Synchronizing Automata and The Černý Conjecture

William DeMeo
williamdemeo@gmail.com

University of South Carolina

Graduate Algebra Seminar
University of Colorado

April 11, 2013

These slides and other resources are available at

http://williamdemeo.wordpress.com

Synchronizing automata

A ***finite automaton*** is a triple $\mathbf{A} = \langle Q, \Sigma, \delta \rangle$ where

- Q is a set of *states*
- Σ is a set of *letters* (the *input alphabet*)
- $\delta : Q \times \Sigma \to Q$ is a *transition function*

Synchronizing automata

A ***finite automaton*** is a triple $\mathbf{A} = \langle Q, \Sigma, \delta \rangle$ where

- Q is a set of *states*
- Σ is a set of *letters* (the *input alphabet*)
- $\delta : Q \times \Sigma \to Q$ is a *transition function*

If $\mathbf{A}$ is currently in state $q \in Q$, then upon input $a \in \Sigma$ the next state is $\delta(q, a)$.

A ***finite automaton*** is a triple $\mathbf{A} = \langle Q, \Sigma, \delta \rangle$ where

- Q is a set of *states*
- Σ is a set of *letters* (the *input alphabet*)
- $\delta : Q \times \Sigma \to Q$ is a *transition function*

If $\mathbf{A}$ is currently in state $q \in Q$, then upon input $a \in \Sigma$ the next state is $\delta(q, a)$.

Given an input sequence of letters $a_0, a_1, \ldots, a_{n-1}$, the resulting state is

$$\delta(\cdots \delta(\delta(q, a_0), a_1) \cdots , a_{n-1}) \tag{1.1}$$

Synchronizing automata

A ***finite automaton*** is a triple $\mathbf{A} = \langle Q, \Sigma, \delta \rangle$ where

- Q is a set of *states*
- Σ is a set of *letters* (the *input alphabet*)
- $\delta : Q \times \Sigma \to Q$ is a *transition function*

If $\mathbf{A}$ is currently in state $q \in Q$, then upon input $a \in \Sigma$ the next state is $\delta(q, a)$.

Given an input sequence of letters $a_0, a_1, \dots, a_{n-1}$, the resulting state is

$$\delta(\cdots \delta(\delta(q, a_0), a_1) \cdots, a_{n-1}) \tag{1.1}$$

If we write aq in place of $\delta(q, a)$, then (1.1) is simply $a_{n-1}a_{n-2}\dots a_1 a_0 q$.

A ***finite automaton*** is a triple $\mathbf{A} = \langle Q, \Sigma, \delta \rangle$ where

- Q is a set of *states*
- Σ is a set of *letters* (the *input alphabet*)
- $\delta : Q \times \Sigma \to Q$ is a *transition function*

If $\mathbf{A}$ is currently in state $q \in Q$, then upon input $a \in \Sigma$ the next state is $\delta(q, a)$.

Given an input sequence of letters $a_0, a_1, \ldots, a_{n-1}$, the resulting state is

$$\delta(\cdots \delta(\delta(q, a_0), a_1) \cdots , a_{n-1}) \tag{1.1}$$

If we write aq in place of $\delta(q, a)$, then (1.1) is simply $a_{n-1}a_{n-2} \ldots a_1 a_0 q$.

Let Σ^* denote the free monoid obtained by composing letters.

A *word* $w \in \Sigma^*$ is just a string of letters, $w = a_0 a_1 \ldots a_{n-1}$, which acts on a state $q \in Q$ as you expect:

$$wq = a_0 a_1 \ldots a_{n-1} q$$

SYNCHRONIZING AUTOMATA

Viewed this way, the automaton $\langle Q, \Sigma, \delta\rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma\rangle$ with universe Q and basic operations Σ.

Synchronizing automata

Viewed this way, the automaton $\langle Q, \Sigma, \delta\rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma\rangle$ with universe Q and basic operations Σ.

$\mathbf{A}$ is called *synchronizing* if there exists a constant word in Σ^*; that is, a term operation w such that $wx = wy$ for all $x, y \in Q$.

Such w are called *reset words* for $\mathbf{A}$.

Synchronizing automata

Viewed this way, the automaton $\langle Q, \Sigma, \delta \rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma \rangle$ with universe Q and basic operations Σ.

$\mathbf{A}$ is called *synchronizing* if there exists a constant word in Σ^*; that is, a term operation w such that $wx = wy$ for all $x, y \in Q$.

Such w are called *reset words* for $\mathbf{A}$.

Example

$$Q = \{0, 1, 2, 3\}$$
$$a = (1, 1, 2, 3)$$
$$b = (1, 2, 3, 0)$$

Synchronizing automata

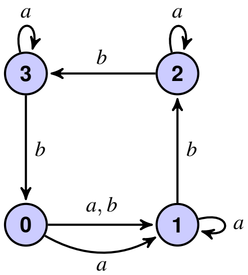
Viewed this way, the automaton $\langle Q, \Sigma, \delta\rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma\rangle$ with universe Q and basic operations Σ.

$\mathbf{A}$ is called *synchronizing* if there exists a constant word in Σ^*; that is, a term operation w such that $wx = wy$ for all $x, y \in Q$.

Such w are called *reset words* for $\mathbf{A}$.

Example

$$Q = \{0, 1, 2, 3\}$$
$$a = (1, 1, 2, 3)$$
$$b = (1, 2, 3, 0)$$
$$\mathbf{A} = \langle Q, \{a, b\}\rangle$$



Synchronizing automata

Viewed this way, the automaton $\langle Q, \Sigma, \delta \rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma \rangle$ with universe Q and basic operations Σ.

$\mathbf{A}$ is called *synchronizing* if there exists a constant word in Σ^*; that is, a term operation w such that $wx = wy$ for all $x, y \in Q$.

Such w are called *reset words* for $\mathbf{A}$.

