St Andrews, September 5–9, 2006

Interpreting graphs in 0-simple semigroups with involution with applications to computational complexity and the finite basis problem

Mikhail Volkov (joint work with Marcel Jackson)

Ural State University, Ekaterinburg, Russia

Overview

We are interested in:

Overview

We are interested in:

- Computational complexity of the (Pseudo)variety Membership Problem (PMP)

Overview

We are interested in:

- Computational complexity of the (Pseudo)variety Membership Problem (PMP)

and

- the Finite Basis Problem (FBP)

Overview

We are interested in:

- Computational complexity of the (Pseudo)variety Membership Problem (PMP)

and

- the Finite Basis Problem (FBP)

In Lecture 1 I'll explain our motivation while in Lectures 2 and 3 I'll present some new techniques that has proved to be very efficient for semigroups equipped with an additional unary operation

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

- is easy to explain;

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

- is easy to explain;
- is hard to prove;

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

- is easy to explain;
- is hard to prove;
- has many surprising connections,

Motivation

For motivation, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

- is easy to explain;
- is hard to prove;
- has many surprising connections,
- is related to my recent research.

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

- a tape divided into cells filled with letters of a finite input alphabet Σ (the numbers of cells is not bounded);

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

- a tape divided into cells filled with letters of a finite input alphabet Σ (the numbers of cells is not bounded);
- h reading heads that can move along the tape independently of each other (but preserving the relative order of the heads: the first head always remains on the left of the second etc) and read symbols in the cells that they observe;

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

- a tape divided into cells filled with letters of a finite input alphabet Σ (the numbers of cells is not bounded);
- h reading heads that can move along the tape independently of each other (but preserving the relative order of the heads: the first head always remains on the left of the second etc) and read symbols in the cells that they observe;
- finite read-only memory that contains two lists of words of length $\leq h$ over Σ:

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

- a tape divided into cells filled with letters of a finite input alphabet Σ (the numbers of cells is not bounded);
- h reading heads that can move along the tape independently of each other (but preserving the relative order of the heads: the first head always remains on the left of the second etc) and read symbols in the cells that they observe;
- finite read-only memory that contains two lists of words of length $\leq h$ over Σ: passwords

Hydra automata

An *h-head hydra automaton* $\mathscr{H}$ is a very simple device consisting of:

- a tape divided into cells filled with letters of a finite input alphabet Σ (the numbers of cells is not bounded);
- h reading heads that can move along the tape independently of each other (but preserving the relative order of the heads: the first head always remains on the left of the second etc) and read symbols in the cells that they observe;
- finite read-only memory that contains two lists of words of length $\leq h$ over Σ: passwords and taboos.

