WORDS'03 (Turku, September 11, 2003)

Reflexive relations, extensive transformations and piecewise testable languages of a given height

Dedicated to Imre Simon on the occasion of his 60th birthday

Mikhail Volkov

Ural State University, Ekaterinburg, Russia

Simon’s Theorem

Instead of trying to overview all Imre’s achievement, we look in some detail at one of his first results:

Simon’s Theorem

Instead of trying to overview all Imre’s achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

Simon's Theorem

Instead of trying to overview all Imre's achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

- is easy to explain;

Simon's Theorem

Instead of trying to overview all Imre's achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

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

Simon's Theorem

Instead of trying to overview all Imre's achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

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

Simon's Theorem

Instead of trying to overview all Imre's achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

- is easy to explain;
- is hard to prove;
- has many surprising connections, including those with combinatorics of words theory;

Simon's Theorem

Instead of trying to overview all Imre's achievement, we look in some detail at one of his first results:

his famous characterization of *piecewise testable languages*

- is easy to explain;
- is hard to prove;
- has many surprising connections, including those with combinatorics of words theory;
- has opened an area which still remains very vivid.

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

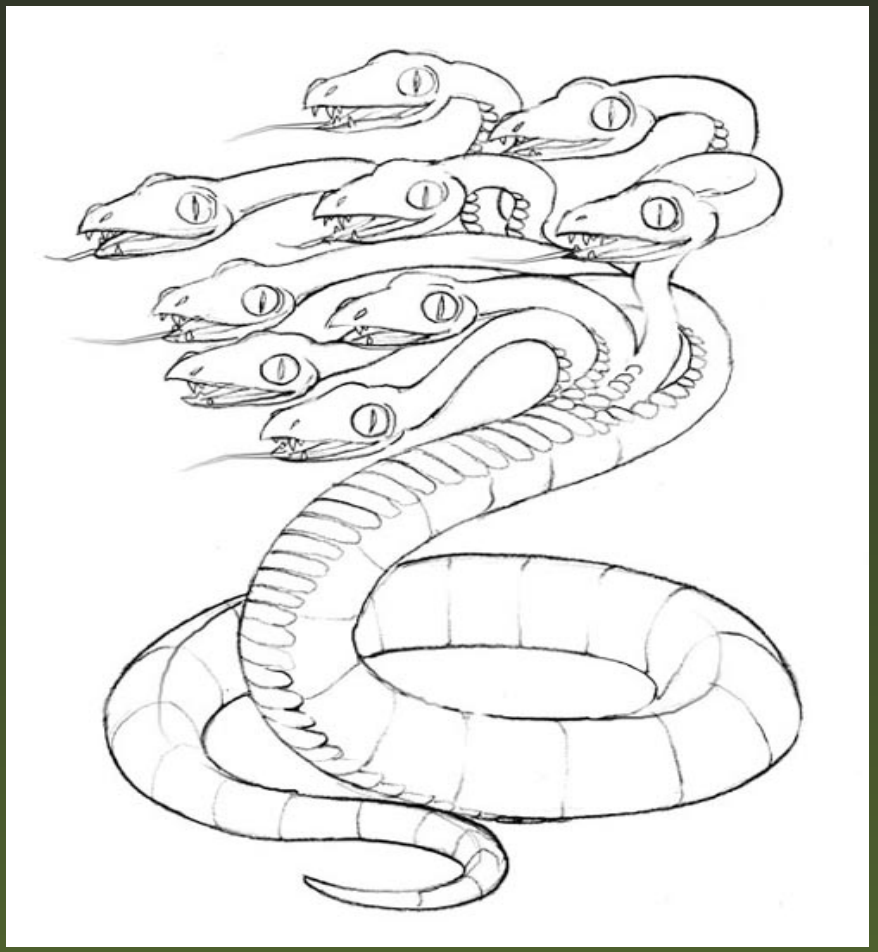


Figure 1: A 9-head hydra

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