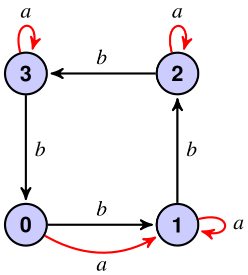
Example

$$Q = \{0, 1, 2, 3\}$$

$$a = (1, 1, 2, 3)$$

$$b = (1, 2, 3, 0)$$

$$\mathbf{A} = \langle Q, \{a, b\} \rangle$$



Synchronizing automata

Viewed this way, the automaton $\langle Q, \Sigma, \delta\rangle$ is simply a unary algebra $\mathbf{A} = \langle Q, \Sigma\rangle$ with universe Q and basic operations Σ.

$\mathbf{A}$ is called *synchronizing* if there exists a constant word in Σ^*; that is, a term operation w such that $wx = wy$ for all $x, y \in Q$.

Such w are called *reset words* for $\mathbf{A}$.

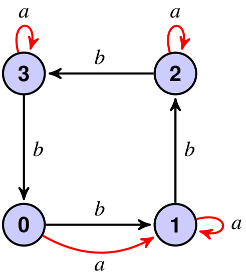
Example

$$Q = \{0, 1, 2, 3\}$$

$$a = (1, 1, 2, 3)$$

$$b = (1, 2, 3, 0)$$

$$\mathbf{A} = \langle Q, \{a, b\}\rangle$$



reset word:

abbbabbba

The notion was formalized in 1964 in a paper by Jan Černý, though implicitly it had been around since at least 1956.

The idea of synchronization is natural and of obvious importance: we aim to restore control over a device whose current state is unknown.

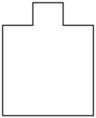
For example, our view of an orbiting satellite may be temporarily obstructed by the Moon; once it comes back into view, we regain control and reorient it (Černý's original motivation).

Synchronizing automata

In the 80's, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

In the 80's, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

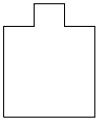
Suppose that one of the parts of a certain device has the following shape:



Such parts will move along a conveyor belt and must be sorted and oriented before assembly.

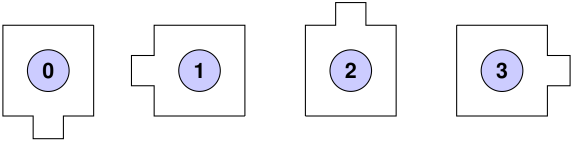
In the 80's, the notion was reinvented by engineers working in a branch of robotics which deals with part handling problems in industrial automation.

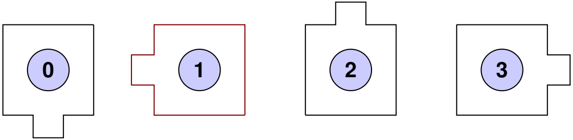
Suppose that one of the parts of a certain device has the following shape:



Such parts will move along a conveyor belt and must be sorted and oriented before assembly.

Assume that only four initial orientations are possible, namely,



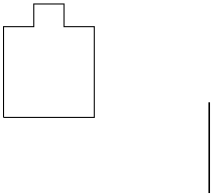


Prior to assembly the part must be in position **1**, *"bump-left"* orientation.

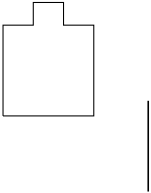
Problem: construct an orienter that will put the part in bump-left position independently of its initial orientation.

Synchronizing automata

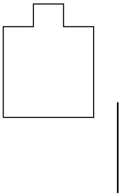
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



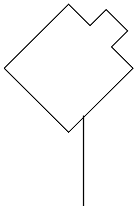
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



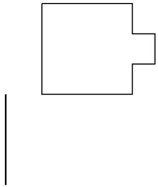
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



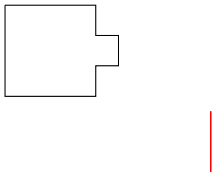
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



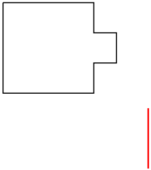
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



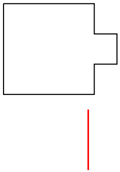
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



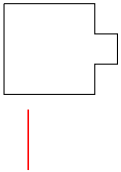
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.

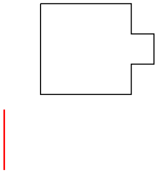


Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.

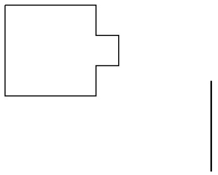


SYNCHRONIZING AUTOMATA

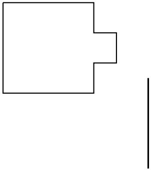
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



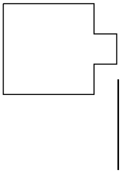
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



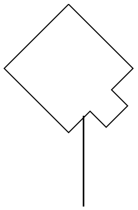
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



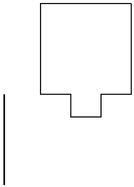
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



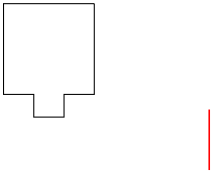
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



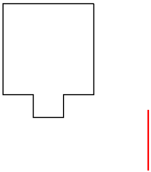
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



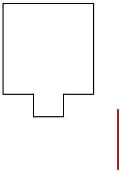
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



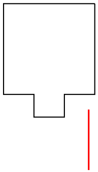
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



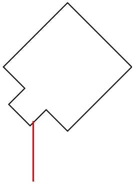
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



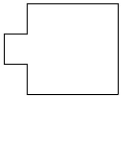
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.



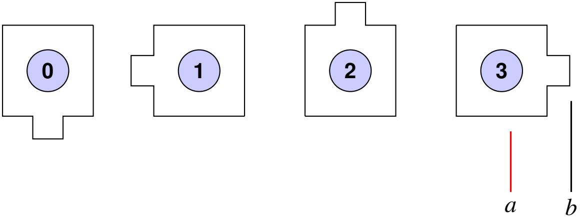
Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.

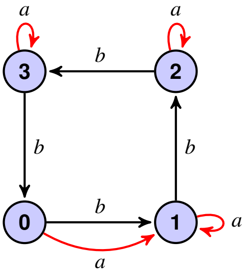


Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.