Hydra automata

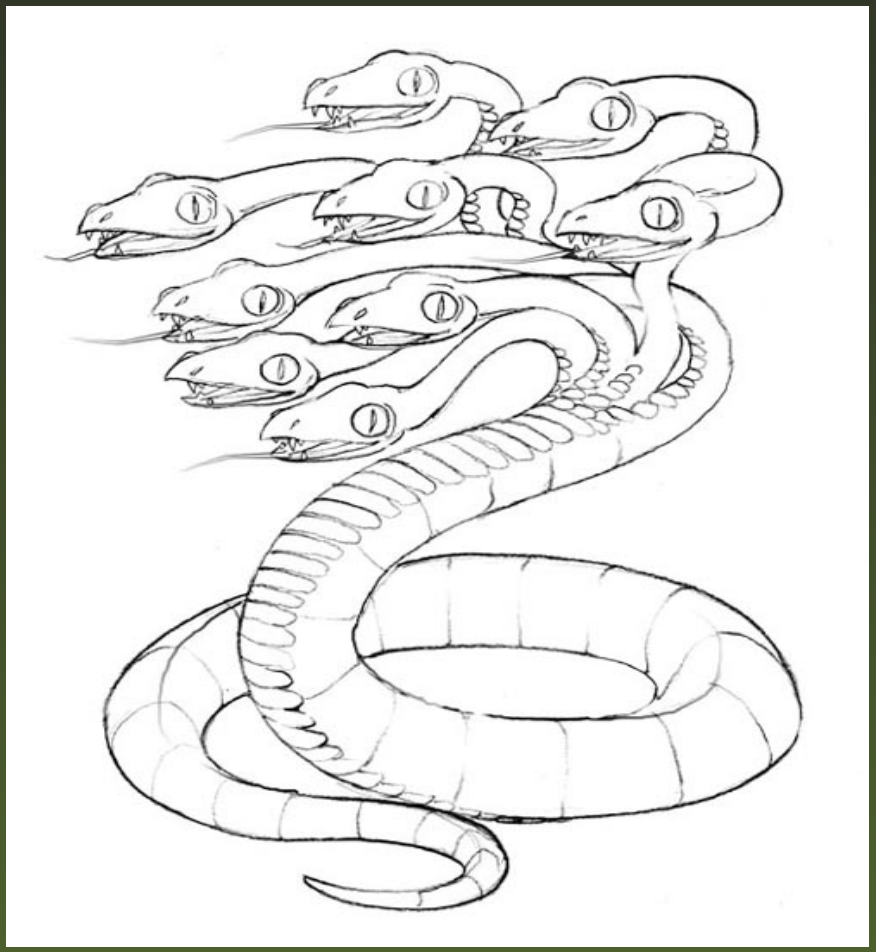


Figure 1: A 9-head hydra

Hydra automata

Figure 3: A 7-head hydra automaton

Hydra automata

Figure 2: A 7-head hydra automaton

Hydra automata

Figure 2: A 7-head hydra automaton

Hydra automata

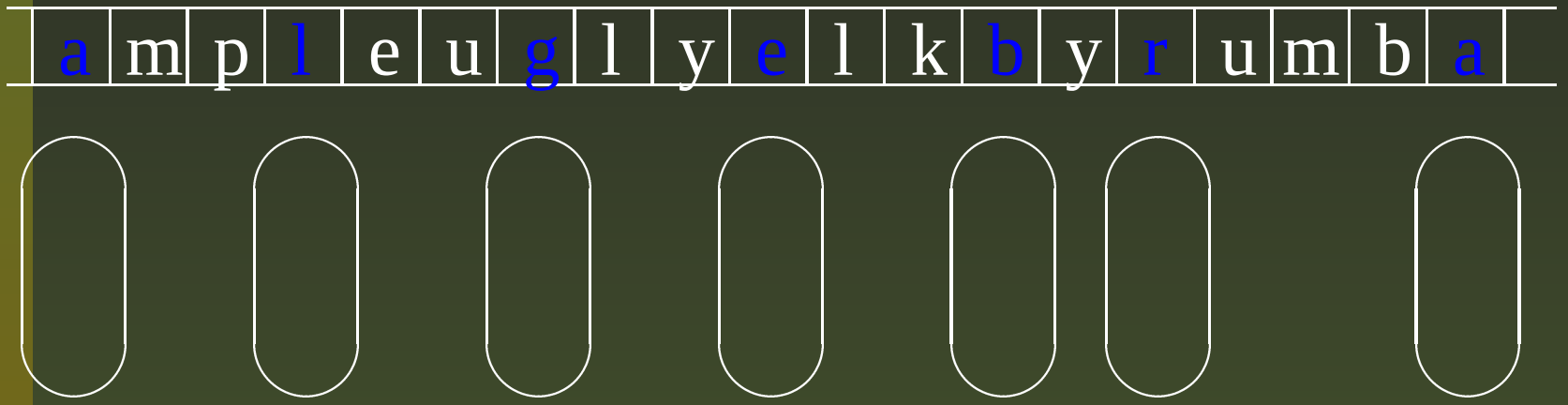


Figure 2: A 7-head hydra automaton

Hydra automata

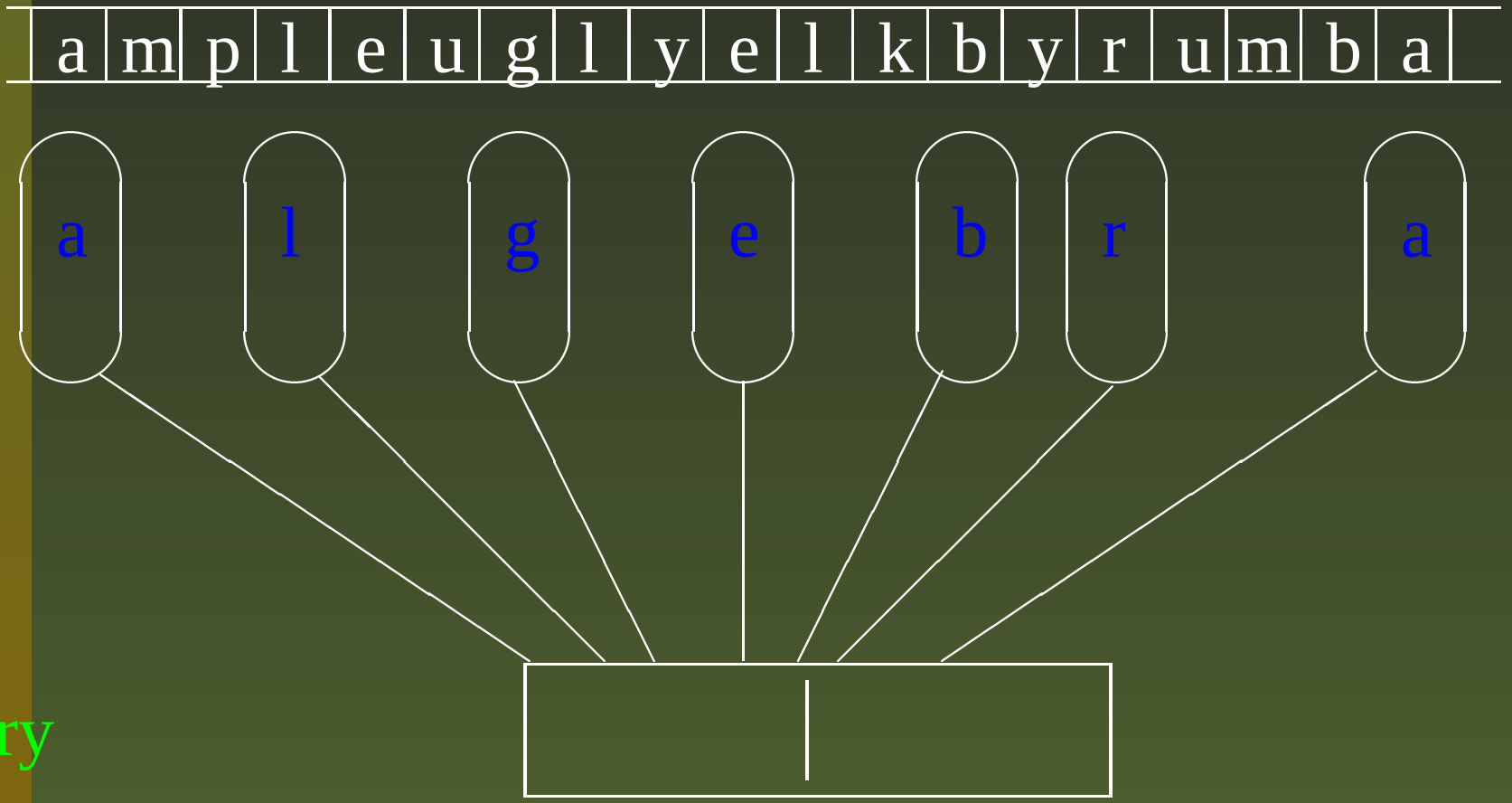


Figure 2: A 7-head hydra automaton

Hydra automata

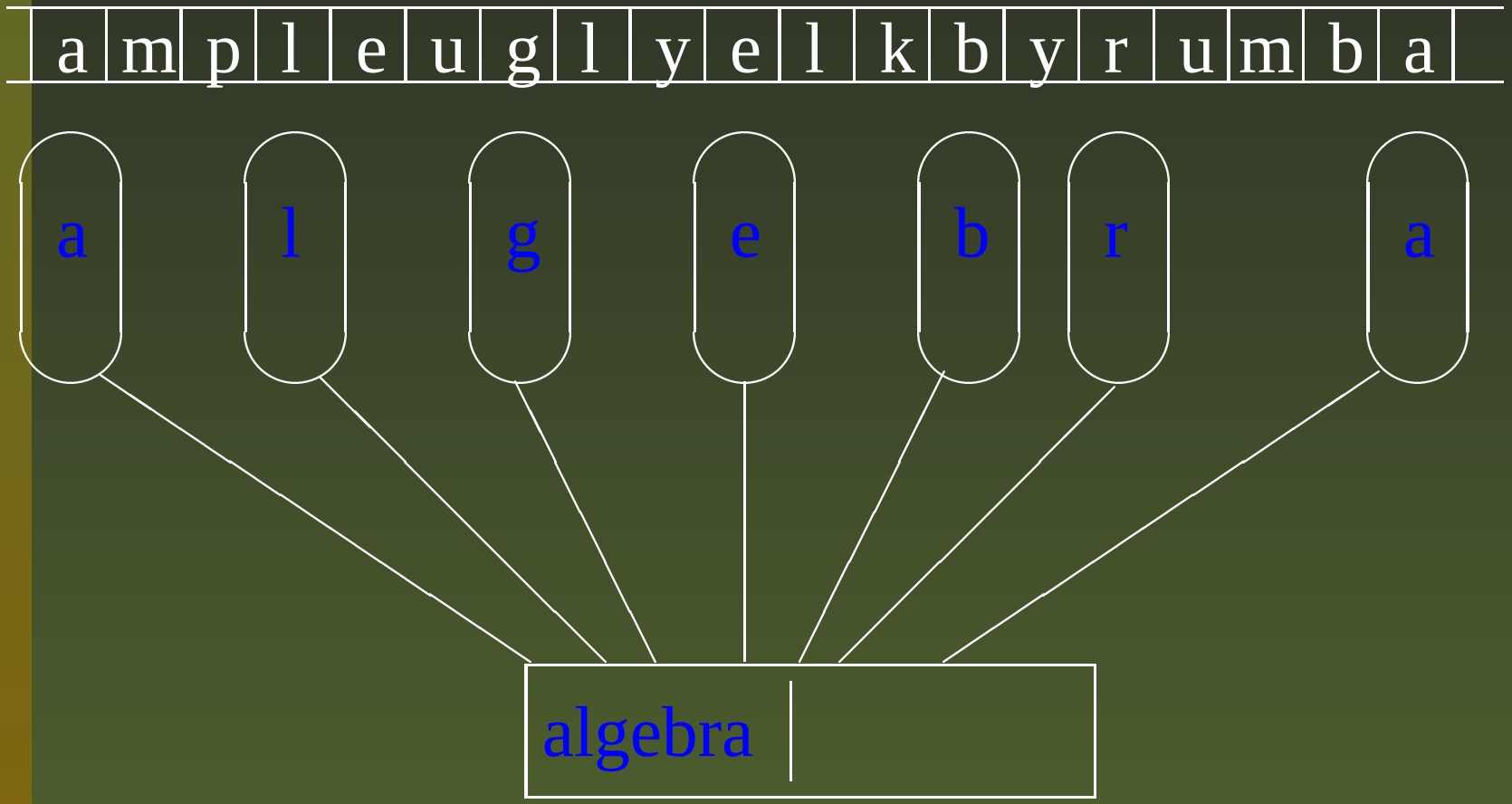


Figure 2: A 7-head hydra automaton

Hydra automata

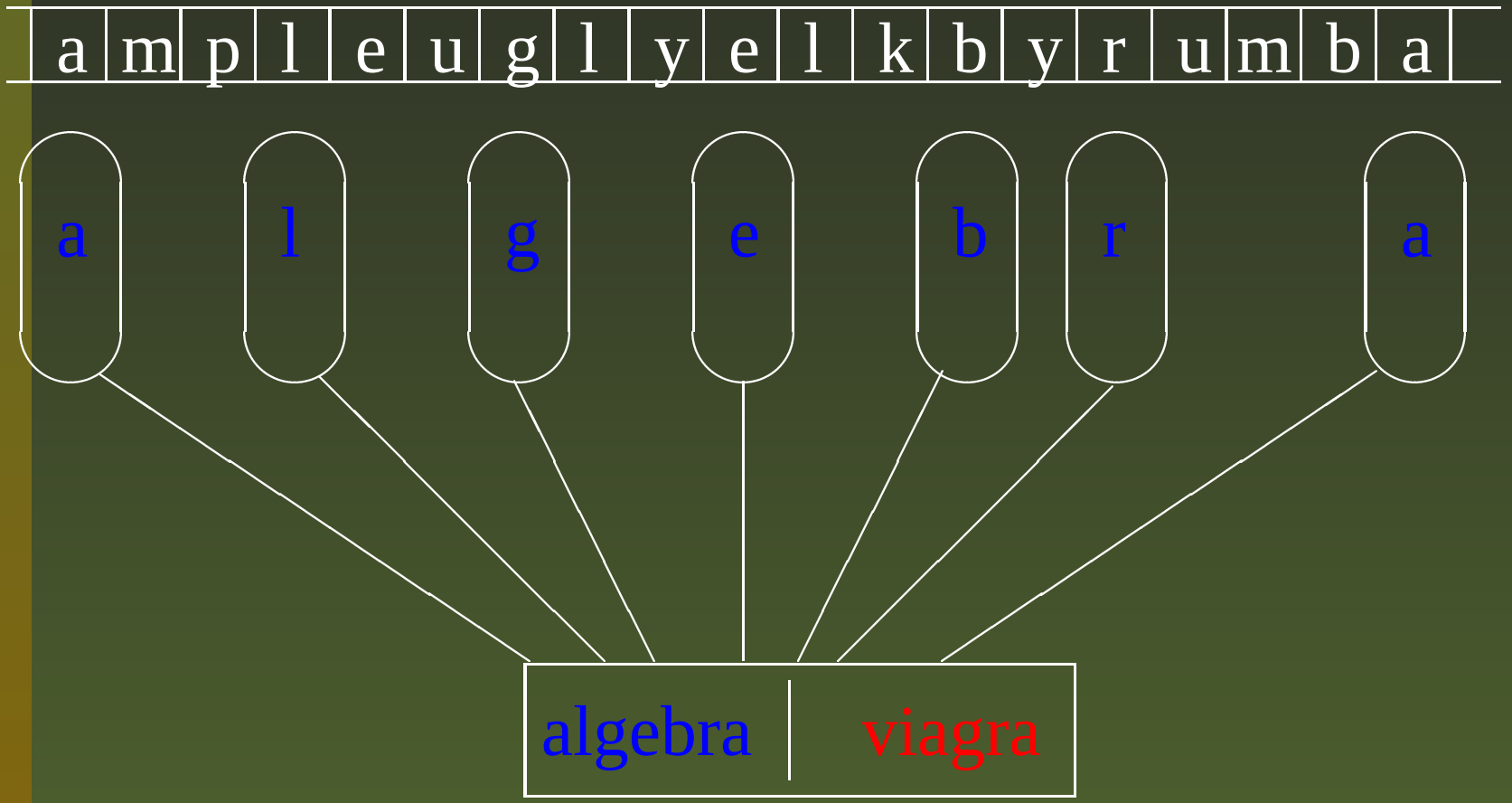


Figure 2: A 7-head hydra automaton

Hydra automata

A hydra automaton $\mathscr{H}$ *accepts* a word $w \in \Sigma^*$ if it finds in w one of the passwords but none of the taboos. Otherwise it *rejects* w.

Hydra automata

A hydra automaton $\mathscr{H}$ *accepts* a word $w \in \Sigma^*$ if it finds in w one of the passwords but none of the taboos. Otherwise it *rejects* w.

For instance the automaton on Fig. 2 accepts the word written on the tape (AmpleUglyElkByRumba) as it finds in it the password algebra but not the tabooed word viagra.

Hydra automata

A hydra automaton $\mathscr{H}$ *accepts* a word $w \in \Sigma^*$ if it finds in w one of the passwords but none of the taboos. Otherwise it *rejects* w.

For instance the automaton on Fig. 2 accepts the word written on the tape (AmpleUglyElkByRumba) as it finds in it the password algebra but not the tabooed word viagra.

A language $L \subseteq \Sigma^*$ is said to be *recognized* by a hydra automaton $\mathscr{H}$ if $\mathscr{H}$ accepts exactly words that are members of L. Such languages are called *piecewise testable*.

Piecewise testable languages

More precisely, a language $L \subseteq \Sigma^*$ is called *piecewise testable* of height $\leq h$ if L can be recognized by a hydra automaton with h heads.

Piecewise testable languages

More precisely, a language $L \subseteq \Sigma^*$ is called *piecewise testable* of height $\leq h$ if L can be recognized by a hydra automaton with h heads.

Let $PTL(\Sigma)$ [resp. $PTL_h(\Sigma)$] denote the family of all piecewise testable languages [of height $\leq h$] over a fixed alphabet Σ.

