Symmetries and Ordered Structures

Hattingen, July 26–31, 2003

Endomorphisms of linear orders: the finite axiomatization problem and complexity of membership testing

Mikhail Volkov

Ural State University, Ekaterinburg, Russia

Motivation

Everybody knows that linear orders are important . . .

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

But we are going to speak about linear orders which are neither countably infinite nor dense and possess extremes

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

But we are going to speak about linear orders which are neither countably infinite nor dense and possess extremes . . . because they are finite.

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

But we are going to speak about linear orders which are neither countably infinite nor dense and possess extremes . . . because they are finite.

They even have no automorphisms to study!

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

But we are going to speak about linear orders which are neither countably infinite nor dense and possess extremes . . . because they are finite.

They even have no automorphisms to study! Still they have got some endomorphisms.

Motivation

Everybody knows that linear orders are important . . . whenever they are countably infinite, dense and without extreme elements.

But we are going to speak about linear orders which are neither countably infinite nor dense and possess extremes . . . because they are finite.

They even have no automorphisms to study! Still they have got some endomorphisms.

Can finite linear orders and their endomorphisms be of any interest and any importance?

Motivation

Instead of answering this question, we look in some detail at a classic application of semigroup theory to computer science:

Motivation

Instead of answering this question, we look in some detail at a classic application of semigroup theory to computer science:

Imre Simon's characterization of *piecewise testable languages*

Motivation

Instead of answering this question, 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

Instead of answering this question, 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

Instead of answering this question, 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

Instead of answering this question, 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, including those with model theory;

Motivation

Instead of answering this question, 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, including those with model theory;
- 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