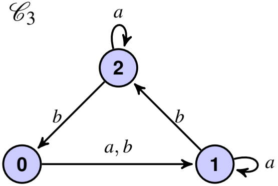


Solution: position two types of obstacles, short and tall, in the path of the part. When the part passes a tall obstacle, it always rotates 90°. When it passes a short obstacle, it rotates by 90° iff it is in bump-down orientation.





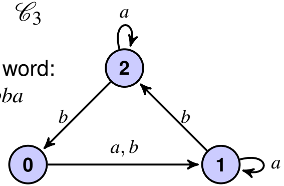
Černý's infinite family



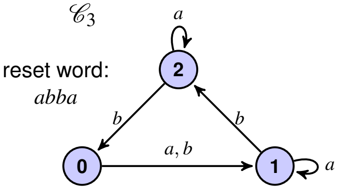
Černý's infinite family

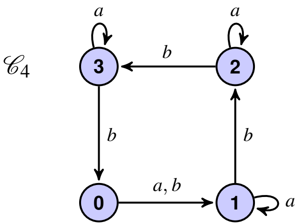
$\mathscr{C}_3$

reset word:
abba



Černý's infinite family

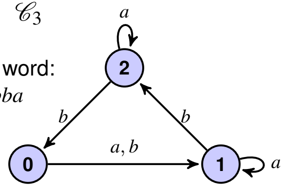




Černý's infinite family

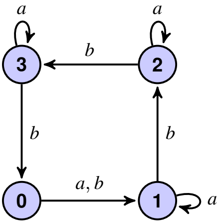
$\mathscr{C}_3$

reset word:
abba



$\mathscr{C}_4$

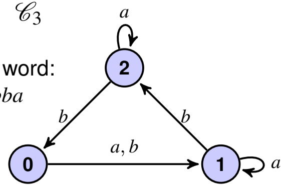
reset word:
abbbabbba



ČERNÝ'S INFINITE FAMILY

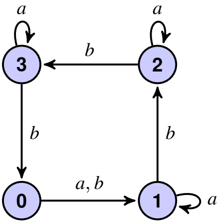
$\mathscr{C}_3$

reset word:
abba

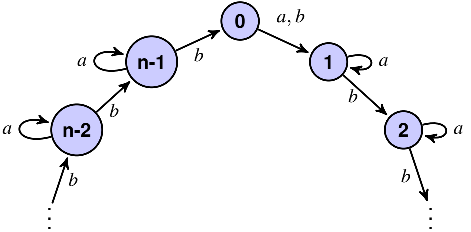


$\mathscr{C}_4$

reset word:
abbbabbba



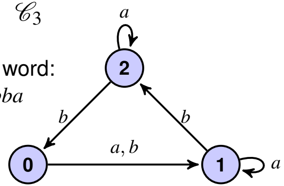
$\mathscr{C}_n$



ČERNÝ'S INFINITE FAMILY

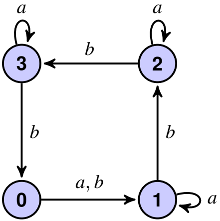
$\mathscr{C}_3$

reset word: *abba*

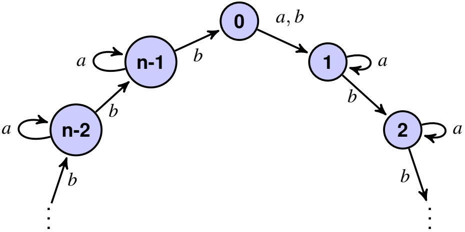


$\mathscr{C}_4$

reset word: *abbbabbba*



$\mathscr{C}_n$

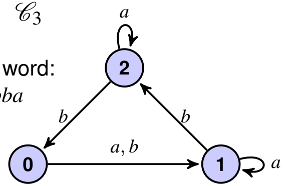


reset word: $abbb\cdots babbb\cdots ba = (ab^{n-1})^{n-2}a$

ČERNÝ'S INFINITE FAMILY

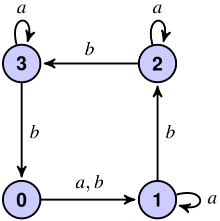
$\mathscr{C}_3$

reset word:
abba

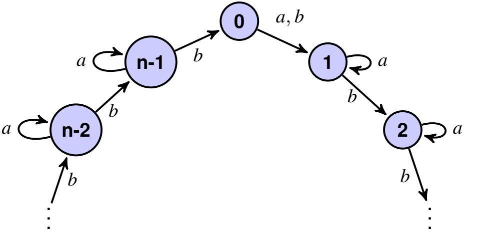


$\mathscr{C}_4$

reset word:
abbbabbba



$\mathscr{C}_n$



reset word: $abbb\cdots babbb\cdots ba = (ab^{n-1})^{n-2}a$

length: $n(n-2)+1 = (n-1)^2$

The Černý bound

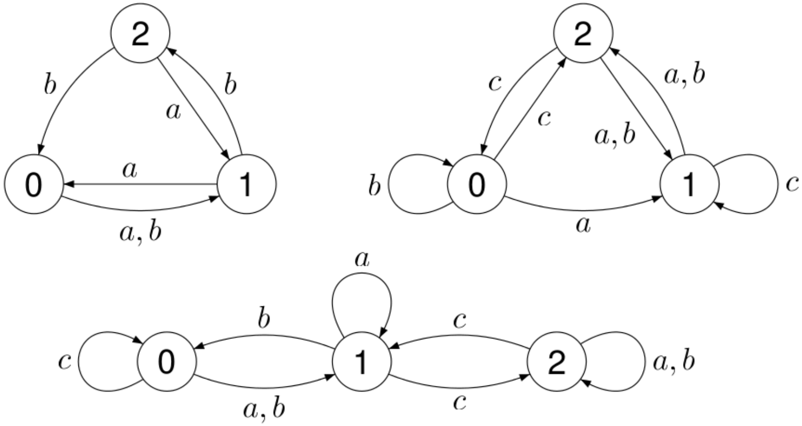
- A synchronizing automaton with n states *reaches the Černý bound* if the minimum length of all reset words is $(n-1)^2$.

