})$ states – Robson, loc. cit. For a lower bound, one needs short “positive” identities in the symmetric group $\mathbb{S}_k$.

Again, there is a one-letter identity of length $\mathrm{lcm}(1, 2, \ldots, k)$ which is exponential of k. However, at least for some k there are shorter identities, for instance, the identity

$$xyxxyyyyy = yyyyyxxyxx$$

of length 11 holds in $\mathbb{S}_4$ while $\mathrm{lcm}(1, 2, 3, 4) = 12$.

Identities in Symmetric Groups

A similar problem concerns separation of words by permutation automata (DFAs in which each letter acts as a permutation of the state set). Here the best upper bound so far is $O(n^{\frac{1}{2}})$ – this means that every two distinct words of length at most n can be separated by a permutation automaton with $O(n^{\frac{1}{2}})$ states – Robson, loc. cit. For a lower bound, one needs short "positive" identities in the symmetric group $\mathbb{S}_k$.

Again, there is a one-letter identity of length $\mathrm{lcm}(1, 2, \ldots, k)$ which is exponential of k. However, at least for some k there are shorter identities, for instance, the identity

$$xxyxxyyyyy = yyyyyxxyxx$$

of length 11 holds in $\mathbb{S}_4$ while $\mathrm{lcm}(1, 2, 3, 4) = 12$.

Conclusion

Identities have been in focus of general algebra since its early days however it seems that algebraists (including myself) do not really care about the size of identities.

Now we see that several popular and apparently hard questions in the theory of finite automata amount to ask for short identities in certain algebras.

I hope this fact may be interesting for algebraists and should stimulate a systematic study of shortest identities in various algebraic structures.

Conclusion

Identities have been in focus of general algebra since its early days however it seems that algebraists (including myself) do not really care about the size of identities.

Now we see that several popular and apparently hard questions in the theory of finite automata amount to ask for short identities in certain algebras.

I hope this fact may be interesting for algebraists and should stimulate a systematic study of shortest identities in various algebraic structures.

Conclusion

Identities have been in focus of general algebra since its early days however it seems that algebraists (including myself) do not really care about the size of identities.

Now we see that several popular and apparently hard questions in the theory of finite automata amount to ask for short identities in certain algebras.

I hope this fact may be interesting for algebraists and should stimulate a systematic study of shortest identities in various algebraic structures.

AAA84, Dresden, June 8, 2012