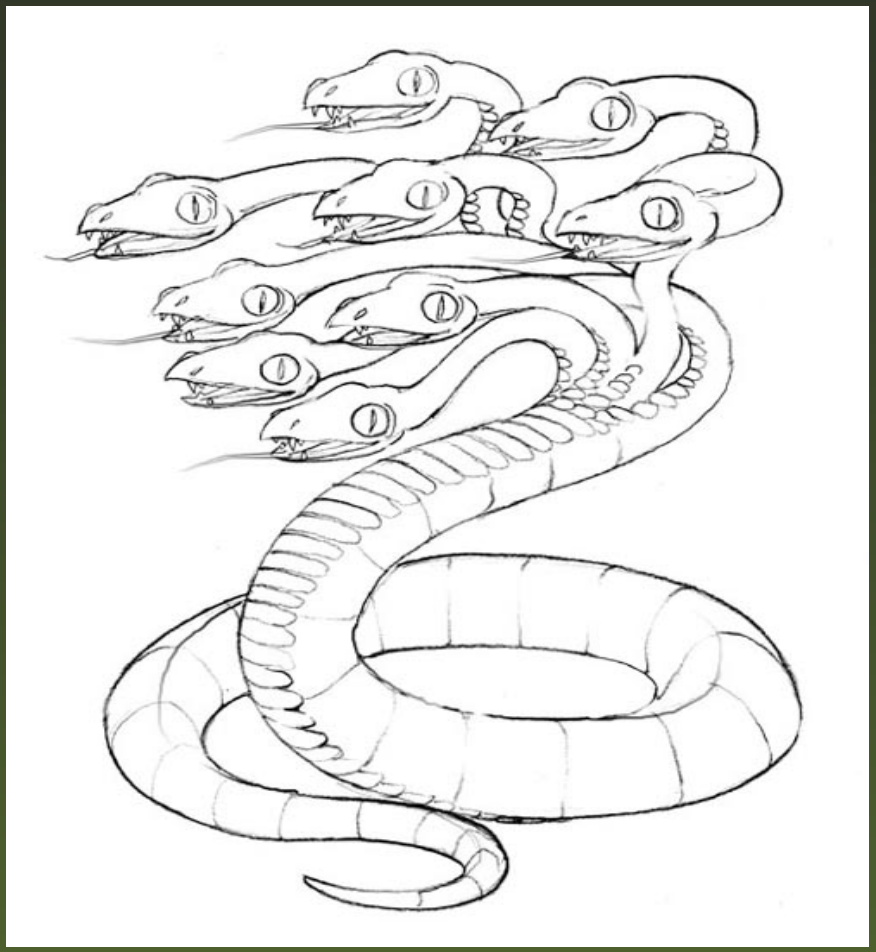


Figure 1: A 9-head hydra

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

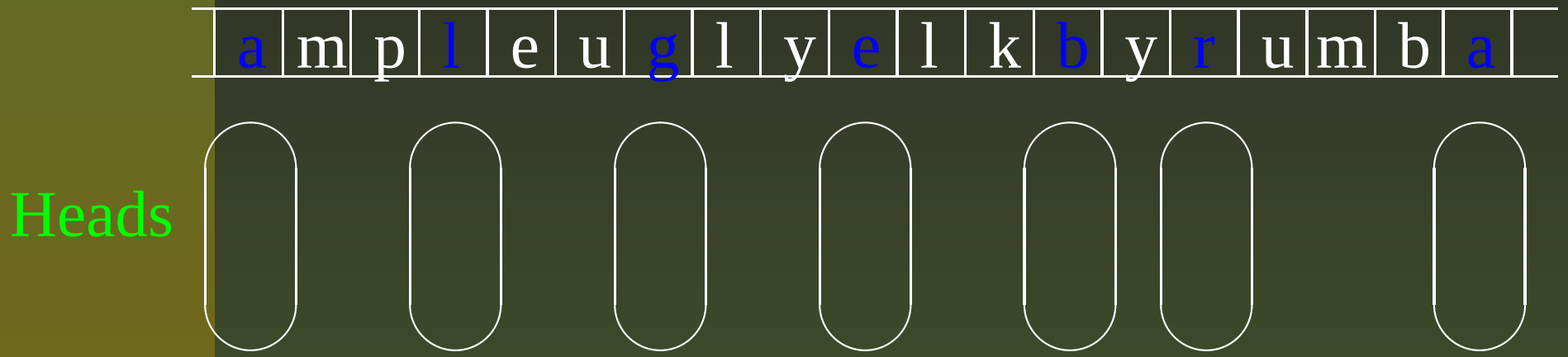


Figure 2: A 7-head hydra automaton

Hydra automata

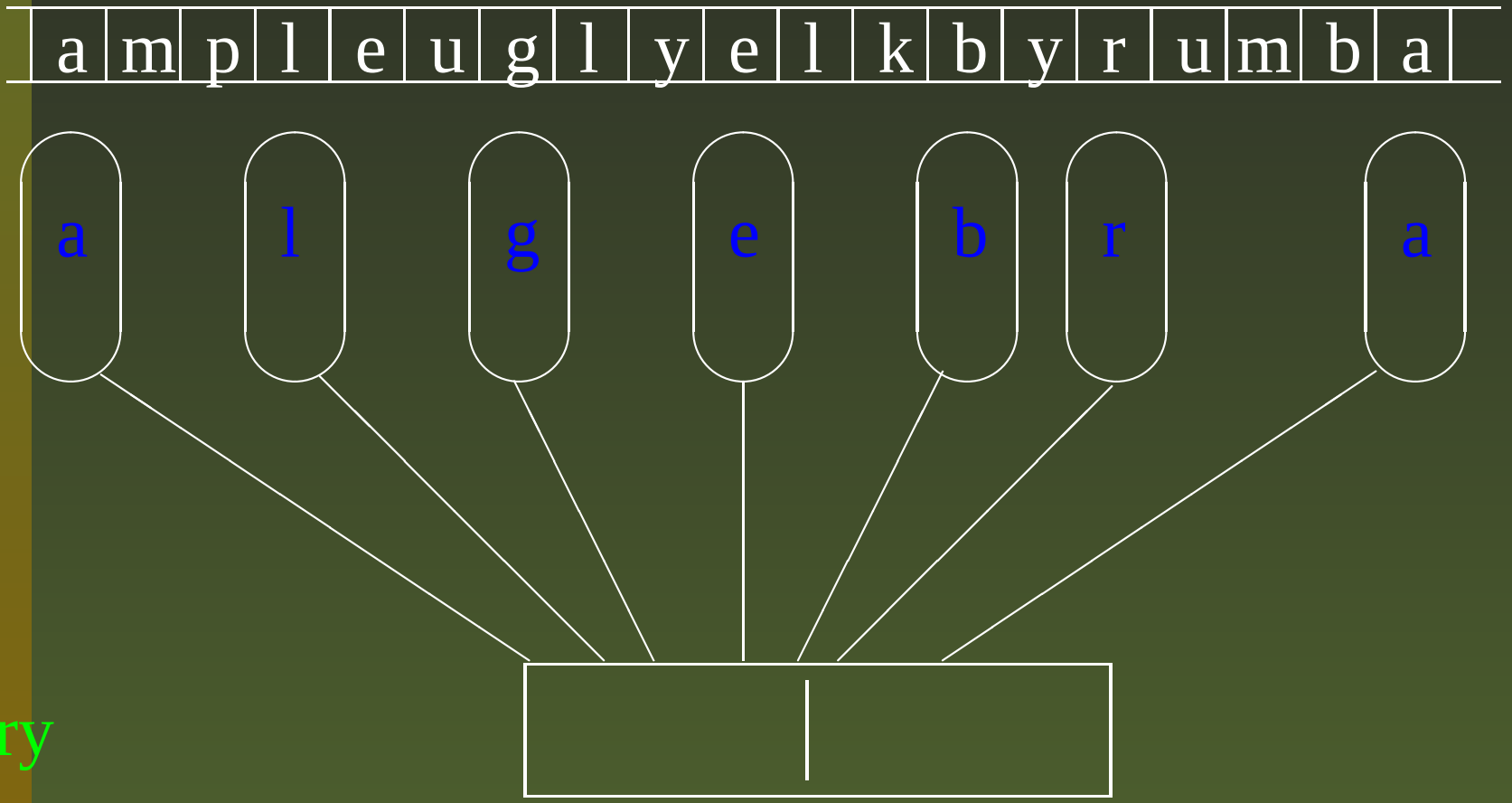


Figure 2: A 7-head hydra automaton

Hydra automata

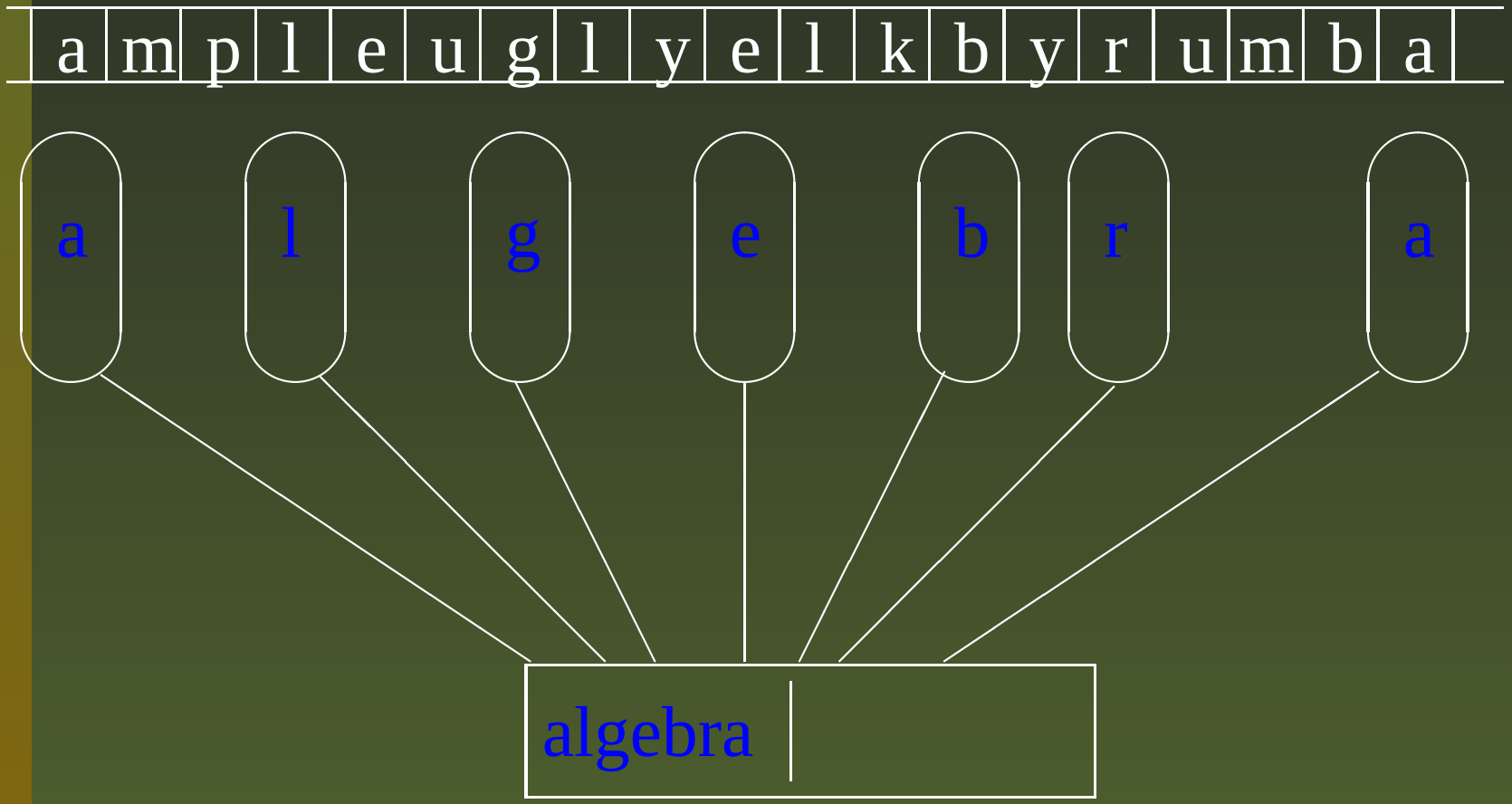


Figure 2: A 7-head hydra automaton

Hydra automata

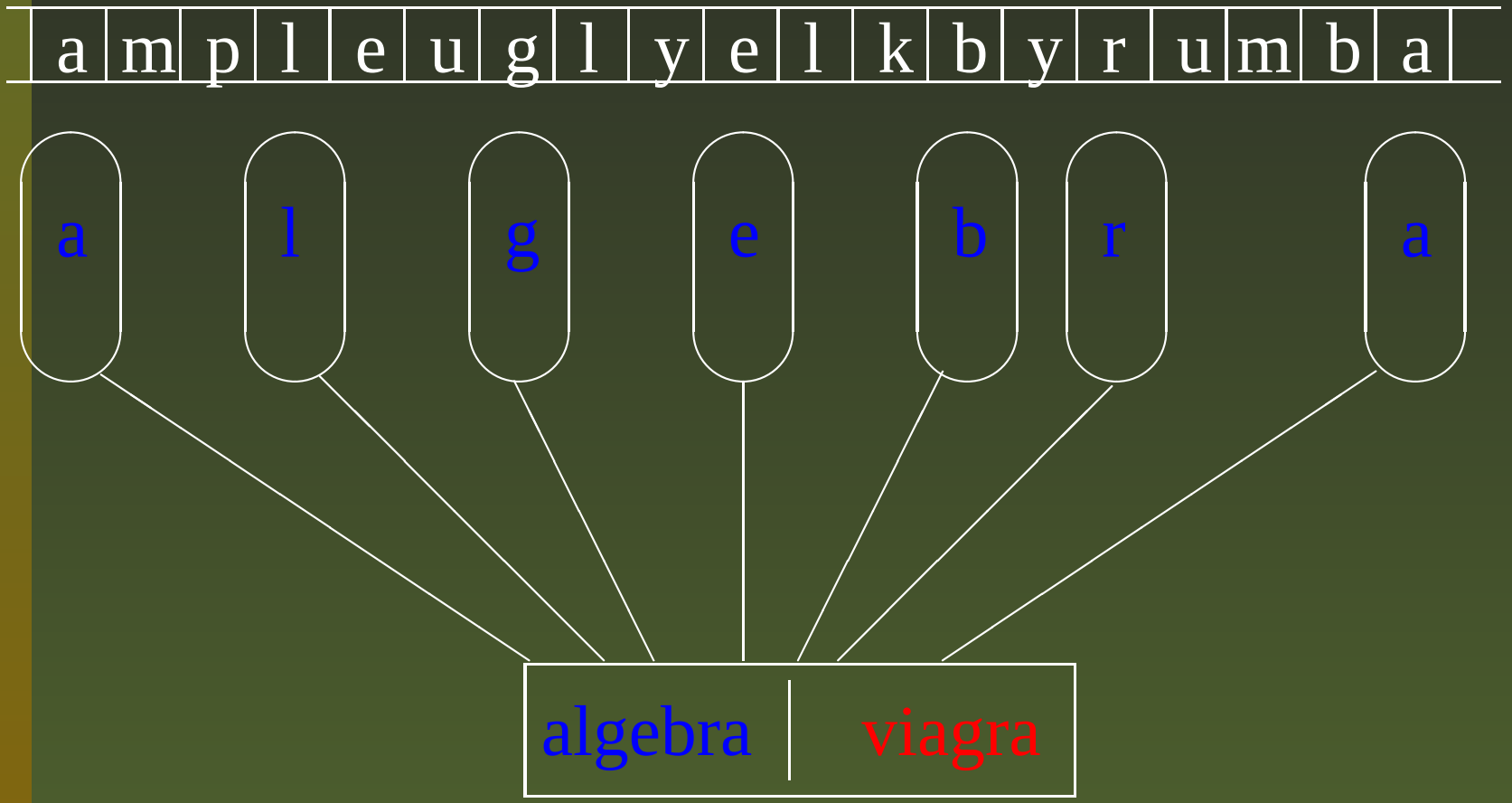


Figure 2: A 7-head hydra automaton

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

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 1. (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

- 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 vs. Schützenberger

Compare with Schützenberger's theorem (1966) that provides an algebraic characterization of star-free languages: *a language L can be defined by a star-free expression* (that is, involving only Boolean operations and products but not Kleene's star) *if and only if the syntactic monoid $M(L)$ has only trivial subgroups*.

Simon vs. Schützenberger

Compare with Schützenberger's theorem (1966) that provides an algebraic characterization of star-free languages: *a language L can be defined by a star-free expression* (that is, involving only Boolean operations and products but not Kleene's star) *if and only if the syntactic monoid $M(L)$ has only trivial subgroups*.

Again a very natural language property is related to a natural semigroup property that can be verified in time $O(|M(L)|^2)$.

Simon vs. Schützenberger

Compare with Schützenberger's theorem (1966) that provides an algebraic characterization of star-free languages: *a language L can be defined by a star-free expression* (that is, involving only Boolean operations and products but not Kleene's star) *if and only if the syntactic monoid $M(L)$ has only trivial subgroups*.