- We present all known proper synchronizing automata with $n > 2$ states.

("proper" means removal of any letter results in a nonsynchronizing automaton)

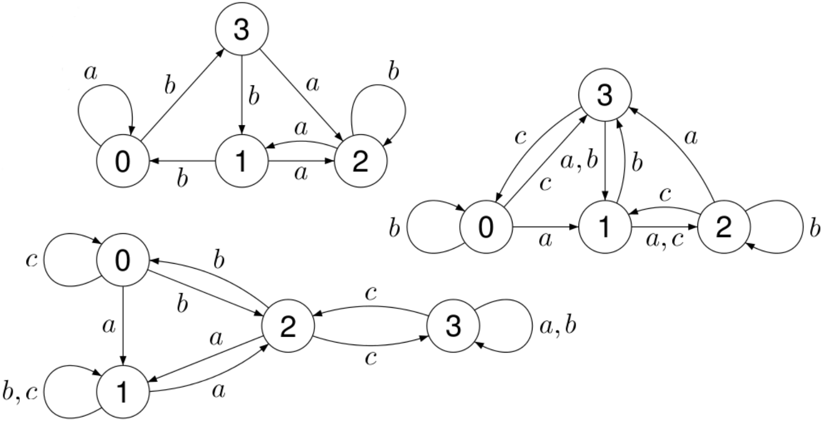
SPORADIC EXAMPLES

There are three sporadic examples on 3 states:



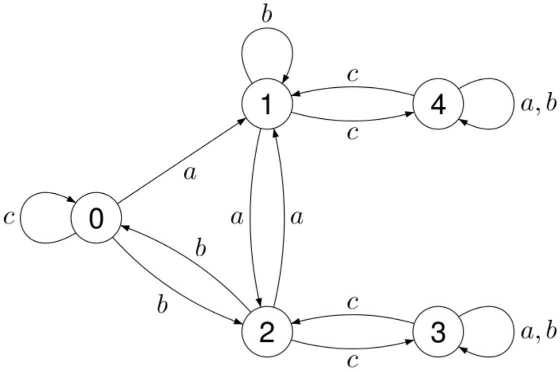
SPORADIC EXAMPLES

For 4 states, three sporadic examples are known:

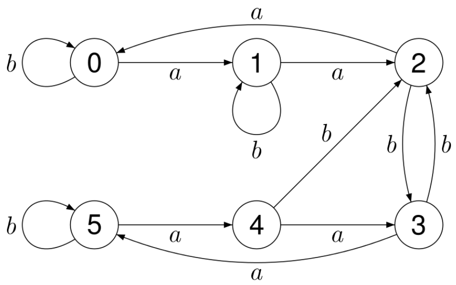


ROMAN'S EXAMPLE

For 5 states, a synchronizing automaton reaching the Černý bound was discovered by Adam Roman:

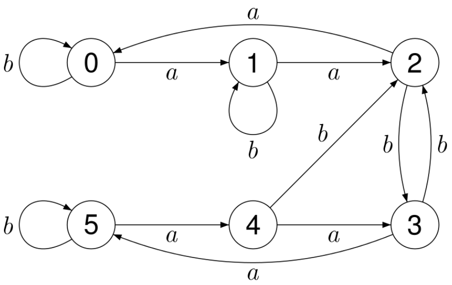


The last in our list and the most remarkable example was published in 2001 by Jarkko Kari.



Kari's Example

The last in our list and the most remarkable example was published in 2001 by Jarkko Kari.



Minimum length reset word: $a\,b\,b\,a\,b\,a\,b\,b\,a\,b\,b\,a\,b\,a\,a\,b\,b\,a\,b\,a\,b\,a\,b\,b\,a$

CLONES

A BRIEF REVIEW

- For nonempty set A, let $\mathrm{Op}(A)$ be the set of all operations on A. That is,

$$\mathrm{Op}(A) = \bigcup_{n<\omega} A^{(A^n)}$$

- For nonempty set A, let $\mathrm{Op}(A)$ be the set of all operations on A. That is,

$$\mathrm{Op}(A) = \bigcup_{n<\omega} A^{(A^n)}$$

- For each $k \leq n < \omega$, the kth *projection operation* is

$$p_k^n(x_1, \ldots, x_n) = x_k$$

- If $f \in A^{(A^n)}$ and $g_1, \ldots, g_n \in A^{(A^k)}$, then a *generalized composition* is

$$f[g_1, \ldots, g_n] : (x_1, \ldots, x_k) \mapsto f(g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k))$$

Clones

A brief review

- For nonempty set A, let $\mathrm{Op}(A)$ be the set of all operations on A. That is,

$$\mathrm{Op}(A) = \bigcup_{n<\omega} A^{(A^n)}$$

- For each $k \leq n < \omega$, the kth *projection operation* is

$$p_k^n(x_1, \ldots, x_n) = x_k$$

- If $f \in A^{(A^n)}$ and $g_1, \ldots, g_n \in A^{(A^k)}$, then a *generalized composition* is

$$f[g_1, \ldots, g_n] : (x_1, \ldots, x_k) \mapsto f(g_1(x_1, \ldots, x_k), \ldots, g_n(x_1, \ldots, x_k))$$

- A *clone* on A is a subset of $\mathrm{Op}(A)$ that contains all projections and is closed under generalized compositions.

- For nonempty set A, let $\mathrm{Op}(A)$ be the set of all operations on A. That is,

$$\mathrm{Op}(A) = \bigcup_{n<\omega} A^{(A^n)}$$

- For each $k \le n < \omega$, the kth *projection operation* is

$$p_k^n(x_1, \dots, x_n) = x_k$$

- If $f \in A^{(A^n)}$ and $g_1, \dots, g_n \in A^{(A^k)}$, then a *generalized composition* is

$$f[g_1, \dots, g_n] : (x_1, \dots, x_k) \mapsto f(g_1(x_1, \dots, x_k), \dots, g_n(x_1, \dots, x_k))$$

- A *clone* on A is a subset of $\mathrm{Op}(A)$ that contains all projections and is closed under generalized compositions.
- Given $F \subseteq \mathrm{Op}(A)$, let $\mathrm{Clo}^A(F)$ denote the smallest clone on A containing F.
- Let $\mathrm{Clo}_n^A(F)$ denote the n-ary members of $\mathrm{Clo}^A(F)$.