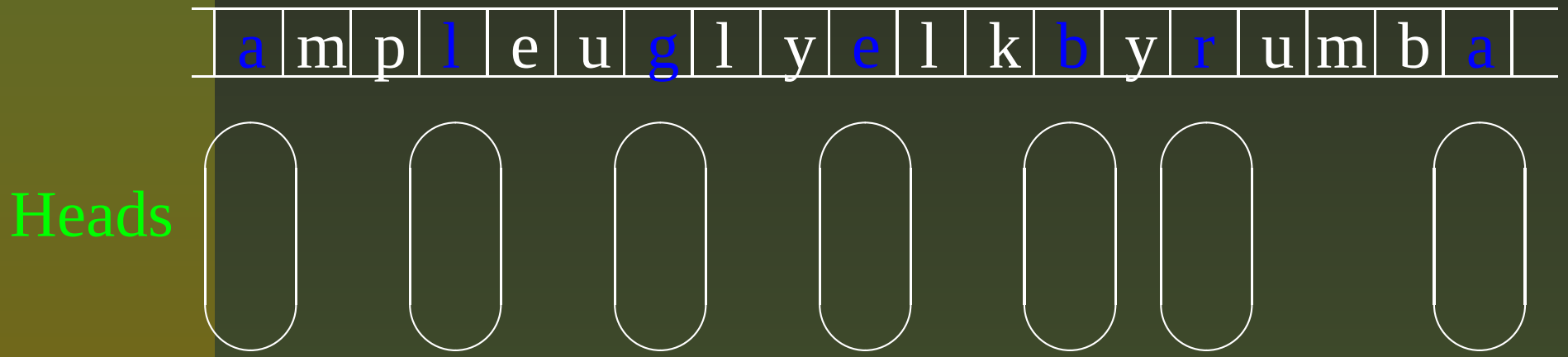


Figure 2: A 7-head hydra automaton

Hydra Automata

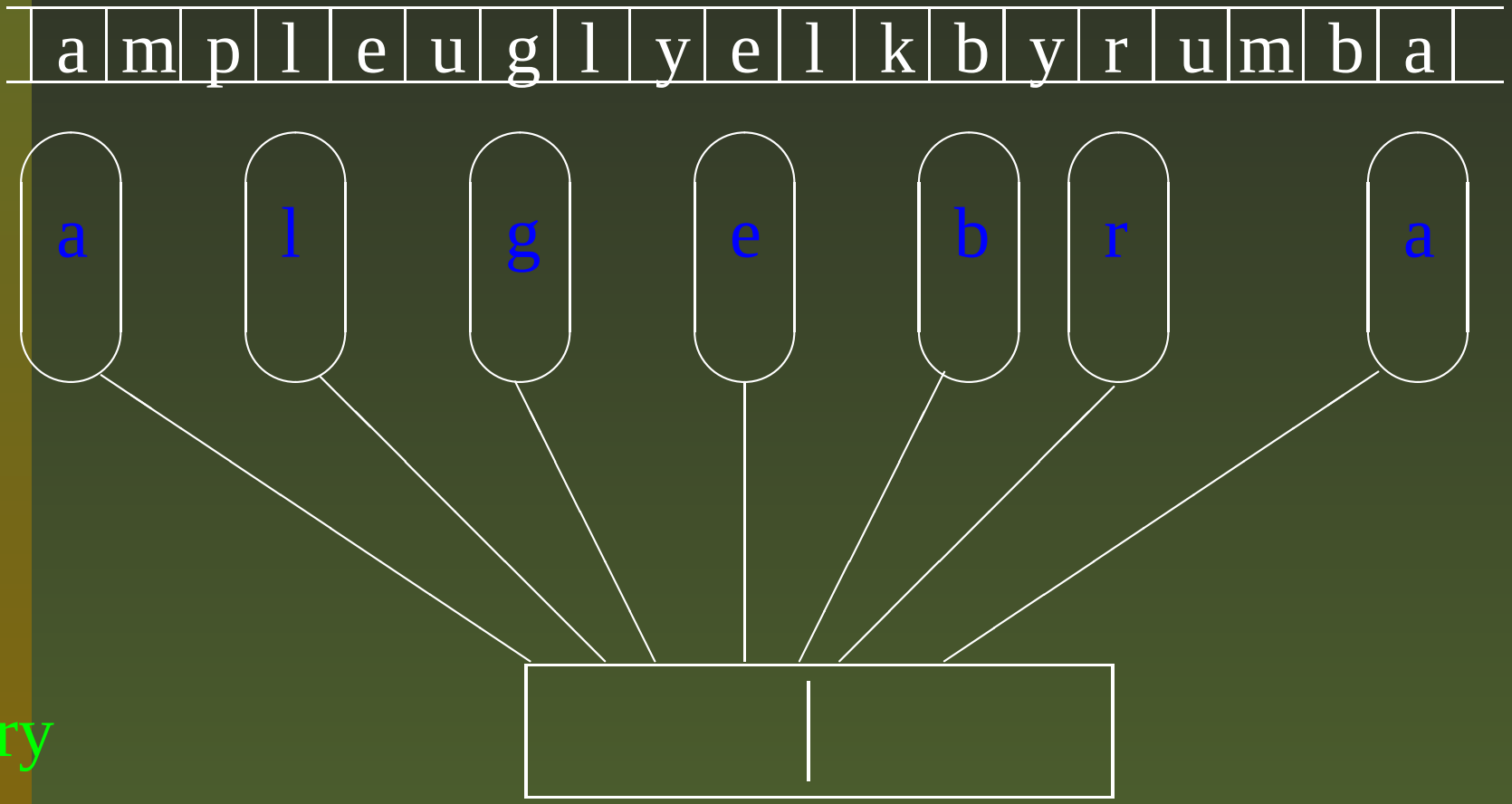


Figure 2: A 7-head hydra automaton

Hydra Automata

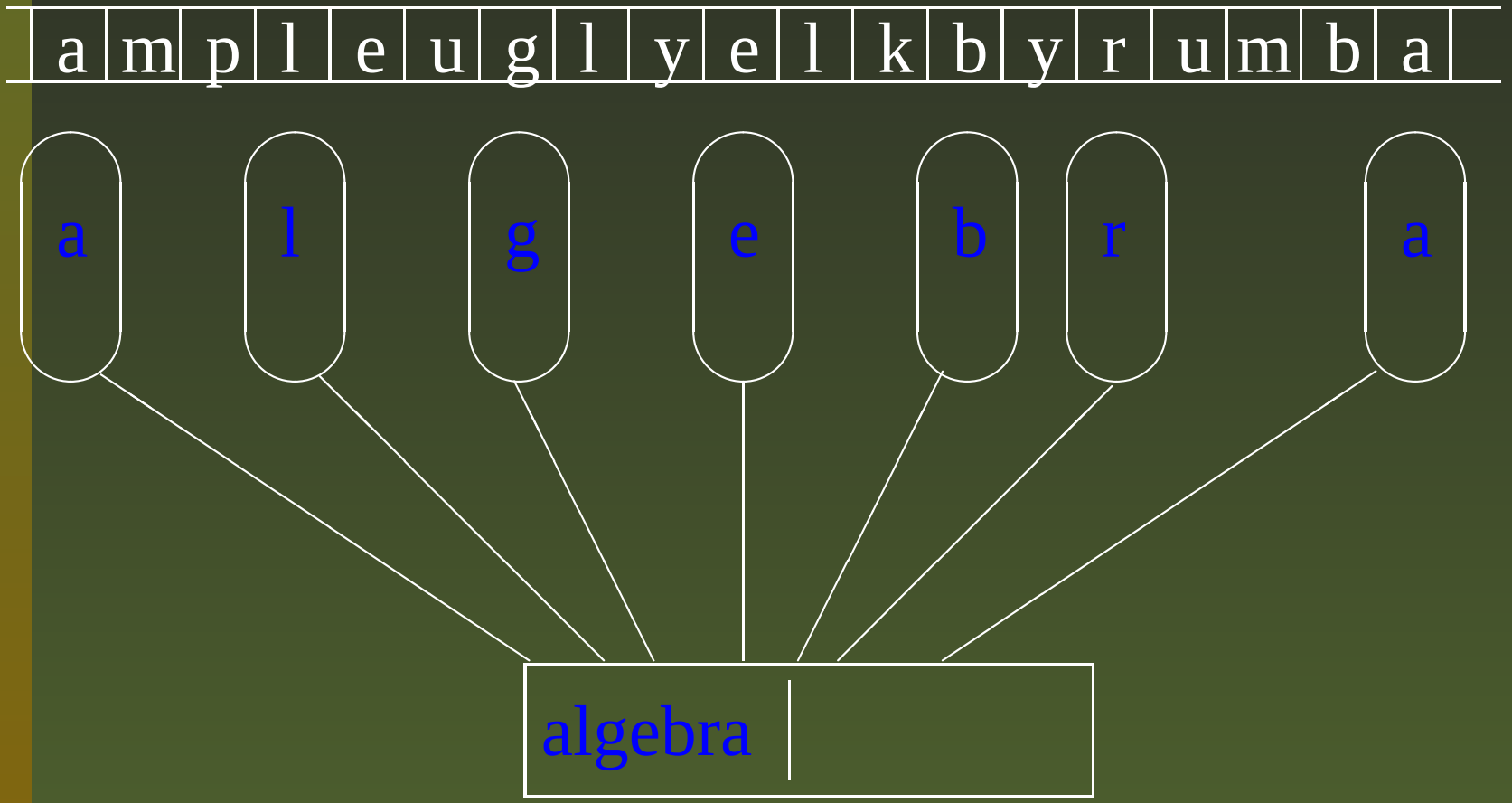


Figure 2: A 7-head hydra automaton

Hydra Automata

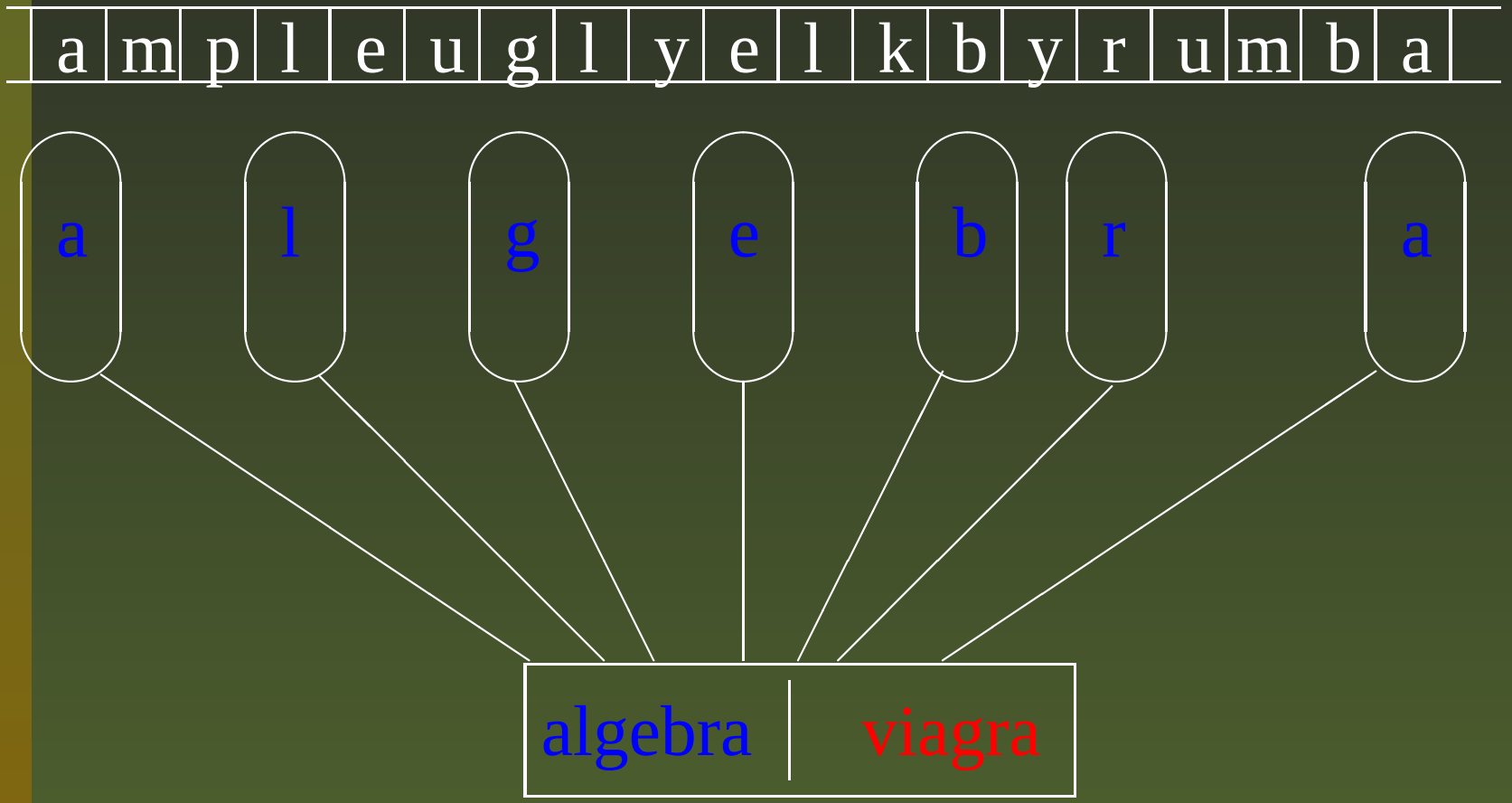


Figure 2: A 7-head hydra automaton

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

$u \sim_L v$ 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 proofs, 1972, 1975;

Simon's Theorem

- Deep: a crossing where many ideas meet.