Piecewise testable languages

More precisely, a language $L \subseteq \Sigma^*$ is called *piecewise testable* of height $\leq h$ if L can be recognized by a hydra automaton with h heads.

Let $PTL(\Sigma)$ [resp. $PTL_h(\Sigma)$] denote the family of all piecewise testable languages [of height $\leq h$] over a fixed alphabet Σ.

Simon's hierarchy of piecewise testable languages:

$$PTL_1(\Sigma) \subset PTL_2(\Sigma) \subset PTL_3(\Sigma) \subset \ldots$$

$$\subset PTL(\Sigma) = \bigcup_{h=1}^{\infty} PTL_h(\Sigma)$$

Piecewise testable languages

Question 1. *Given a language $L \subseteq \Sigma^*$, how to decide whether or not L is piecewise testable?*

Piecewise testable languages

Question 1. *Given a language $L \subseteq \Sigma^*$, how to decide whether or not L is piecewise testable?*

Question 2. *Given a piecewise testable language $L \subseteq \Sigma^*$, how to determine its height (the least h such that L belongs to PTL_h but not to PTL_{h-1})?*

Piecewise testable languages

Question 1. *Given a language $L \subseteq \Sigma^*$, how to decide whether or not L is piecewise testable?*

Question 2. *Given a piecewise testable language $L \subseteq \Sigma^*$, how to determine its height (the least h such that L belongs to PTL_h but not to PTL_{h-1})?*

Exercise. Is the language $\Sigma^* ab \Sigma^*$ ($a, b \in \Sigma$) piecewise testable?

Piecewise testable languages

Question 1. *Given a language $L \subseteq \Sigma^*$, how to decide whether or not L is piecewise testable?*

Question 2. *Given a piecewise testable language $L \subseteq \Sigma^*$, how to determine its height (the least h such that L belongs to PTL_h but not to PTL_{h-1})?*

Exercise. Is the language $\Sigma^* ab \Sigma^*$ ($a, b \in \Sigma$) piecewise testable?

☐ Yes ☐ No ☐ Don't know ☐ It depends

Syntactic monoids

For a language $L \subseteq \Sigma^*$ its *syntactic congruence* $\sim_L$ is defined by

$u \sim_L v$ if, for any $x, y \in \Sigma^*$, $xuy \in L \iff xvy \in L$.

Thus, u and v occur in L in the same contexts.

Syntactic monoids

For a language $L \subseteq \Sigma^*$ its *syntactic congruence* $\sim_L$ is defined by

$$u \sim_L v \quad \text{if, for any } x, y \in \Sigma^*,\ xuy \in L \iff xvy \in L.$$

Thus, u and v occur in L in the same contexts.

One can check that $\sim_L$ is the largest congruence on Σ^* for which L is a union of classes. The quotient monoid $M(L) = \Sigma^* / \sim_L$ is called the *syntactic monoid* of the language L.

Syntactic monoids

For a language $L \subseteq \Sigma^*$ its *syntactic congruence* $\sim_L$ is defined by

$$u \sim_L v \quad \text{if, for any } x, y \in \Sigma^*,\ xuy \in L \iff xvy \in L.$$

Thus, u and v occur in L in the same contexts.

One can check that $\sim_L$ is the largest congruence on Σ^* for which L is a union of classes. The quotient monoid $M(L) = \Sigma^* / \sim_L$ is called the *syntactic monoid* of the language L.

For a regular language L, the syntactic monoid $M(L)$ can be also defined as the *transition monoid* of the *minimal automaton* of L.

Syntactic monoids

Rather than formal definitions from the previous slide, the following crucial ideas are to be understood:

Syntactic monoids

Rather than formal definitions from the previous slide, the following crucial ideas are to be understood:

- For a regular language L, its syntactic monoid $M(L)$ is always finite (and vice versa) — this is Myhill's form of Kleene's theorem.

Syntactic monoids

Rather than formal definitions from the previous slide, the following crucial ideas are to be understood:

- For a regular language L, its syntactic monoid $M(L)$ is always finite (and vice versa) — this is Myhill's form of Kleene's theorem.

- The syntactic monoid $M(L)$ can be efficiently calculated whenever L is efficiently presented — say, by a regular expression or by a finite automaton.

Syntactic monoids

Rather than formal definitions from the previous slide, the following crucial ideas are to be understood:

- For a regular language L, its syntactic monoid $M(L)$ is always finite (and vice versa) — this is Myhill's form of Kleene's theorem.

- The syntactic monoid $M(L)$ can be efficiently calculated whenever L is efficiently presented — say, by a regular expression or by a finite automaton.

Thus, whenever L is "given", so is $M(L)$.

Simon's theorem

A monoid M is said to be *$\mathscr{J}$-trivial* if every principal ideal of M has a unique generator:

$$MsM = MtM \Longrightarrow s = t.$$

Simon's theorem

A monoid M is said to be *$\mathscr{J}$-trivial* if every principal ideal of M has a unique generator:

$$MsM = MtM \Longrightarrow s = t.$$

In different terms, being $\mathscr{J}$-trivial amounts to saying that the (*bilateral*) *divisibility relation*

$$s \leq_{\mathscr{J}} t \Longleftrightarrow s \in MtM$$

is an order relation on M.

Simon's theorem

A monoid M is said to be *$\mathscr{J}$-trivial* if every principal ideal of M has a unique generator:

$$MsM = MtM \Longrightarrow s = t.$$

In different terms, being $\mathscr{J}$-trivial amounts to saying that the *(bilateral) divisibility relation*

$$s \leq_{\mathscr{J}} t \Longleftrightarrow s \in MtM$$

is an order relation on M.

Theorem. (Imre Simon, 1972) *A language L is piecewise testable if and only if its syntactic monoid $M(L)$ is $\mathscr{J}$-trivial.*

Simon's theorem

	0	1	a	b	c	d
0	0	0	0	0	0	0
1	0	1	a	b	c	d
a	0	a	a	0	c	0
b	0	b	d	b	b	d
c	0	c	a	c	c	a
d	0	d	d	0	b	0

The monoid A_2^1

Simon's theorem

	0	1	a	b	c	d
0	0	0	0	0	0	0
1	0	1	a	b	c	d
a	0	a	a	0	c	0
b	0	b	d	b	b	d
c	0	c	a	c	c	a
d	0	d	d	0	b	0

$$aA_2^1 = \{0, a, c\}$$

Simon's theorem

	0	1	a	b	c	d
0	0	0	0	0	0	0
1	0	1	a	b	c	d
a	0	a	a	0	c	0
b	0	b	d	b	b	d
c	0	c	a	c	c	a
d	0	d	d	0	b	0

$$A_2^1 a A_2^1 = \{0, a, b, c, d\}$$

Simon's theorem

	0	1	a	b	c	d
0	0	0	0	0	0	0
1	0	1	a	b	c	d
a	0	a	a	0	c	0
b	0	b	d	b	b	d
c	0	c	a	c	c	a
d	0	d	d	0	b	0

$$A_2^1 a A_2^1 = \{0, a, b, c, d\}$$

Similarly one can verify that $A_2^1 b A_2^1 = \{0, a, b, c, d\}$ whence the monoid A_2^1 is not $\mathscr{J}$-trivial.