Clones

A brief review

- If $\mathbf{A} = \langle A, F \rangle$ is an algebra, then $\mathrm{Clo}^A(F)$ is called the *clone of term operations on* $\mathbf{A}$, which we denote by $\mathrm{Clo}(\mathbf{A})$.

- If $\mathbf{A} = \langle A, F \rangle$ is an algebra, then $\mathrm{Clo}^A(F)$ is called the *clone of term operations on* $\mathbf{A}$, which we denote by $\mathrm{Clo}(\mathbf{A})$.

Theorem

Given $\mathbf{A}$ *and* $n \in \omega$*, the set* $\mathrm{Clo}_n(\mathbf{A})$ *is a subuniverse of the direct power algebra* $\mathbf{A}^{(A^n)}$*. It is generated by the projections* $\{p^n_1, p^n_2, \ldots, p^n_n\}$*.*

Clones

A brief review

- If $\mathbf{A} = \langle A, F \rangle$ is an algebra, then $\mathrm{Clo}^A(F)$ is called the *clone of term operations on* $\mathbf{A}$, which we denote by $\mathrm{Clo}(\mathbf{A})$.

Theorem

Given $\mathbf{A}$ and $n \in \omega$, the set $\mathrm{Clo}_n(\mathbf{A})$ is a subuniverse of the direct power algebra $\mathbf{A}^{(A^n)}$. It is generated by the projections $\{p_1^n, p_2^n, \ldots, p_n^n\}$.

(Note that $\mathrm{Clo}_1(\mathbf{A})$ is the subuniverse of $\mathbf{A}^A$ generated by p_1^1, the identity map.)

- If $\mathbf{A} = \langle A, F\rangle$ is an algebra, then $\mathrm{Clo}^A(F)$ is called the *clone of term operations on* $\mathbf{A}$, which we denote by $\mathrm{Clo}(\mathbf{A})$.

THEOREM

Given $\mathbf{A}$ *and* $n \in \omega$*, the set* $\mathrm{Clo}_n(\mathbf{A})$ *is a subuniverse of the direct power algebra* $\mathbf{A}^{(A^n)}$*. It is generated by the projections* $\{p^n_1, p^n_2, \ldots, p^n_n\}$*.*

(Note that $\mathrm{Clo}_1(\mathbf{A})$ is the subuniverse of $\mathbf{A}^A$ generated by p^1_1, the identity map.)

EXAMPLE

$$A = \{0, 1, 2\} \qquad a = (1, 1, 2) \qquad b = (1, 2, 0) \qquad \mathbf{A} = \langle A, \{a, b\}\rangle$$

- If $\mathbf{A} = \langle A, F \rangle$ is an algebra, then $\mathrm{Clo}^A(F)$ is called the *clone of term operations on* $\mathbf{A}$, which we denote by $\mathrm{Clo}(\mathbf{A})$.

THEOREM

Given $\mathbf{A}$ *and* $n \in \omega$*, the set* $\mathrm{Clo}_n(\mathbf{A})$ *is a subuniverse of the direct power algebra* $\mathbf{A}^{(A^n)}$*. It is generated by the projections* $\{p_1^n, p_2^n, \ldots, p_n^n\}$*.*

(Note that $\mathrm{Clo}_1(\mathbf{A})$ is the subuniverse of $\mathbf{A}^A$ generated by p_1^1, the identity map.)

EXAMPLE

$$A = \{0, 1, 2\} \qquad a = (1, 1, 2) \qquad b = (1, 2, 0) \qquad \mathbf{A} = \langle A, \{a, b\} \rangle$$

$\mathrm{Clo}_1(\mathbf{A})$ is the subuniverse of $\mathbf{A}^A = \mathbf{A} \times \mathbf{A} \times \mathbf{A}$ generated by $(0, 1, 2)$.

Terms

A brief review

- Let $\mathcal{F}$ be a set of operation symbols and let $\rho : \mathcal{F} \to \omega$ be a similarity type.
- Let $F_n = \{f \in \mathcal{F} \mid \rho(f) = n\}$ be the set of n-ary operation symbols in $\mathcal{F}$.

TERMS

A BRIEF REVIEW

- Let $\mathcal{F}$ be a set of operation symbols and let $\rho : \mathcal{F} \to \omega$ be a similarity type.
- Let $F_n = \{f \in \mathcal{F} \mid \rho(f) = n\}$ be the set of n-ary operation symbols in $\mathcal{F}$.
- Let X be a set, disjoint from $\mathcal{F}$, whose elements we call *variables*.
- A *word on the alphabet* $X \cup \mathcal{F}$ is a finite string $a_1 a_2 \cdots a_n$, where $a_i \in X \cup \mathcal{F}$. The *product* of words is defined by concatenation.

Terms

A brief review

- Let $\mathcal{F}$ be a set of operation symbols and let $\rho : \mathcal{F} \to \omega$ be a similarity type.
- Let $F_n = \{f \in \mathcal{F} \mid \rho(f) = n\}$ be the set of n-ary operation symbols in $\mathcal{F}$.
- Let X be a set, disjoint from $\mathcal{F}$, whose elements we call *variables*.
- A *word on the alphabet* $X \cup \mathcal{F}$ is a finite string $a_1 a_2 \cdots a_n$, where $a_i \in X \cup \mathcal{F}$. The *product* of words is defined by concatenation.
- The set $T_\rho(X)$ of *terms of type* ρ *over* X is the smallest set T of words on the alphabet $X \cup \mathcal{F}$ such that
  - $X \cup F_0 \subseteq T$
  - If $t_1, \dots, t_n \in T$ and $f \in F_n$, then $f t_1 t_2 \cdots t_n \in T$.
- If $t_1, \dots, t_n$ are terms and $f \in F_n$ is an n-ary operation symbol, we often write $f(t_1, t_2, \cdots, t_n)$ instead of $f t_1 t_2 \cdots t_n$.