Again a very natural language property is related to a natural semigroup property that can be verified in time $O(|M(L)|^2)$. On the other hand, the problem of deciding whether or not $M(L)$ has only trivial subgroups from the minimal automaton of L is PSPACE-complete!

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;
- Endomorphisms of linear orders — 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

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

— Simon's hierarchy of $\mathcal{J}$-trivial monoids.

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

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

— Simon's hierarchy of $\mathscr{J}$-trivial monoids.

Thus, the algebraic counterpart of Question 2 is:

Question 3. *Given a finite monoid M and a positive integer 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

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

— Simon's hierarchy of $\mathscr{J}$-trivial monoids.

Thus, the algebraic counterpart of Question 2 is:

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

Endomorphisms of linear orders

Now finite linear orders and their endomorphisms come into the play.

Endomorphisms of linear orders

Now finite linear orders and their endomorphisms come into the play.

Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

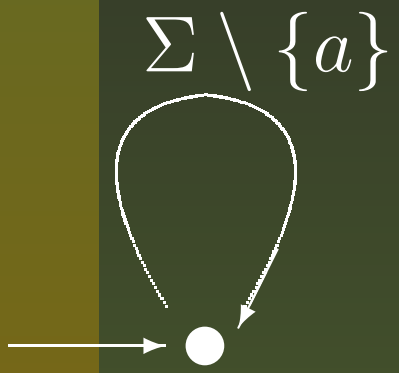
Endomorphisms of linear orders

Now finite linear orders and their endomorphisms come into the play.

Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

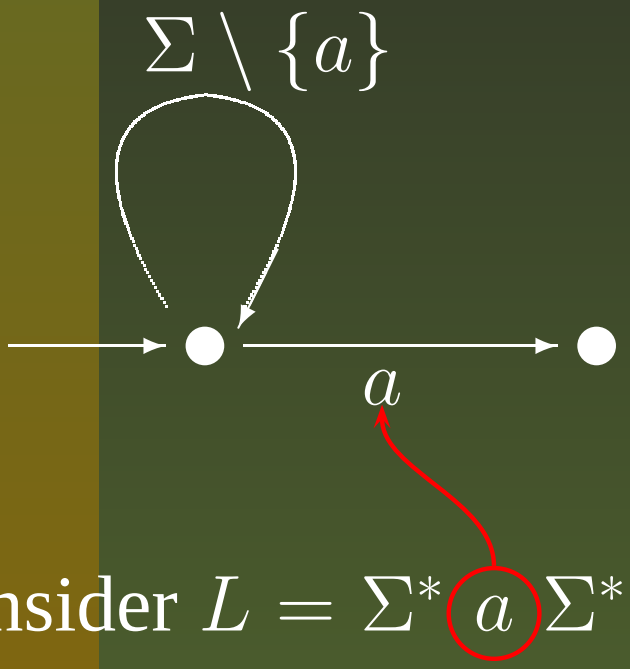
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \, a \, \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

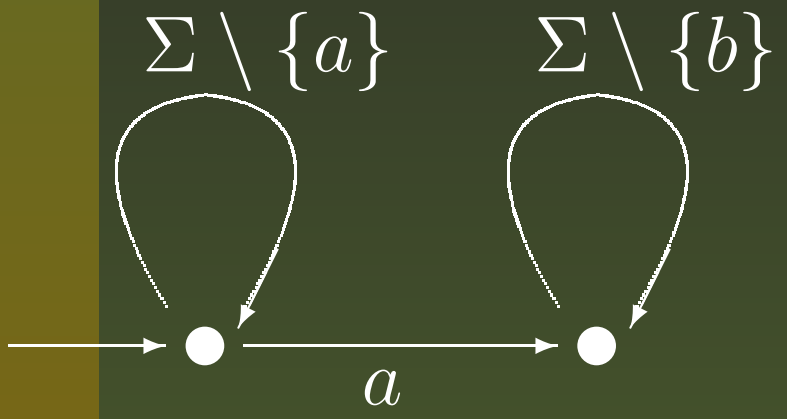
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

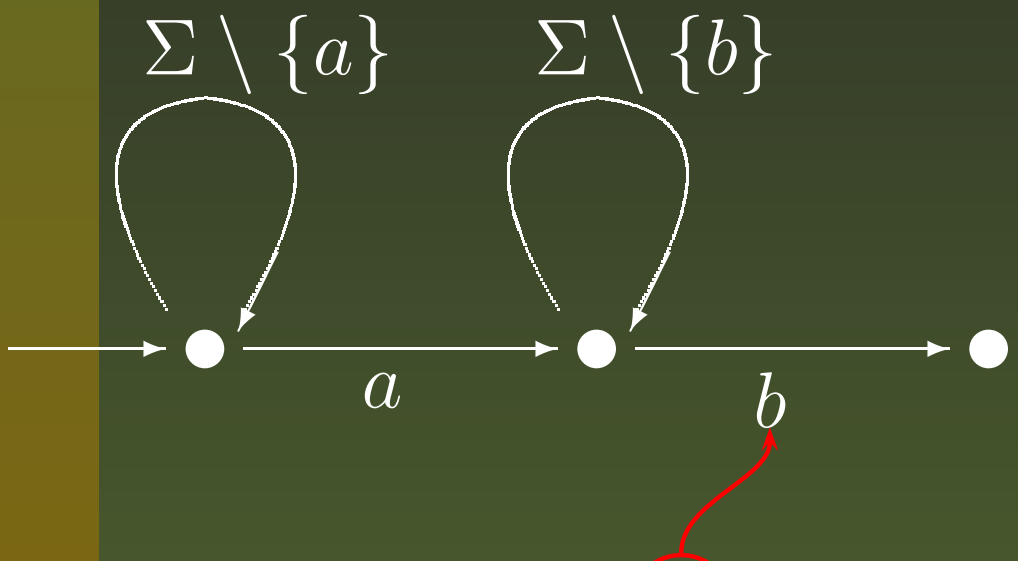
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