Simon's theorem

Example: solution to the above Exercise.

Simon's theorem

Example: solution to the above Exercise.

Exercise. Is the language $\Sigma^* ab \Sigma^*$ $(a, b \in \Sigma)$ piecewise testable?

Simon's theorem

Example: solution to the above Exercise.

Exercise. Is the language $\Sigma^* ab \Sigma^*$ $(a, b \in \Sigma)$ piecewise testable?

☐ Yes ☐ No ☐ Don't know ☐ It depends

Simon's theorem

Example: solution to the above Exercise.

Exercise. Is the language $\Sigma^* ab \Sigma^*$ $(a, b \in \Sigma)$ piecewise testable?

☐ Yes ☐ No ☐ Don't know ☑ It depends !

Simon's theorem

Example: solution to the above Exercise.

Exercise. Is the language $\Sigma^* ab \Sigma^*$ $(a, b \in \Sigma)$ piecewise testable?

☐ Yes ☐ No ☐ Don't know ☑ It depends !

If $\Sigma = \{a, b\}$, the language $\Sigma^* ab \Sigma^*$ is piecewise testable.

Simon's theorem

Example: solution to the above Exercise.

Exercise. Is the language $\Sigma^* ab \Sigma^*$ $(a, b \in \Sigma)$ piecewise testable?

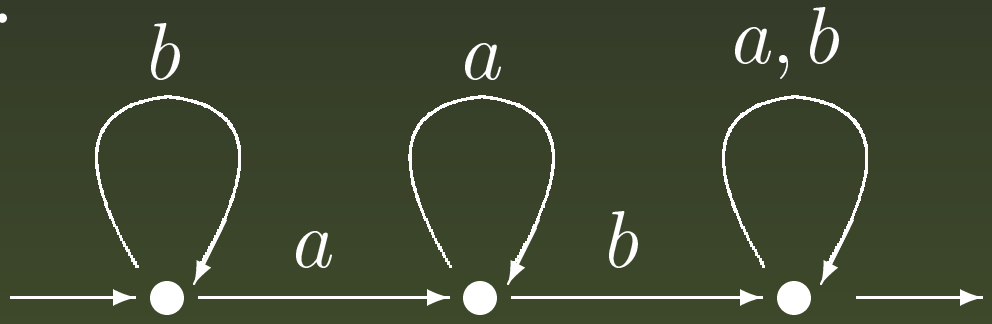
☐ Yes ☐ No ☐ Don't know ☑ It depends !

If $\Sigma = \{a, b\}$, the language $\Sigma^* ab \Sigma^*$ is piecewise testable.

If $\Sigma \supset \{a, b\}$, the language $\Sigma^* ab \Sigma^*$ is not piecewise testable.

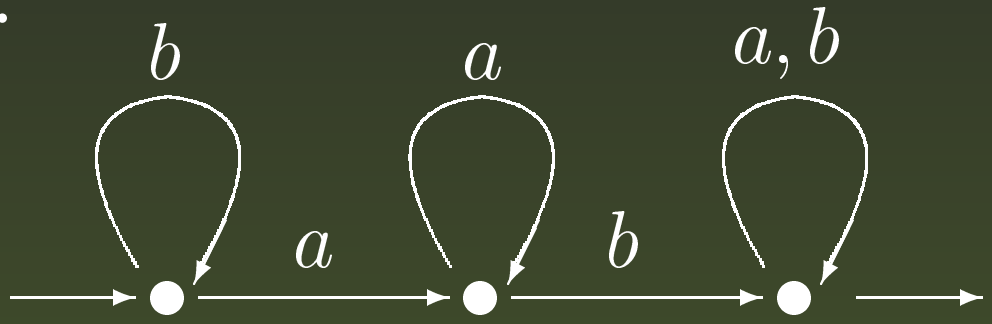
Simon's theorem

If $\Sigma = \{a, b\}$, the minimal automaton of $L = \Sigma^* ab \Sigma^*$ looks as follows:



Simon's theorem

If $\Sigma = \{a, b\}$, the minimal automaton of $L = \Sigma^* ab \Sigma^*$ looks as follows:



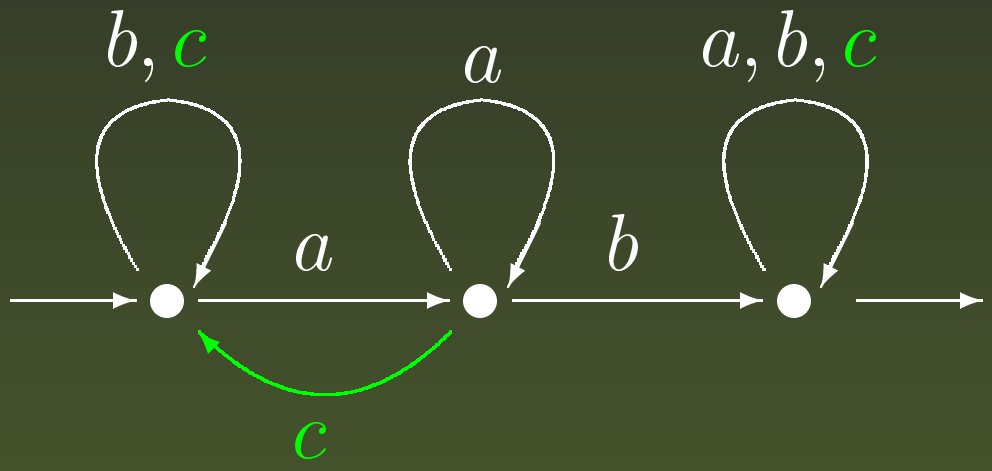
Using this, one readily calculates that

$$M(L) = \{0, 1, a, b, ba\}$$

subject to the relations $a^2 = a,\ b^2 = b, ab = 0$ and that $M(L)$ is $\mathscr{J}$-trivial. In fact, $L = \Sigma^* a \Sigma^* b \Sigma^*$.