Terms

A brief review

- For each $f \in F_n$ let $f^{\mathbf{T}_\rho(X)}$ be the n-ary operation on $T_\rho(X)$ that maps $(t_1, \ldots, t_n)$ to $ft_1 \ldots t_n$.
- Define *term algebra of type* ρ as follows

$$\mathbf{T}_\rho(X) = \langle T_\rho(X), \{f^{\mathbf{T}_\rho(X)} : f \in \mathcal{F}\}\rangle$$

$\mathbf{T}_\rho(X)$ is free in $\mathscr{A}_\rho$ over X.

- For each $f \in F_n$ let $f^{\mathbf{T}_\rho(X)}$ be the n-ary operation on $T_\rho(X)$ that maps $(t_1, \ldots, t_n)$ to $ft_1 \ldots t_n$.

- Define *term algebra of type* ρ as follows

$$\mathbf{T}_\rho(X) = \langle T_\rho(X), \{f^{\mathbf{T}_\rho(X)} : f \in \mathcal{F}\}\rangle$$

  $\mathbf{T}_\rho(X)$ is free in $\mathscr{A}_\rho$ over X.

- Let $X_n = \{x_1, \ldots, x_n\}$. For $t(x_1, \ldots, x_n) \in T_\rho(X_n)$ and $\mathbf{A}$ an algebra of type ρ, define an n-ary operation $t^{\mathbf{A}}$ on A by recursion on the "height" of t:

  – if t is the variable x_i then $t^{\mathbf{A}}(a_1, \ldots, a_n) = a_i$

  – if $t = fs_1 s_2 \ldots s_k$ where $f \in F_k$ and $s_1, \ldots, s_k$ are terms, then

$$t^{\mathbf{A}}(a_1, \ldots, a_n) = f^{\mathbf{A}}(s_1^{\mathbf{A}}(a_1, \ldots, a_n), \ldots, s_k^{\mathbf{A}}(a_1, \ldots, a_n))$$

Connecting clones and terms

Consider the map $\phi : X_n \to A^{(A^n)}$ defined by $\phi(x_i) = p_i^n$.

Connecting clones and terms

Consider the map $\phi : X_n \to A^{(A^n)}$ defined by $\phi(x_i) = p_i^n$.

By freeness of $\mathbf{T}_\rho(X_n)$, there exists a unique hom $\bar{\phi} : \mathbf{T}_\rho(X_n) \to \mathbf{A}^{(A^n)}$ extending ϕ.

...and for any $t \in T_\rho(X_n)$, we have $\bar{\phi}(t) = t^{\mathbf{A}}$ (induct on height of t)

Connecting clones and terms

Consider the map $\phi : X_n \to A^{(A^n)}$ defined by $\phi(x_i) = p_i^n$.

By freeness of $\mathbf{T}_\rho(X_n)$, there exists a unique hom $\bar{\phi} : \mathbf{T}_\rho(X_n) \to \mathbf{A}^{(A^n)}$ extending ϕ.

...and for any $t \in T_\rho(X_n)$, we have $\bar{\phi}(t) = t^{\mathbf{A}}$ (induct on height of t)

Theorem

$$\mathrm{Clo}_n(\mathbf{A}) = \{t^{\mathbf{A}} : t \in T_\rho(X_n)\}$$

Proof: For the unique hom above, we have $\bar{\phi}(t) = t^{\mathbf{A}} \in \mathrm{Clo}_n(\mathbf{A})$.

The image $\bar{\phi}(T_\rho(X_n))$ contains all the projections.

The projections generate the algebra $\mathbf{Clo}_n(\mathbf{A})$, so

$$\bar{\phi}(T_\rho(X_n)) = \mathrm{Clo}_n(\mathbf{A})$$

CONNECTING CLONES AND TERMS

...and finally, we recall that if $\mathbf{F}_{\mathbf{A}}(X_n)$ denotes a free algebra in $V(\mathbf{A})$ with free generating set X_n, then

$$\mathbf{F}_{\mathbf{A}}(X_n) \cong \mathbf{T}_{\rho}(X_n)/\Theta$$

where two terms are equivalent mod Θ when they induce the same term operation on every member of $V(\mathbf{A})$.

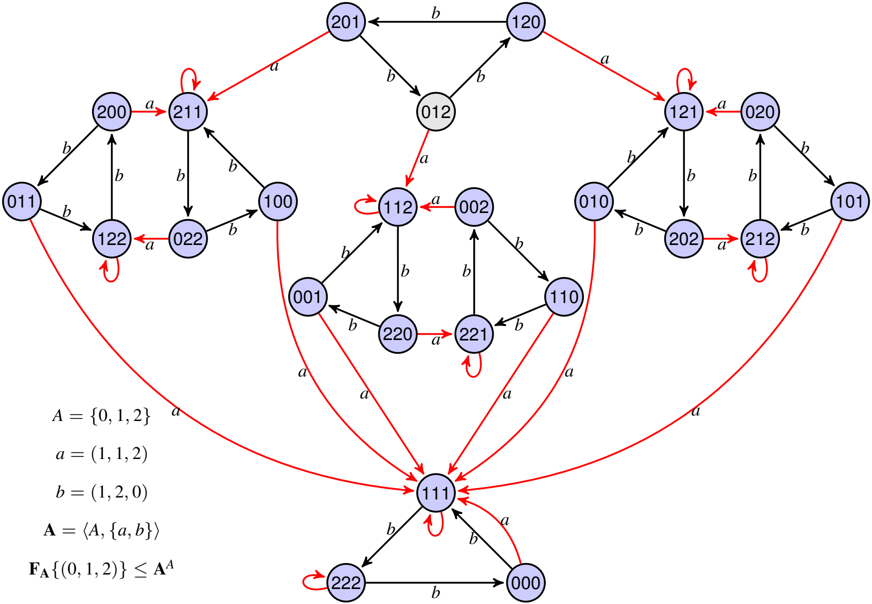
Thus, taking Θ to be the kernel of $\bar{\phi}$, we have $\mathbf{F}_{\mathbf{A}}(X_n) \cong \mathrm{Clo}_n(\mathbf{A})$.