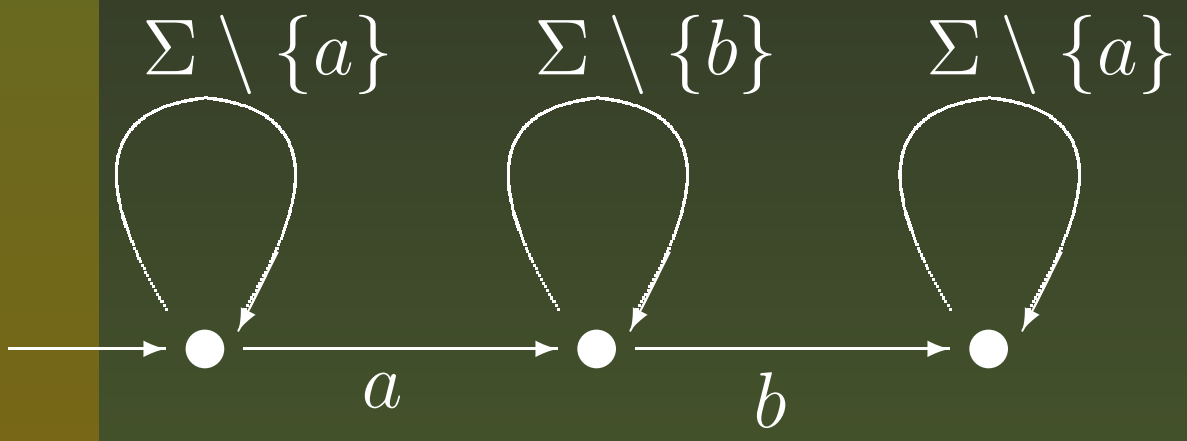
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

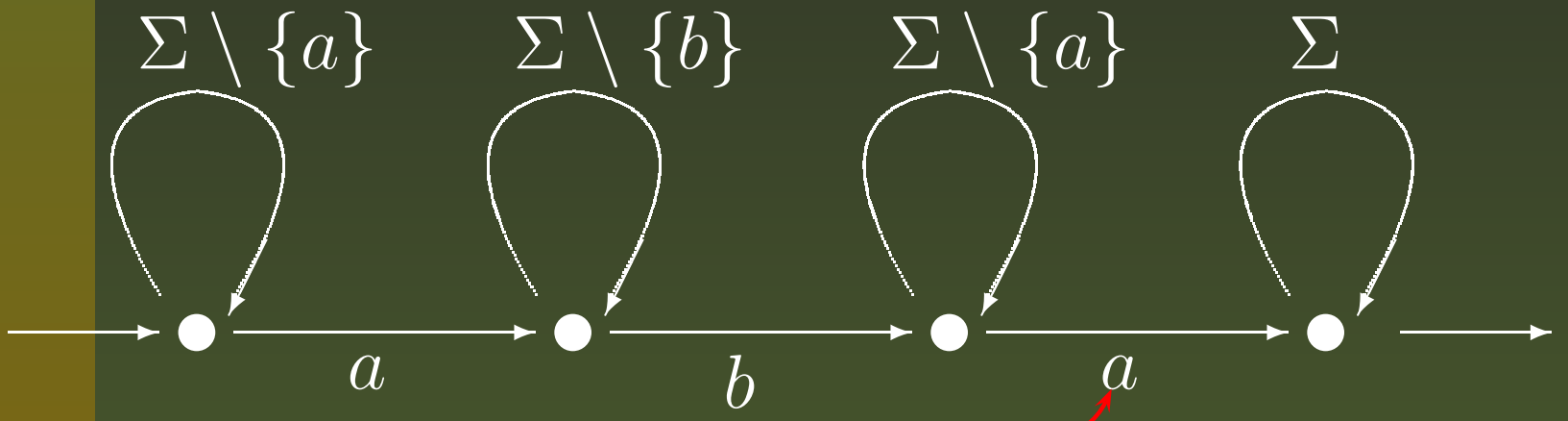
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

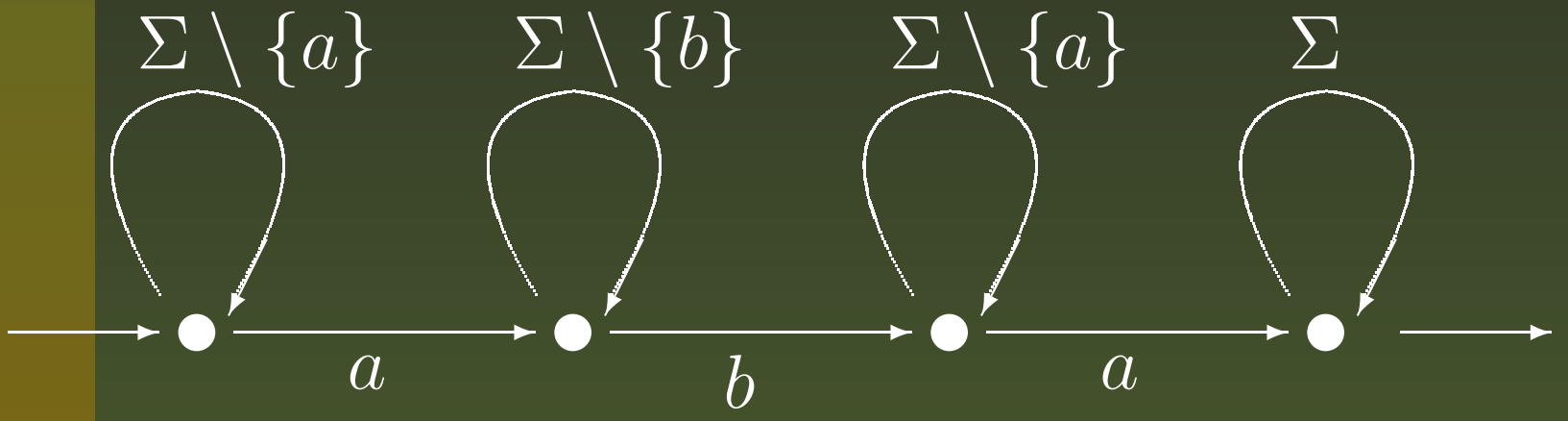
Now finite linear orders and their endomorphisms come into the play.



Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

Now finite linear orders and their endomorphisms come into the play.



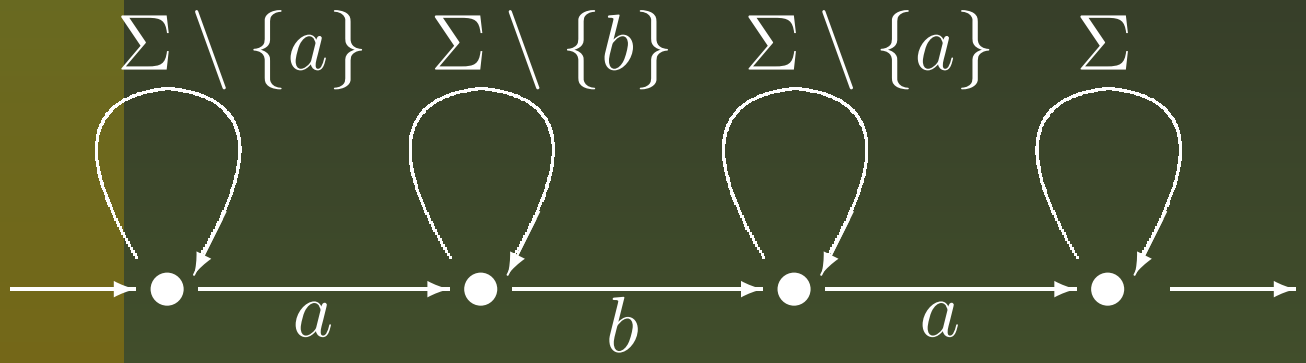
Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Endomorphisms of linear orders

Now impose a linear order on the state set of the automaton we built:

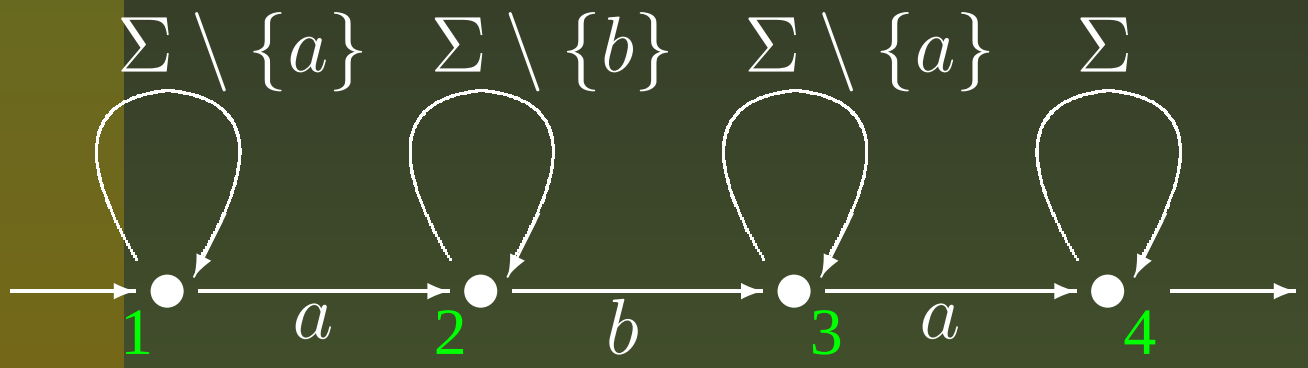
Endomorphisms of linear orders

Now impose a linear order on the state set of the automaton we built:



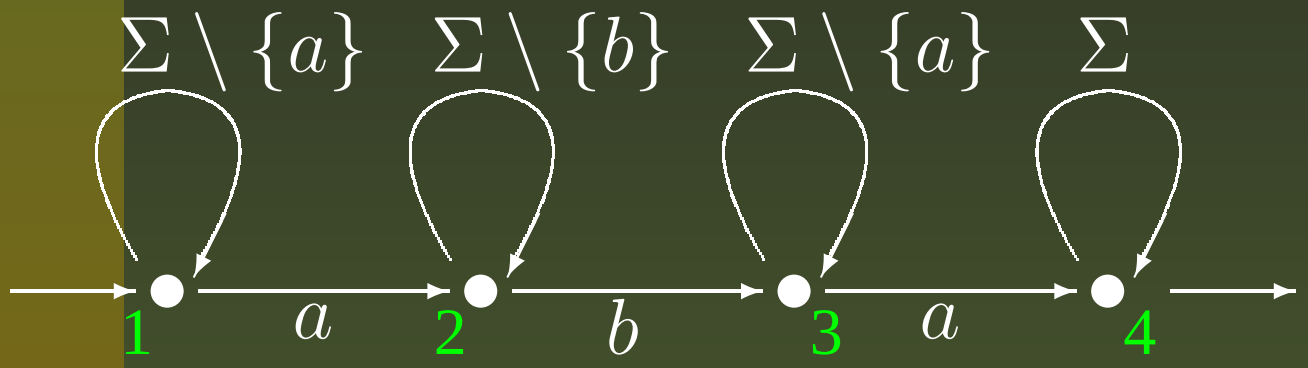
Endomorphisms of linear orders

Now impose a linear order on the state set of the automaton we built:



Endomorphisms of linear orders

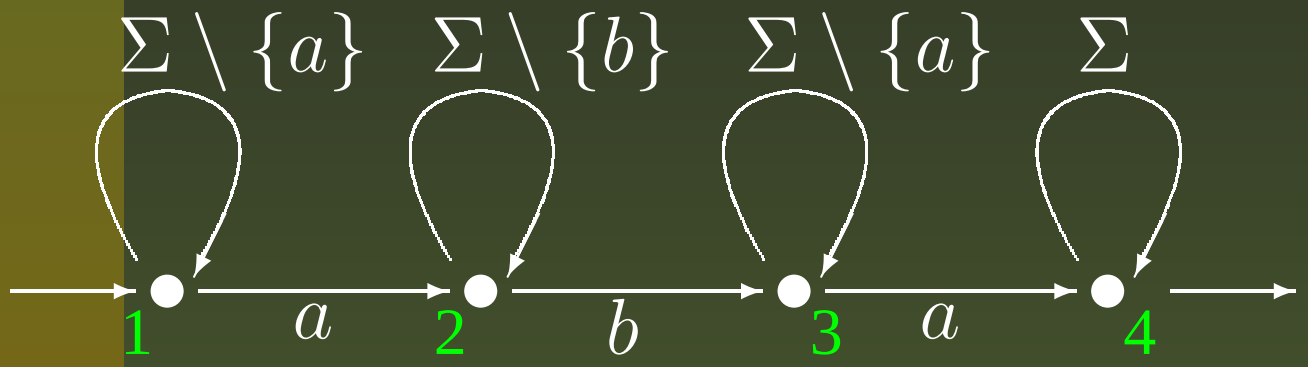
Now impose a linear order on the state set of the automaton we built:



With respect to this order the transformations induced by the letters are extensive endomorphisms.

Endomorphisms of linear orders

Now impose a linear order on the state set of the automaton we built:

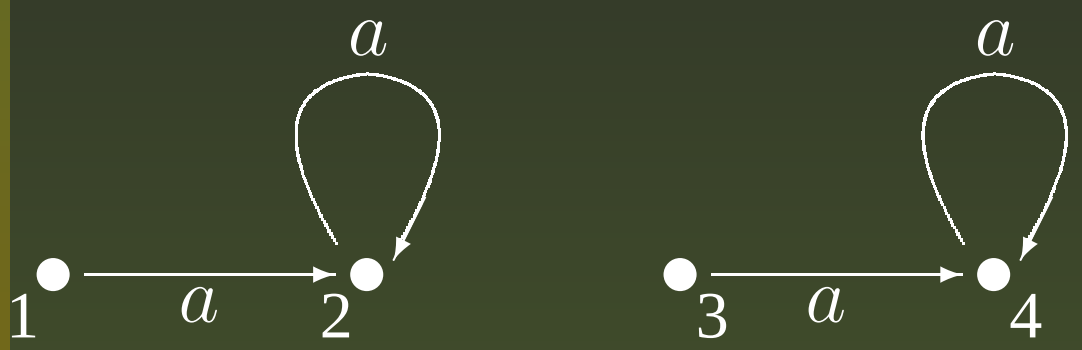


With respect to this order the transformations induced by the letters are extensive endomorphisms.

Recall that a transformation α of a linear order is said to be *extensive* if $i \leq i\alpha$ for every i in the domain of α.

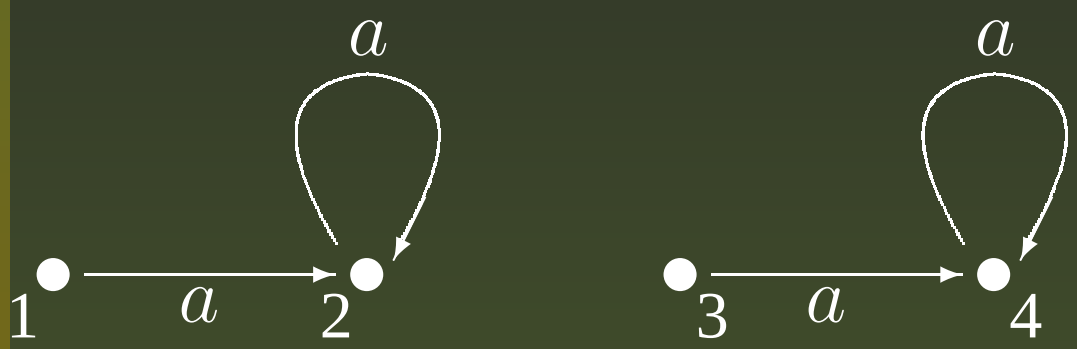
Endomorphisms of linear orders

For instance, this is the one induced by a:

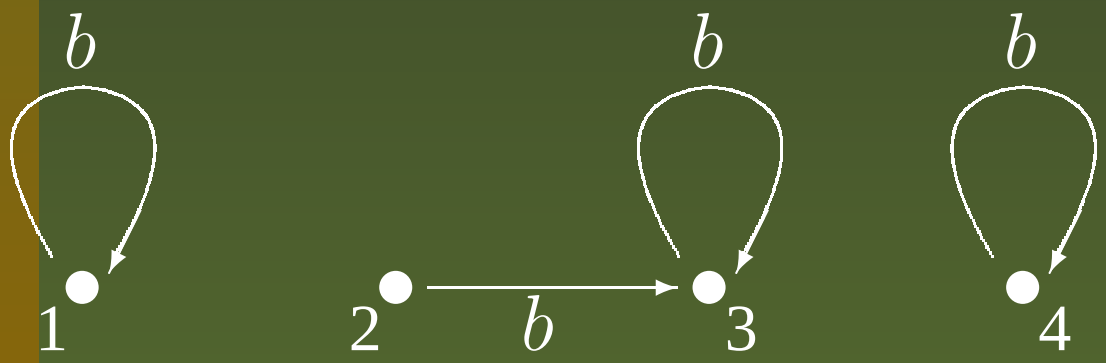


Endomorphisms of linear orders

For instance, this is the one induced by a:



And this is the action of b:



Endomorphisms of linear orders

Hence the transition monoid of the deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of extensive endomorphisms of the chain $1 < 2 < 3 < 4$.