Proofs come from:

- Combinatorics on words — Simon's original proofs, 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 proofs, 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 proofs, 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 proofs, 1972, 1975;
- Model theory — Stern, 1985;
- Ordered monoids — Straubing and Thérien, 1988;
- Profinite topology — Almeida, 1990;
- Endomorphisms of linear orders — Higgins, 1997.

Simon vs. Eilenberg

Simon's theorem is an *instance* of the Eilenberg correspondence between varieties of recognizable languages and pseudovarieties of finite monoids. (A *pseudovariety* is a class of finite monoids closed under submonoids, morphic images and finite direct products.)

Simon vs. Eilenberg

Simon's theorem is an *instance* of the Eilenberg correspondence between varieties of recognizable languages and pseudovarieties of finite monoids. (A *pseudovariety* is a class of finite monoids closed under submonoids, morphic images and finite direct products.) This shouldn't be understood as claiming Simon's theorem be a *consequence* of Eilenberg's theorem!

Simon vs. Eilenberg

Simon's theorem is an *instance* of the Eilenberg correspondence between varieties of recognizable languages and pseudovarieties of finite monoids. (A *pseudovariety* is a class of finite monoids closed under submonoids, morphic images and finite direct products.) This shouldn't be understood as claiming Simon's theorem be a *consequence* of Eilenberg's theorem!

$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$ (Euler) vs. $\sum_{n=1}^{\infty} \frac{1}{n^z} = \zeta(z)$ (Riemann)

Euler's result can be now written as $\zeta(2) = \frac{\pi^2}{6}$ but this is not a *consequence* of Riemann's considerations.

Recognizing Height

In terms of the Eilenberg correspondence Simon's theorem means that the pseudovariety **J** of all finite $\mathscr{J}$-trivial monoids and the variety of all piecewise testable languages correspond to each other.

Recognizing Height

In terms of the Eilenberg correspondence Simon's theorem means that the pseudovariety $\mathbf{J}$ of all finite $\mathscr{J}$-trivial monoids and the variety of all piecewise testable languages correspond to each other.

Let $\mathbf{J}_h$ denote the pseudovariety of finite monoids that corresponds to the class 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.

Recognizing Height

Recall that by the definition $\mathbf{J}_h$ is the pseudovariety generated by the syntactic monoids of languages from $PLT_h(\Sigma)$ for all finite alphabets Σ. Thus, the algebraic counterpart of Question 2 is the following:

Recognizing Height

Recall that by the definition $\mathbf{J}_h$ is the pseudovariety generated by the syntactic monoids of languages from $PLT_h(\Sigma)$ for all finite alphabets Σ. Thus, the algebraic counterpart of Question 2 is the following:

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

Recall that by the definition $\mathbf{J}_h$ is the pseudovariety generated by the syntactic monoids of languages from $PLT_h(\Sigma)$ for all finite alphabets Σ. Thus, the algebraic counterpart of Question 2 is the following:

Question 3. *Given a finite monoid M and a positive integer 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 2. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*

Straubing's Theorem

Theorem 2. (Howard Straubing, 1980) *For a finite monoid M the following are equivalent:*
(i) *M is $\mathscr{J}$-trivial;*

Straubing's Theorem

Theorem 2. (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 2. (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 2. (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 2. (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 3. ($\sim$, 2003) *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 3. ($\sim$, 2003) *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 3. ($\sim$, 2003) *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.*

Theorem 3: Transformations

By now we have seen how identities come into the play.
But where do relations and transformations come from?

Theorem 3: Transformations

By now we have seen how identities come into the play. But where do relations and transformations come from?