Example

$$A = \{0, 1, 2\} \qquad a = (1, 1, 2) \qquad b = (1, 2, 0) \qquad \mathbf{A} = \langle A, \{a, b\}\rangle$$

$\mathrm{Clo}_1(\mathbf{A})$ is the subuniverse of $\mathbf{A}^A = \mathbf{A} \times \mathbf{A} \times \mathbf{A}$ generated by $(0, 1, 2)$

$$\mathrm{Clo}_1(\mathbf{A}) \cong \mathbf{F}_{\mathbf{A}}\{(0, 1, 2)\}$$



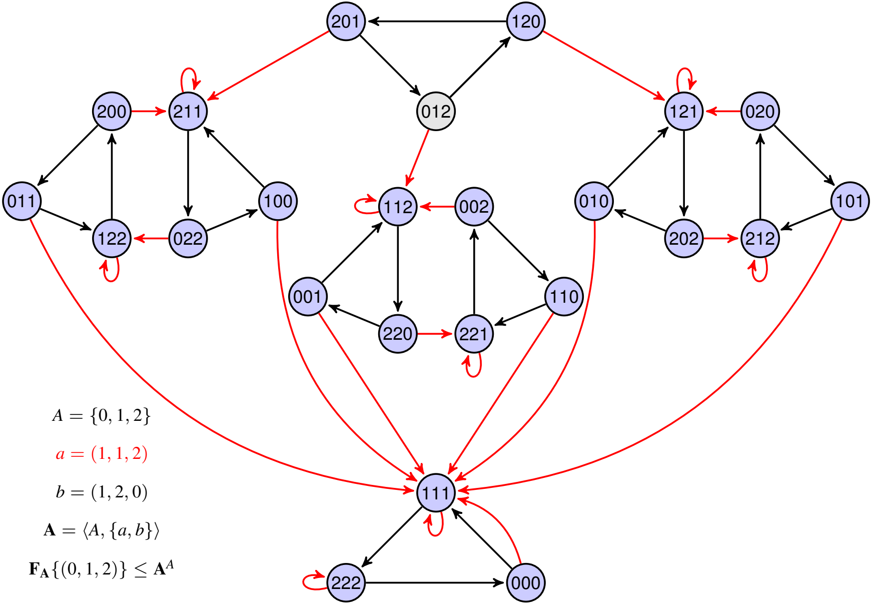
$$A = \{0, 1, 2\}$$

$$a = (1, 1, 2)$$

$$b = (1, 2, 0)$$

$$\mathbf{A} = \langle A, \{a, b\}\rangle$$

$$\mathbf{F}_{\mathbf{A}}\{(0, 1, 2)\} \leq \mathbf{A}^{A}$$



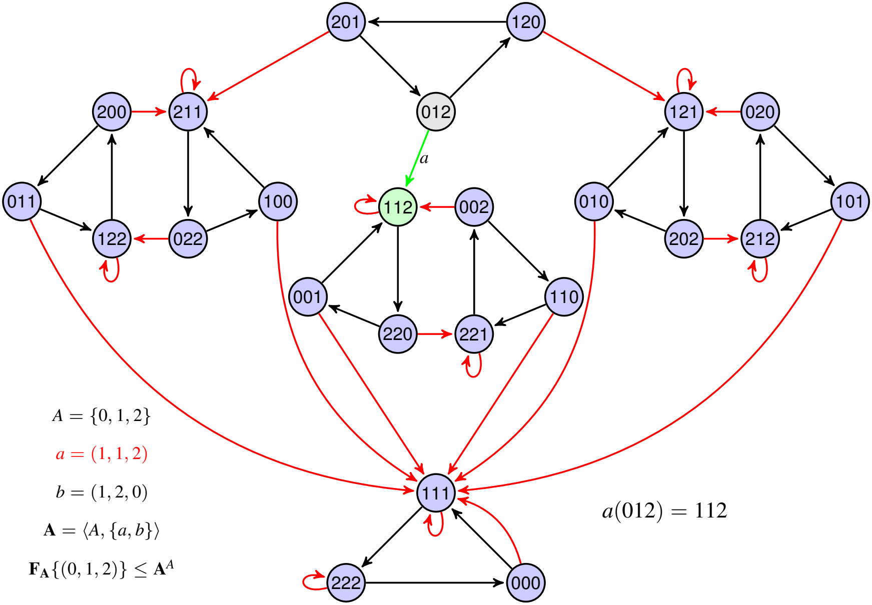
$A = \{0, 1, 2\}$

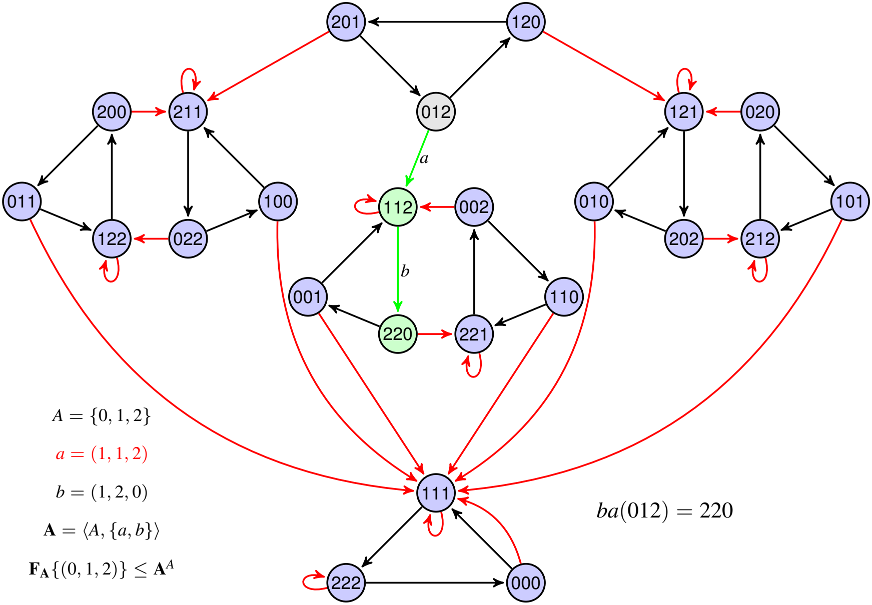
$a = (1, 1, 2)$

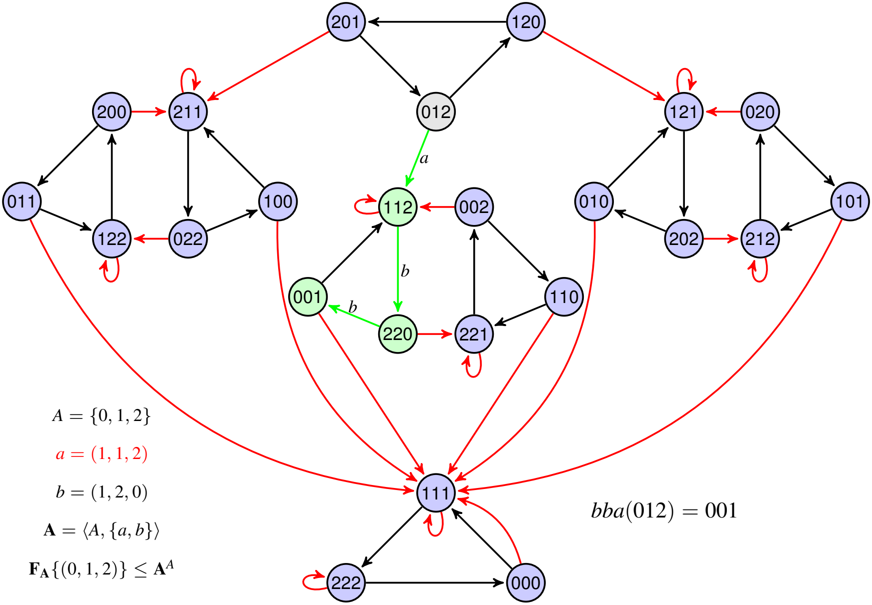
$b = (1, 2, 0)$

$\mathbf{A} = \langle A, \{a, b\}\rangle$

$\mathbf{F}_{\mathbf{A}}\{(0, 1, 2)\} \leq \mathbf{A}^A$


201
120
200
211
012
121
020
a
011
100