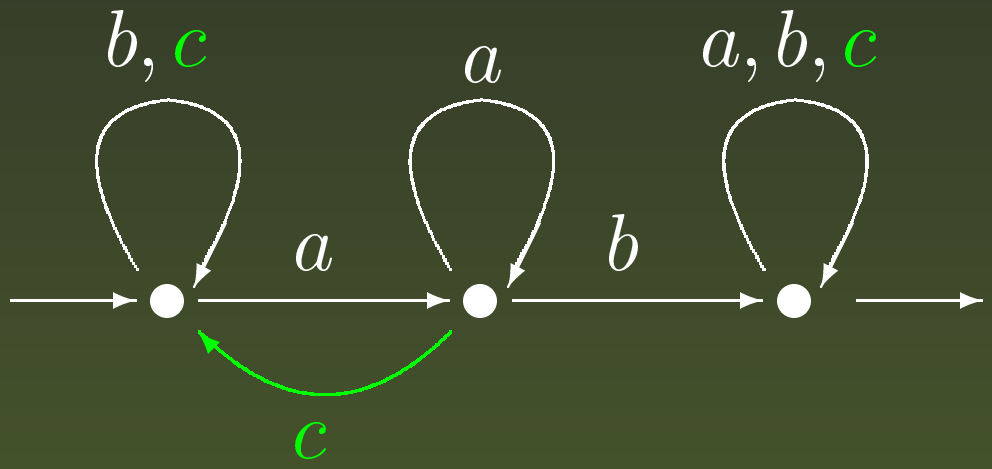
Simon's theorem

If $\Sigma = \{a, b, c\}$, the minimal automaton of $L = \Sigma^* ab \Sigma^*$ only slightly changes:



Simon's theorem

If $\Sigma = \{a, b, c\}$, the minimal automaton of $L = \Sigma^* ab \Sigma^*$ only slightly changes:



One gets

$$M(L) = \{0, 1, a, b, c, ba(= d)\} = A_2^1$$

and we already know that A_2^1 is not $\mathscr{J}$-trivial. Thus, the language L is not piecewise testable.

Simon's theorem

- Nice: relates a very natural combinatorial property to a very natural semigroup-theoretic property.

Simon's theorem

- Nice: relates a very natural combinatorial property to a very natural semigroup-theoretic property.
- Efficient: given a monoid M (by its Cayley table, say), one can easily (in $O(|M|^2)$ time) verify whether or not M is $\mathscr{J}$-trivial.

Simon's theorem

- Nice: relates a very natural combinatorial property to a very natural semigroup-theoretic property.
- Efficient: given a monoid M (by its Cayley table, say), one can easily (in $O(|M|^2)$ time) verify whether or not M is $\mathscr{J}$-trivial.
- Very efficient: There are polynomial time algorithms to verify if the syntactic monoid $M(L)$ is $\mathscr{J}$-trivial when presented the minimal automaton of L.

Simon's theorem

- Nice: relates a very natural combinatorial property to a very natural semigroup-theoretic property.
- Efficient: given a monoid M (by its Cayley table, say), one can easily (in $O(|M|^2)$ time) verify whether or not M is $\mathscr{J}$-trivial.
- Very efficient: There are polynomial time algorithms to verify if the syntactic monoid $M(L)$ is $\mathscr{J}$-trivial when presented the minimal automaton of L. Such a description of $M(L)$ is much more compact than the Cayley table — recall that the transition monoid of an automaton with n states may consist of as many as n^n elements!

Simon’s theorem

- Deep: a crossing where many ideas meet.

Simon's theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

Simon’s theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon’s original proof, 1972, 1975;

Simon's theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon's original proof, 1972, 1975;
- Model theory — Stern, 1985;

Simon's theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon's original proof, 1972, 1975;
- Model theory — Stern, 1985;
- Ordered monoids — Straubing and Thérien, 1988;

Simon's theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon's original proof, 1972, 1975;
- Model theory — Stern, 1985;
- Ordered monoids — Straubing and Thérien, 1988;
- Profinite topology — Almeida, 1990;

Simon's theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon's original proof, 1972, 1975;
- Model theory — Stern, 1985;
- Ordered monoids — Straubing and Thérien, 1988;
- Profinite topology — Almeida, 1990;
- Transformation semigroups — Higgins, 1997.

Recognizing height

A *pseudovariety* of finite monoids is a class of finite monoids closed under taking submonoids, morphic images and finite direct products.

Recognizing height

A *pseudovariety* of finite monoids is a class of finite monoids closed under taking submonoids, morphic images and finite direct products.

Fact (Eilenberg). *Each pseudovariety is generated by syntactic monoids it contains.*

Recognizing height

A *pseudovariety* of finite monoids is a class of finite monoids closed under taking submonoids, morphic images and finite direct products.

Fact (Eilenberg). *Each pseudovariety is generated by syntactic monoids it contains.*

Simon's theorem means that the pseudovariety **J** of all finite $\mathscr{J}$-trivial monoids is generated by syntactic monoids of piecewise testable languages.

Recognizing height

Let $\mathbf{J}_h$ denote the pseudovariety of finite monoids generated by the syntactic monoids of piecewise testable languages of height $\leq h$. We have

$$\mathbf{J}_1 \subset \mathbf{J}_2 \subset \mathbf{J}_3 \subset \cdots \subset \mathbf{J} = \bigcup_{h=1}^{\infty} \mathbf{J}_h$$

Recognizing height

Let $\mathbf{J}_h$ denote the pseudovariety of finite monoids generated by the syntactic monoids of piecewise testable languages of height $\leq h$. We have

$$\mathbf{J}_1 \subset \mathbf{J}_2 \subset \mathbf{J}_3 \subset \cdots \subset \mathbf{J} = \bigcup_{h=1}^{\infty} \mathbf{J}_h$$

Thus, the algebraic counterpart of Question 2 is:

Question 3. *Given a finite monoid M and a number h, how to determine whether or not M belongs to $\mathbf{J}_h$?*

Recognizing height

Let $\mathbf{J}_h$ denote the pseudovariety of finite monoids generated by the syntactic monoids of piecewise testable languages of height $\leq h$. We have

$$\mathbf{J}_1 \subset \mathbf{J}_2 \subset \mathbf{J}_3 \subset \cdots \subset \mathbf{J} = \bigcup_{h=1}^{\infty} \mathbf{J}_h$$

Thus, the algebraic counterpart of Question 2 is:

Question 3. *Given a finite monoid M and a number h, how to determine whether or not M belongs to $\mathbf{J}_h$?*

This is a typical instance of the **PMP** (Pseudovariety Membership Problem). The PMP has proved to systematically arise whenever one translates a "real world" (computer science) question into algebra.

Straubing's theorem

- $\mathscr{R}_n$ — the monoid of all reflexive binary relations on a set with n elements. It can be thought of as the monoid of all $n \times n$ matrices whose diagonal entries are 1 over the boolean semiring $\mathscr{B} = \langle \{0, 1\}; +, \cdot \rangle$.

Straubing's theorem

- $\mathscr{R}_n$ — the monoid of all reflexive binary relations on a set with n elements. It can be thought of as the monoid of all $n \times n$ matrices whose diagonal entries are 1 over the boolean semiring $\mathscr{B} = \langle \{0, 1\}; +, \cdot \rangle$.
- $\mathscr{U}_n$ — the submonoid of $\mathscr{R}_n$ consisting of upper triangular matrices.

Straubing's theorem

- $\mathscr{R}_n$ — the monoid of all reflexive binary relations on a set with n elements. It can be thought of as the monoid of all $n \times n$ matrices whose diagonal entries are 1 over the boolean semiring $\mathscr{B} = \langle \{0, 1\}; +, \cdot \rangle$.
- $\mathscr{U}_n$ — the submonoid of $\mathscr{R}_n$ consisting of upper triangular matrices.
- $\mathscr{C}_n$ — the monoid of all order preserving and extensive transformations of a chain with n elements.