Endomorphisms of linear orders

Hence the transition monoid of the deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of extensive endomorphisms of the chain $1 < 2 < 3 < 4$.

Let $\mathscr{C}_n$ denote the monoid of all extensive endomorphisms of the chain $1 < 2 < \cdots < n$.

Endomorphisms of linear orders

Hence the transition monoid of the deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of extensive endomorphisms of the chain $1 < 2 < 3 < 4$.

Let $\mathscr{C}_n$ denote the monoid of all extensive endomorphisms of the chain $1 < 2 < \cdots < n$.

In general, when starting with the language

$$L = \Sigma^* a_1 \Sigma^* a_2 \cdots \Sigma^* a_h \Sigma^*,$$

we end up in the monoid $\mathscr{C}_{h+1}$.

Endomorphisms of linear orders

Hence the transition monoid of the deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of extensive endomorphisms of the chain $1 < 2 < 3 < 4$.

Let $\mathscr{C}_n$ denote the monoid of all extensive endomorphisms of the chain $1 < 2 < \cdots < n$.

In general, when starting with the language

$$L = \Sigma^* a_1 \Sigma^* a_2 \cdots \Sigma^* a_h \Sigma^*,$$

we end up in the monoid $\mathscr{C}_{h+1}$. Therefore the pseudovariety $\mathbf{J}_h$ is contained in the pseudovariety $\operatorname{pvar} \mathscr{C}_{h+1}$ generated by $\mathscr{C}_{h+1}$.

Endomorphisms of linear orders

Theorem 2. ($\sim$, 2003) *For every $h = 1, 2, \ldots,$ the monoid $\mathscr{C}_{h+1}$ generates the pseudovariety $\mathbf{J}_h$.*

Endomorphisms of linear orders

Theorem 2. ($\sim$, 2003) *For every $h = 1, 2, \ldots,$ the monoid $\mathscr{C}_{h+1}$ 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 membership in a (pseudo)variety generated by a single finite algebra is always decidable.

Endomorphisms of linear orders

Theorem 2. ($\sim$, 2003) *For every $h = 1, 2, \ldots,$ the monoid $\mathscr{C}_{h+1}$ 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 membership in 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 3. ($\sim$, 2003) 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 3. ($\sim$, 2003) 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 3. ($\sim$, 2003) 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 3. ($\sim$, 2003) 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!

Conclusion

What we have seen is just a sample from a rather big area in which natural combinatorial properties of languages lead to certain classes of transformation monoids of finite linear orders (endomorphisms, partial endomorphisms, partial automorphisms, extensive mappings, etc).

Conclusion

What we have seen is just a sample from a rather big area in which natural combinatorial properties of languages lead to certain classes of transformation monoids of finite linear orders (endomorphisms, partial endomorphisms, partial automorphisms, extensive mappings, etc). In each case we encounter two problems: decidability of membership and finite axiomatizability for equational theory.

Conclusion

What we have seen is just a sample from a rather big area in which natural combinatorial properties of languages lead to certain classes of transformation monoids of finite linear orders (endomorphisms, partial endomorphisms, partial automorphisms, extensive mappings, etc). In each case we encounter two problems: decidability of membership and finite axiomatizability for equational theory. By now we have solved the finite axiomatizability problem for almost all cases but the membership problem remains open, say, for the pseudovariety generated by all endomorphism monoids of finite chains.

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

- being *total* (everywhere defined);

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

- being *total* (everywhere defined);
- being *injective*;

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

- being *total* (everywhere defined);
- being *injective*;
- being *order preserving*, that is, endomorphic;

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

- being *total* (everywhere defined);
- being *injective*;
- being *order preserving*, that is, endomorphic;
- being *extensive*.

Conclusion

We fix an integer $n > 2$ and consider the following 4 properties of partial transformations of $\{1, 2, \ldots, n\}$:

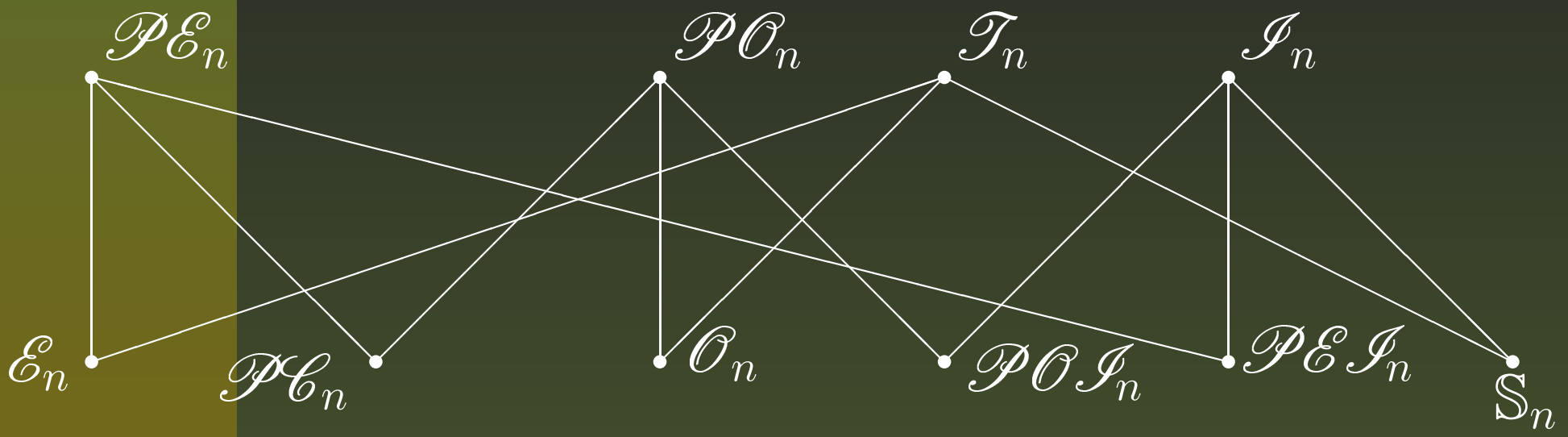
- being *total* (everywhere defined);
- being *injective*;
- being *order preserving*, that is, endomorphic;
- being *extensive*.

These properties define 4 monoids of partial transformations of $\{1, 2, \ldots, n\}$: $\mathscr{T}_n$, $\mathscr{I}_n$, $\mathscr{PO}_n$, $\mathscr{PE}_n$.