112
002
010
101
122
022
202
212
001
110
220
221
111
222
000
$A = \{0, 1, 2\}$
$a = (1, 1, 2)$
$b = (1, 2, 0)$
$\mathbf{A} = \langle A, \{a, b\} \rangle$
$\mathbf{F_A}\{(0, 1, 2)\} \leq \mathbf{A}^A$
$a(012) = 112$


201
120
200
211
012
121
020
011
122
022
100
112
002
001
220
221
110
010
202
212
101
111
222
000
a
b
$A = \{0, 1, 2\}$
$a = (1, 1, 2)$
$b = (1, 2, 0)$
$\mathbf{A} = \langle A, \{a, b\} \rangle$
$\mathbf{F_A}\{(0, 1, 2)\} \leq \mathbf{A}^A$
$ba(012) = 220$



$A = \{0, 1, 2\}$

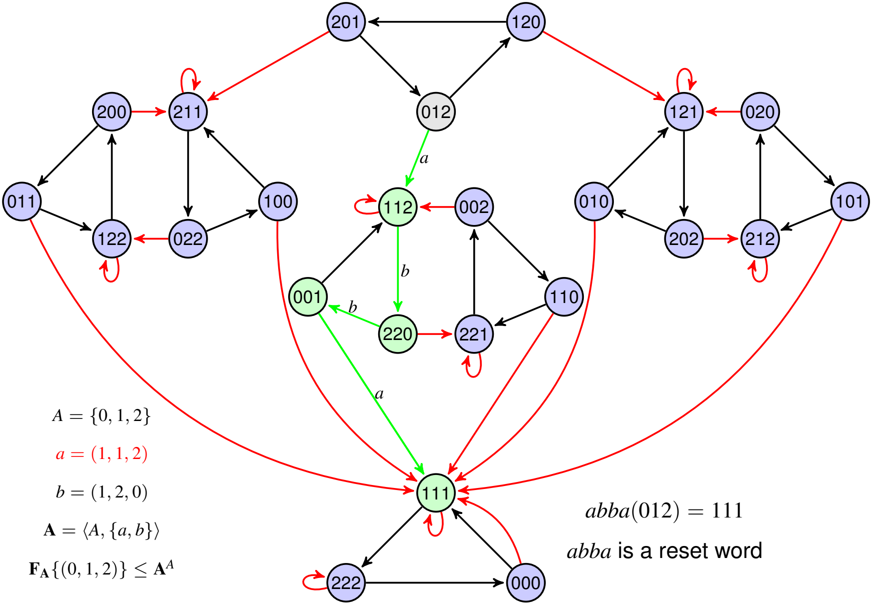
$a = (1, 1, 2)$

$b = (1, 2, 0)$

$\mathbf{A} = \langle A, \{a, b\} \rangle$

$\mathbf{F_A}\{(0, 1, 2)\} \leq \mathbf{A}^A$

$bba(012) = 001$



$A = \{0, 1, 2\}$

$a = (1, 1, 2)$

$b = (1, 2, 0)$

$\mathbf{A} = \langle A, \{a, b\} \rangle$

$\mathbf{F_A}\{(0, 1, 2)\} \leq \mathbf{A}^A$

$abba(012) = 111$

abba is a reset word