Straubing's theorem

- $\mathscr{R}_n$ — the monoid of all reflexive binary relations on a set with n elements. It can be thought of as the monoid of all $n \times n$ matrices whose diagonal entries are 1 over the boolean semiring $\mathscr{B} = \langle \{0, 1\}; +, \cdot \rangle$.
- $\mathscr{U}_n$ — the submonoid of $\mathscr{R}_n$ consisting of upper triangular matrices.
- $\mathscr{C}_n$ — the monoid of all order preserving and extensive transformations of a chain with n elements.

  A transformation α of a chain $\langle Q, \leq \rangle$ is *order preserving* if $q \leq q'$ implies $q.\alpha \leq q'.\alpha$ for all $q, q' \in Q$ and *extensive* if $q \leq q.\alpha$ for every $q \in Q$.

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*
(ii) *M divides* (is a morphic image of a submonoid of) *$\mathscr{R}_n$ for some n;*

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*
(ii) *M divides* (is a morphic image of a submonoid of) *$\mathscr{R}_n$ for some n;*
(iii) *M divides $\mathscr{U}_n$ for some n;*

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*
(ii) *M divides* (is a morphic image of a submonoid of) *$\mathscr{R}_n$ for some n;*
(iii) *M divides $\mathscr{U}_n$ for some n;*
(iv) *M divides $\mathscr{C}_n$ for some n.*

Straubing's theorem

Theorem. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*
(ii) *M divides* (is a morphic image of a submonoid of) *$\mathscr{R}_n$ for some n;*
(iii) *M divides $\mathscr{U}_n$ for some n;*
(iv) *M divides $\mathscr{C}_n$ for some n.*

This looks as a quite innocent Cayley-type theorem but in fact the proof heavily depends on Simon's theorem, and moreover, it can be shown relatively easily that the two theorems are equivalent.

Straubing's theorem

Corollary. *Each of the three sequences* $\{\mathscr{R}_n\}$, $\{\mathscr{U}_n\}$ *and* $\{\mathscr{C}_n\}$ $(n = 2, 3, \ldots)$ *generates the pseudovariety* **J** *of all finite* $\mathscr{J}$*-trivial monoids.*

Straubing's theorem

Corollary. *Each of the three sequences* $\{\mathscr{R}_n\}$, $\{\mathscr{U}_n\}$ *and* $\{\mathscr{C}_n\}$ $(n = 2, 3, \ldots)$ *generates the pseudovariety* $\mathbf{J}$ *of all finite* $\mathscr{J}$*-trivial monoids.*

We thus have four stratifications for $\mathbf{J}$:

$$\mathbf{J}_1 \subset \mathbf{J}_2 \subset \mathbf{J}_3 \subset \cdots \subset \mathbf{J} = \bigcup_{h=1}^{\infty} \mathbf{J}_h$$

Straubing's theorem

Corollary. *Each of the three sequences* $\{\mathscr{R}_n\}$, $\{\mathscr{U}_n\}$ *and* $\{\mathscr{C}_n\}$ $(n = 2, 3, \ldots)$ *generates the pseudovariety* $\mathbf{J}$ *of all finite* $\mathscr{J}$*-trivial monoids.*

We thus have four stratifications for $\mathbf{J}$:

$$\mathbf{J}_1 \subset \mathbf{J}_2 \subset \mathbf{J}_3 \subset \cdots \subset \mathbf{J} = \bigcup_{h=1}^{\infty} \mathbf{J}_h$$

$$\begin{bmatrix} \operatorname{pvar} \mathscr{R}_2 \\ \operatorname{pvar} \mathscr{U}_2 \\ \operatorname{pvar} \mathscr{C}_2 \end{bmatrix} \subset \begin{bmatrix} \operatorname{pvar} \mathscr{R}_3 \\ \operatorname{pvar} \mathscr{U}_3 \\ \operatorname{pvar} \mathscr{C}_3 \end{bmatrix} \subset \cdots \subset \mathbf{J} = \bigcup_{n=1}^{\infty} \begin{bmatrix} \operatorname{pvar} \mathscr{R}_n \\ \operatorname{pvar} \mathscr{U}_n \\ \operatorname{pvar} \mathscr{C}_n \end{bmatrix}$$

Straubing's theorem: a refinement

Surprisingly enough, the four stratifications coincide:

Straubing's theorem: a refinement

Surprisingly enough, the four stratifications coincide:

Theorem. ($\sim$, 2004) *For every* $h = 1, 2, \ldots,$ *each of the monoids* $\mathscr{R}_{h+1}$, $\mathscr{U}_{h+1}$, $\mathscr{C}_{h+1}$ *generates the pseudovariety* $\mathbf{J}_h$.

Straubing's theorem: a refinement

Surprisingly enough, the four stratifications coincide:

Theorem. ($\sim$, 2004) *For every* $h = 1, 2, \ldots,$ *each of the monoids* $\mathscr{R}_{h+1}$, $\mathscr{U}_{h+1}$, $\mathscr{C}_{h+1}$ *generates the pseudovariety* $\mathbf{J}_h$.

Thus, for each h the pseudovariety $\mathbf{J}_h$ is generated by a single finite monoid. It easily follows from some basic universal algebra that the PMP for a (pseudo)variety generated by a single finite algebra is always decidable.

Straubing's theorem: a refinement

Surprisingly enough, the four stratifications coincide:

Theorem. ($\sim$, 2004) *For every $h = 1, 2, \ldots,$ each of the monoids $\mathscr{R}_{h+1}$, $\mathscr{U}_{h+1}$, $\mathscr{C}_{h+1}$ generates the pseudovariety $\mathbf{J}_h$.*

Thus, for each h the pseudovariety $\mathbf{J}_h$ is generated by a single finite monoid. It easily follows from some basic universal algebra that the PMP for a (pseudo)variety generated by a single finite algebra is always decidable.

Corollary. (Jean-Eric Pin, 1984) *For each $h = 1, 2, \ldots,$ the membership problem for the pseudovariety $\mathbf{J}_h$ is decidable, and hence, given a piecewise testable language, its height can be algorithmically determined.*

Recognizing height

Is this an efficient solution?

Recognizing height

Is this an efficient solution?

It doesn’t seem so — even if we allow ourselves the luxury of the language being presented by its syntactic monoid rather than by its minimal automaton.

Recognizing height

Is this an efficient solution?

It doesn't seem so — even if we allow ourselves the luxury of the language being presented by its syntactic monoid rather than by its minimal automaton.

Indeed, if $|A| = n$ and $|B| = m$, then the only known time bound for the algorithm that recognizes whether or not B belongs to $\mathrm{pvar}\, A$ is $n^{(n^m)}$ — so requires doubly exponential time (as a function of $|B|$).

Recognizing height

Is this an efficient solution?

It doesn't seem so — even if we allow ourselves the luxury of the language being presented by its syntactic monoid rather than by its minimal automaton.

Indeed, if $|A| = n$ and $|B| = m$, then the only known time bound for the algorithm that recognizes whether or not B belongs to $\mathrm{pvar}\, A$ is $n^{(n^m)}$ — so requires doubly exponential time (as a function of $|B|$).

Can we do better?