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

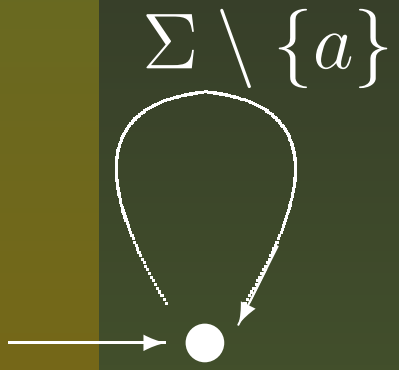
Theorem 3: Transformations

By now we have seen how identities come into the play. But where do relations and transformations come from?

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

Theorem 3: Transformations

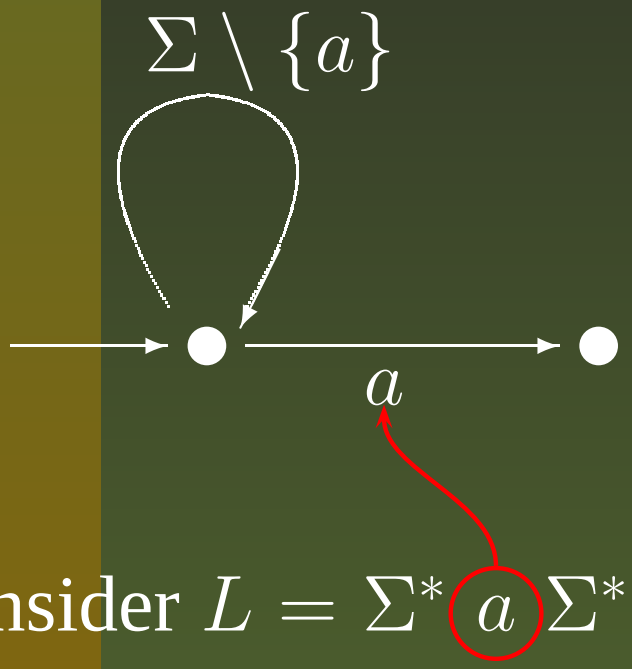
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

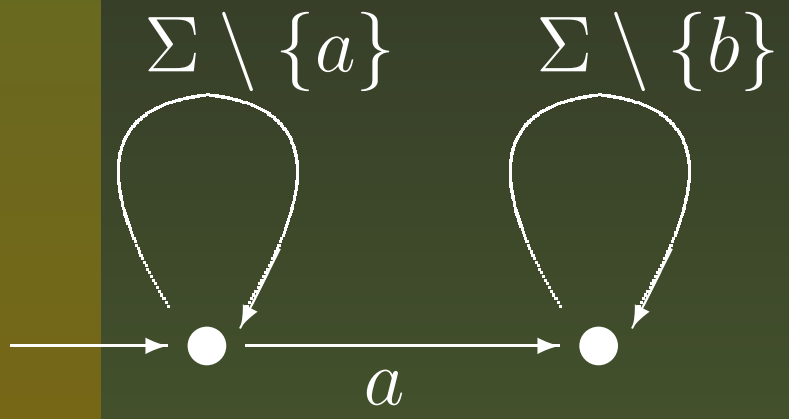
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

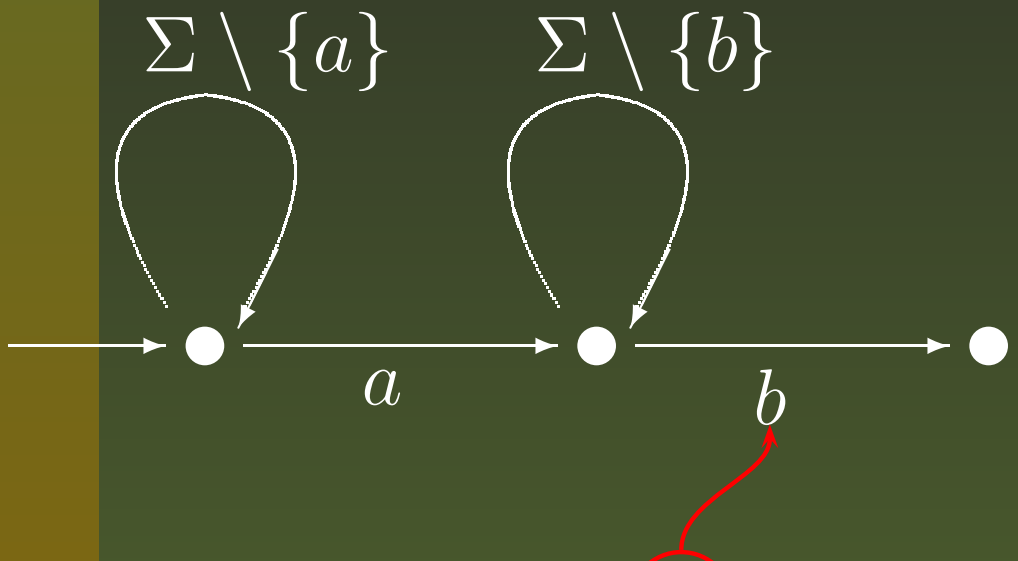
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