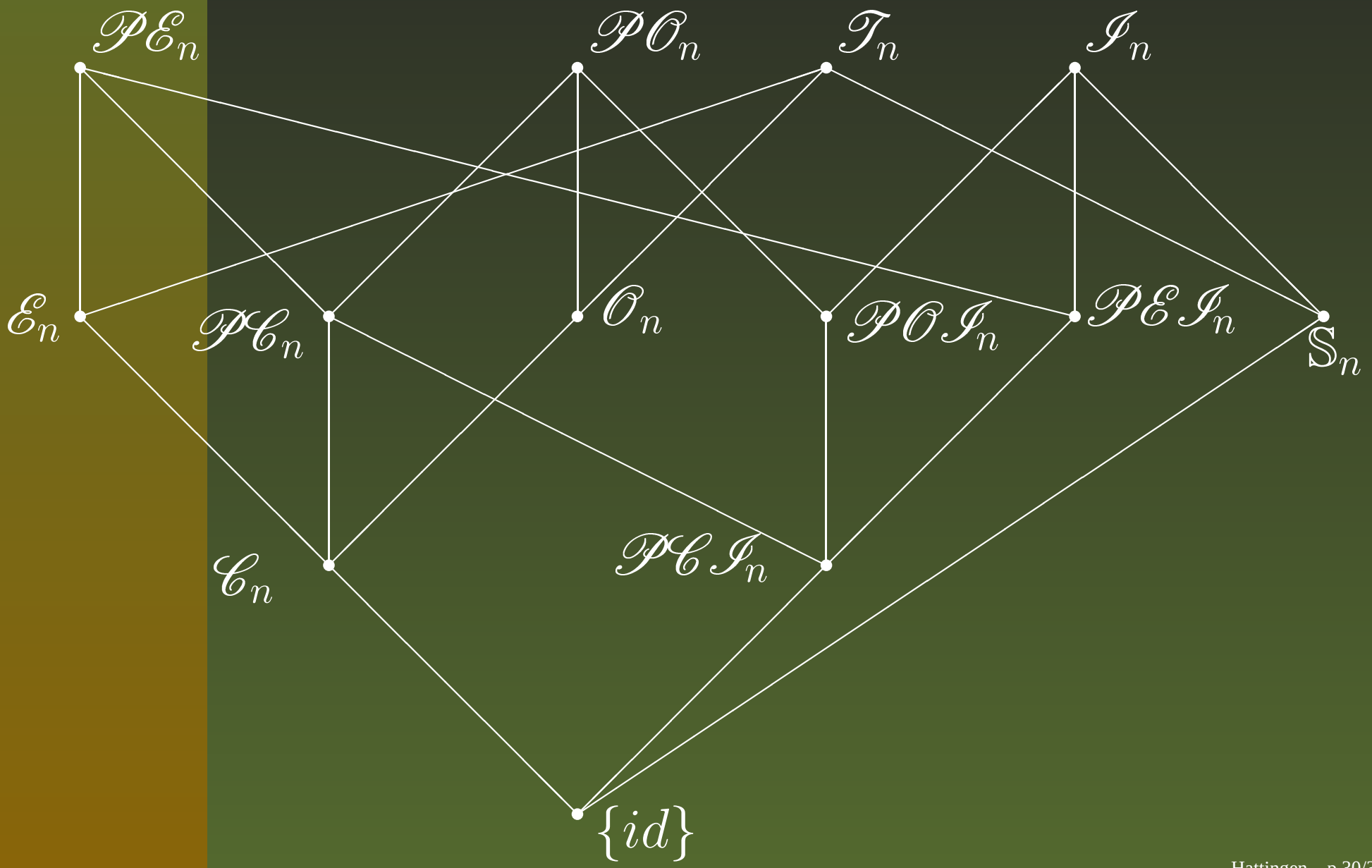
Conclusion

- $\mathscr{PE}_n$
- $\mathscr{PO}_n$
- $\mathscr{T}_n$
- $\mathscr{I}_n$

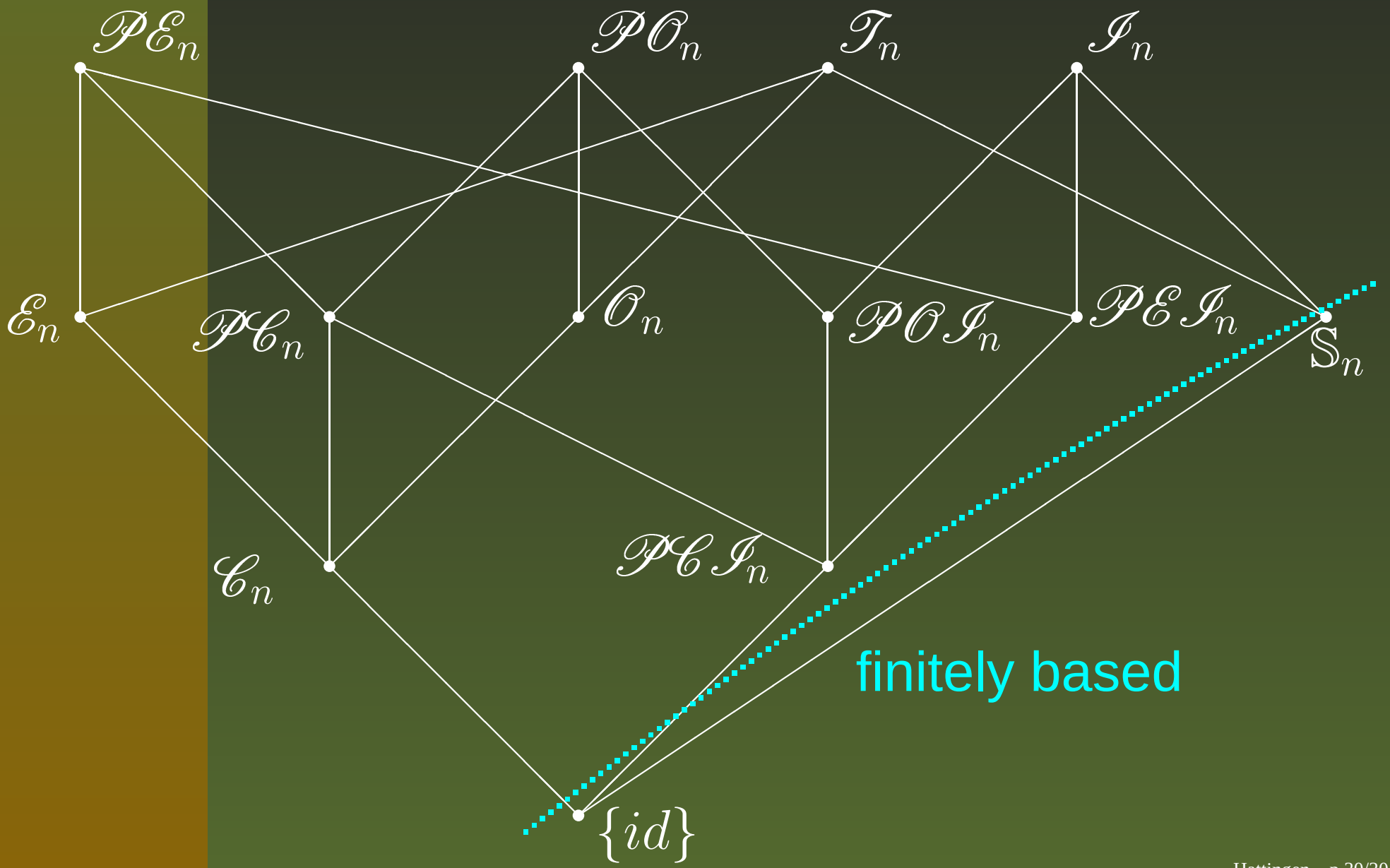
Conclusion


$\mathscr{PE}_n$
$\mathscr{PO}_n$
$\mathscr{T}_n$
$\mathscr{I}_n$
$\mathscr{E}_n$
$\mathscr{PC}_n$
$\mathscr{O}_n$
$\mathscr{POI}_n$
$\mathscr{PEI}_n$
$\mathbb{S}_n$