Recognizing height via identities

Lemma. (Eilenberg-Schützenberger, 1976) *Every pseudovariety generated by a single finite monoid is equational, that is, it consists precisely of finite monoids satisfying a certain system of usual monoid identities.*

Recognizing height via identities

Lemma. (Eilenberg-Schützenberger, 1976) *Every pseudovariety generated by a single finite monoid is equational, that is, it consists precisely of finite monoids satisfying a certain system of usual monoid identities.*
A monoid M is said to be *finitely based* if all identities holding in M follow from a finite set of such identities (an *identity basis* of M).

Recognizing height via identities

Lemma. (Eilenberg-Schützenberger, 1976) *Every pseudovariety generated by a single finite monoid is equational, that is, it consists precisely of finite monoids satisfying a certain system of usual monoid identities.*
A monoid M is said to be *finitely based* if all identities holding in M follow from a finite set of such identities (an *identity basis* of M). If we know a finite identity basis Φ of a monoid M then we can use it to efficiently decide the membership in $\mathrm{pvar}\, M$.

Recognizing height via identities

Lemma. (Eilenberg-Schützenberger, 1976) *Every pseudovariety generated by a single finite monoid is equational, that is, it consists precisely of finite monoids satisfying a certain system of usual monoid identities.*
A monoid M is said to be *finitely based* if all identities holding in M follow from a finite set of such identities (an *identity basis* of M). If we know a finite identity basis Φ of a monoid M then we can use it to efficiently decide the membership in $\mathrm{pvar}\, M$. Indeed, given a finite monoid N, we can simply check if it satisfies each identity in Φ, and this requires polynomial time (as a function of $|N|$).

Recognizing height via identities

Example: The pseudovariety $\mathbf{J}_1$ of all $\mathscr{J}$-trivial monoids of height 1 is generated by the monoid $\mathscr{C}_2$ of extensive endomorphisms of the chain $1 < 2$.

Recognizing height via identities

Example: The pseudovariety $\mathbf{J}_1$ of all $\mathscr{J}$-trivial monoids of height 1 is generated by the monoid $\mathscr{C}_2$ of extensive endomorphisms of the chain $1 < 2$. Therefore $\mathscr{C}_2$ is nothing but the 2-element semilattice.

Recognizing height via identities

Example: The pseudovariety $\mathbf{J}_1$ of all $\mathscr{J}$-trivial monoids of height 1 is generated by the monoid $\mathscr{C}_2$ of extensive endomorphisms of the chain $1 < 2$. Therefore $\mathscr{C}_2$ is nothing but the 2-element semilattice. It is obvious that its identity basis consists of the two identities: the commutative law $xy = yx$ and the idempotency law $x^2 = x$.

Recognizing height via identities

Example: The pseudovariety $\mathbf{J}_1$ of all $\mathscr{J}$-trivial monoids of height 1 is generated by the monoid $\mathscr{C}_2$ of extensive endomorphisms of the chain $1 < 2$. Therefore $\mathscr{C}_2$ is nothing but the 2-element semilattice. It is obvious that its identity basis consists of the two identities: the commutative law $xy = yx$ and the idempotency law $x^2 = x$. Thus, in order to check whether or not a given language L is piecewise testable of height 1, it suffices to verify if its syntactic monoid $M(L)$ is commutative and idempotent.

Recognizing height via identities

Does this approach apply to heights $2, 3, 4, \ldots$?

Recognizing height via identities

Does this approach apply to heights $2, 3, 4, \ldots$?

Theorem. ($\sim$, 2004) a) *The identities* $x^2 = x$, $xy = yx$ *form an identity basis of the monoid* $\mathscr{C}_2$.

Recognizing height via identities

Does this approach apply to heights $2, 3, 4, \ldots$?

Theorem. ($\sim$, 2004) a) *The identities* $x^2 = x$, $xy = yx$ *form an identity basis of the monoid* $\mathscr{C}_2$.
b) *The identities* $xyxzx = xyzx$, $(xy)^2 = (yx)^2$ *form an identity basis of the monoid* $\mathscr{C}_3$.

Recognizing height via identities

Does this approach apply to heights $2, 3, 4, \ldots$?

Theorem. ($\sim$, 2004) a) *The identities* $x^2 = x$, $xy = yx$ *form an identity basis of the monoid* $\mathscr{C}_2$.
b) *The identities* $xyxzx = xyzx$, $(xy)^2 = (yx)^2$ *form an identity basis of the monoid* $\mathscr{C}_3$.
c) *The identities* $xyx^2zx = xyxzx$, $xyzx^2tx = xyxzx^2tx$, $xyx^2ztx = xyx^2zxtx$, $(xy)^3 = (yx)^3$ *form an identity basis of the monoid* $\mathscr{C}_4$.

Recognizing height via identities

Does this approach apply to heights $2, 3, 4, \ldots$?

Theorem. ($\sim$, 2004) a) *The identities* $x^2 = x$, $xy = yx$ *form an identity basis of the monoid* $\mathscr{C}_2$.
b) *The identities* $xyxzx = xyzx$, $(xy)^2 = (yx)^2$ *form an identity basis of the monoid* $\mathscr{C}_3$.
c) *The identities* $xyx^2zx = xyxzx$, $xyzx^2tx = xyxzx^2tx$, $xyx^2ztx = xyx^2zxtx$, $(xy)^3 = (yx)^3$ *form an identity basis of the monoid* $\mathscr{C}_4$.
d) *The monoids* $\mathscr{C}_n$ *with* $n > 4$ *are nonfinitely based.*

Recognizing height via identities

Thus, there is an efficient algorithm to check if a given piecewise testable language can be recognized by a hydra automaton with 1, 2 or 3 heads, but this approach fails for larger numbers of heads.

Recognizing height via identities

Thus, there is an efficient algorithm to check if a given piecewise testable language can be recognized by a hydra automaton with 1, 2 or 3 heads, but this approach fails for larger numbers of heads.

One may conclude that the optimal number of heads is equal to 3!

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and “practical” importance

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and “practical” importance $\Rightarrow$ try to find upper and lower bounds for it!

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and "practical" importance $\Rightarrow$ try to find upper and lower bounds for it!
- A positive answer to some instance of FBP gives an efficient algorithm for the corresponding instance of PMP

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and "practical" importance $\Rightarrow$ try to find upper and lower bounds for it!
- A positive answer to some instance of FBP gives an efficient algorithm for the corresponding instance of PMP $\Rightarrow$ systematically investigate FBP!

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and "practical" importance $\Rightarrow$ try to find upper and lower bounds for it!
- A positive answer to some instance of FBP gives an efficient algorithm for the corresponding instance of PMP $\Rightarrow$ systematically investigate FBP!
- For many important instances, the answer to FBP is negative

Conclusions and directions

- Computational complexity of (some instances of) PMP is of interest and "practical" importance $\Rightarrow$ try to find upper and lower bounds for it!
- A positive answer to some instance of FBP gives an efficient algorithm for the corresponding instance of PMP $\Rightarrow$ systematically investigate FBP!
- For many important instances, the answer to FBP is negative $\Rightarrow$ develop some "equational" approach to PMP for nonfinitely based pseudovarieties!