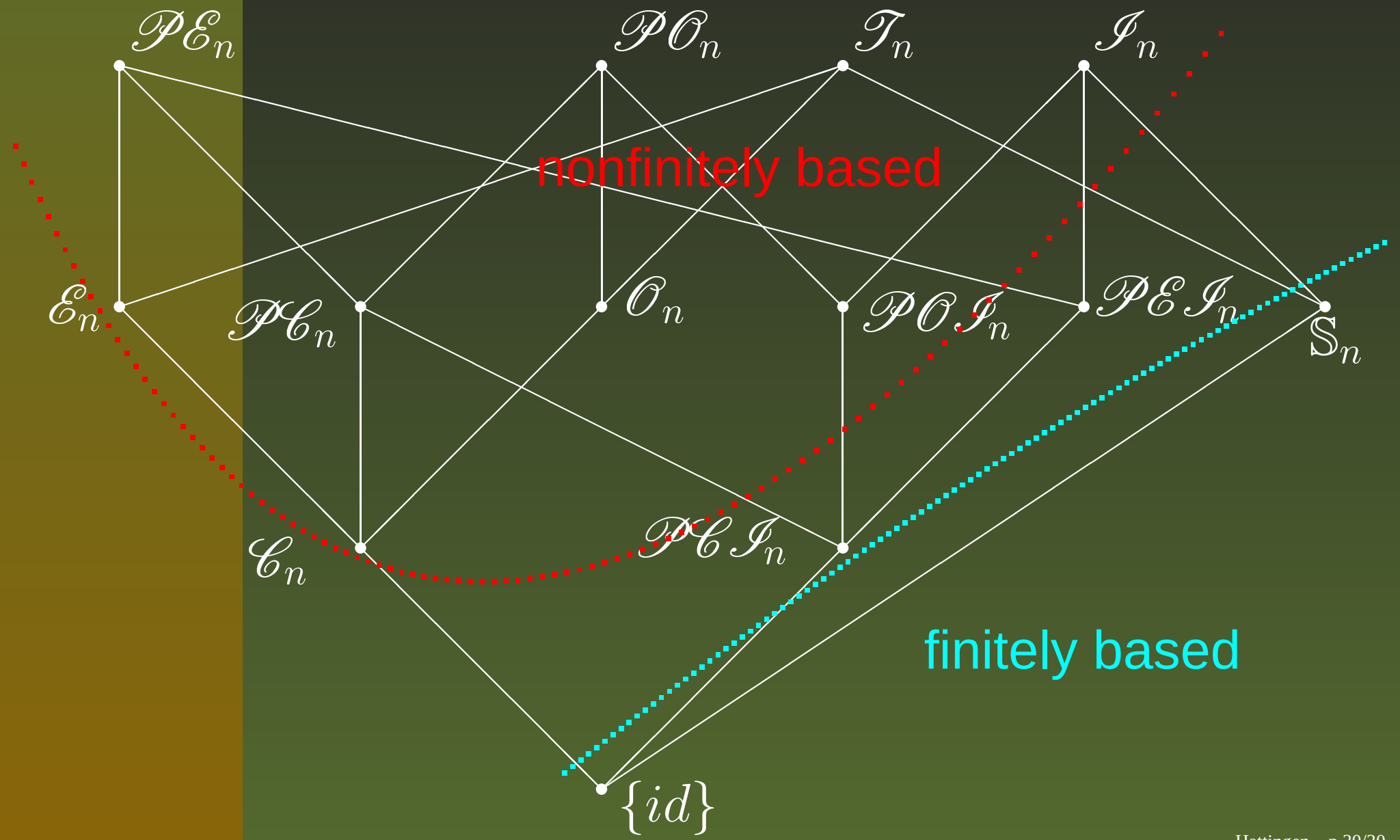
Conclusion


$\mathscr{PE}_n$
$\mathscr{PO}_n$
$\mathscr{T}_n$
$\mathscr{I}_n$
$\mathscr{E}_n$
$\mathscr{PC}_n$
$\mathscr{O}_n$
$\mathscr{POI}_n$
$\mathscr{PEI}_n$
$\mathbb{S}_n$
$\mathscr{C}_n$
$\mathscr{PCI}_n$
$\{id\}$

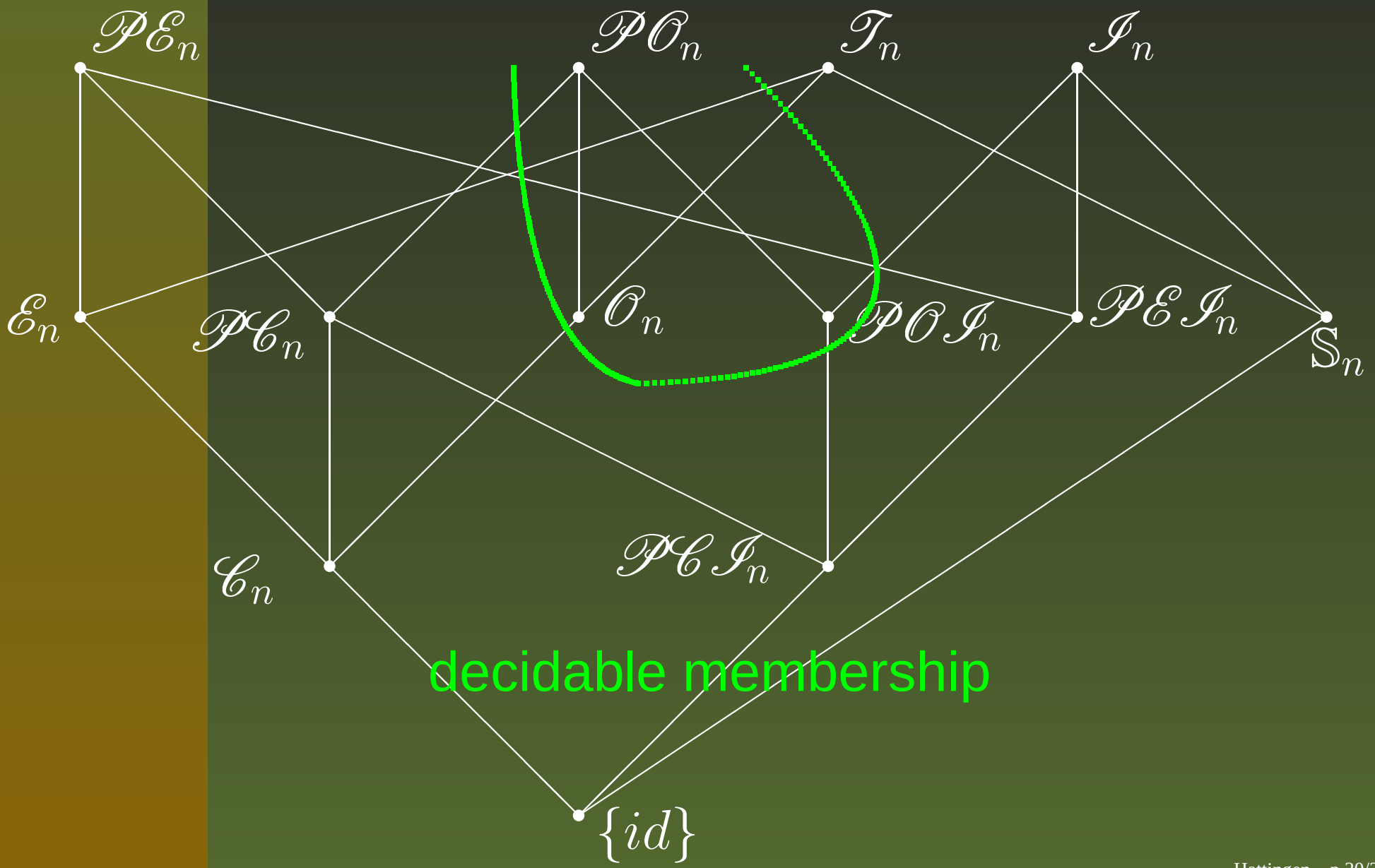
Conclusion


$\mathscr{PE}_n$
$\mathscr{PO}_n$
$\mathscr{T}_n$
$\mathscr{I}_n$
$\mathscr{E}_n$
$\mathscr{PC}_n$
$\mathscr{O}_n$
$\mathscr{POI}_n$
$\mathscr{PEI}_n$
$\mathbb{S}_n$
$\mathscr{C}_n$
$\mathscr{PCI}_n$
$\{id\}$
finitely based

Conclusion


$\mathscr{PE}_n$
$\mathscr{PO}_n$
$\mathscr{T}_n$
$\mathscr{I}_n$
nonfinitely based
$\mathscr{E}_n$
$\mathscr{PC}_n$
$\mathscr{O}_n$
$\mathscr{POI}_n$
$\mathscr{PEI}_n$
$\mathbb{S}_n$
$\mathscr{C}_n$
$\mathscr{PCI}_n$
finitely based
$\{id\}$

Conclusion


$\mathscr{PE}_n$
$\mathscr{PO}_n$
$\mathscr{T}_n$
$\mathscr{I}_n$
$\mathscr{E}_n$
$\mathscr{PC}_n$
$\mathscr{O}_n$
$\mathscr{POI}_n$
$\mathscr{PEI}_n$
$\mathbb{S}_n$
$\mathscr{C}_n$
$\mathscr{PCI}_n$
decidable membership
$\{id\}$