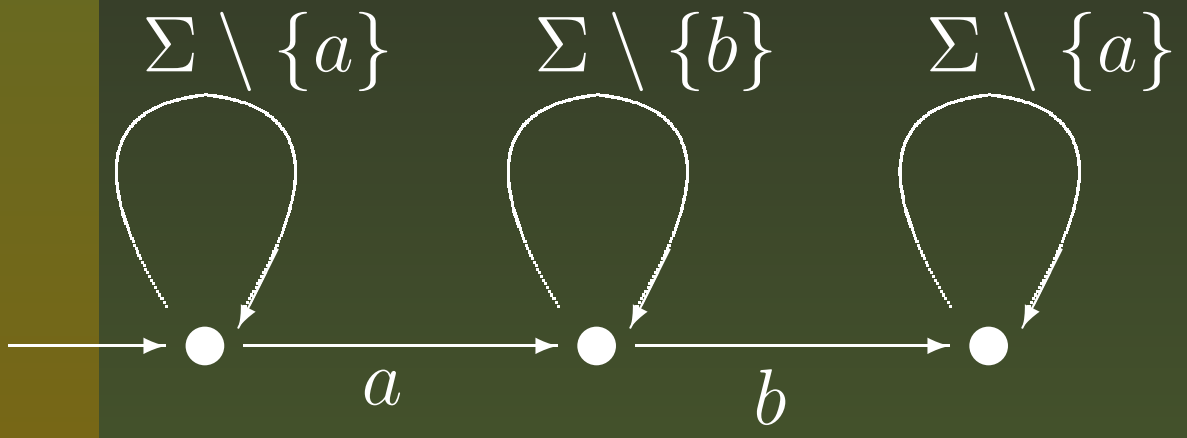
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

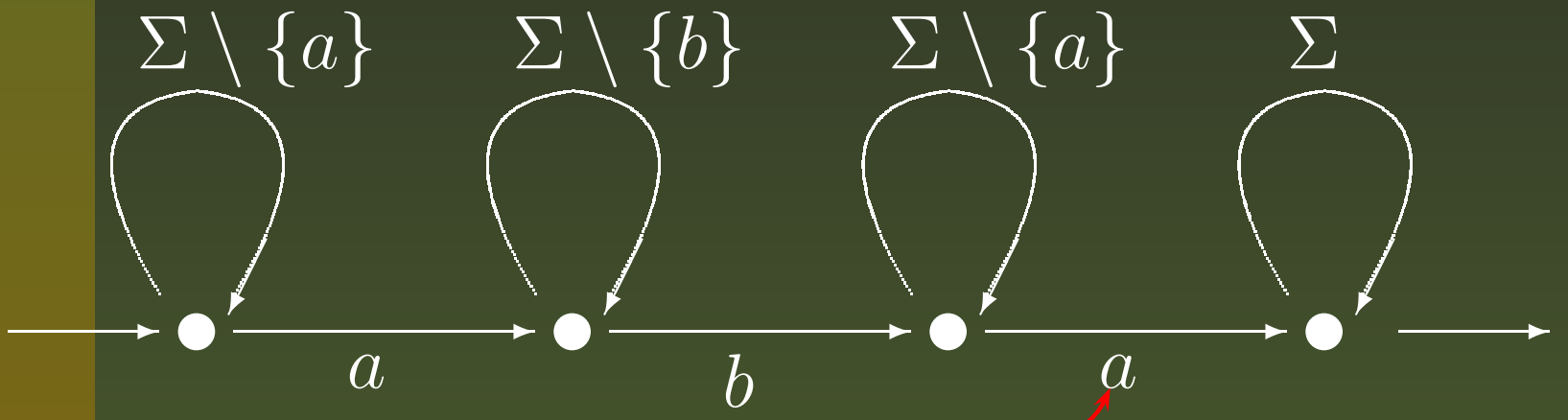
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

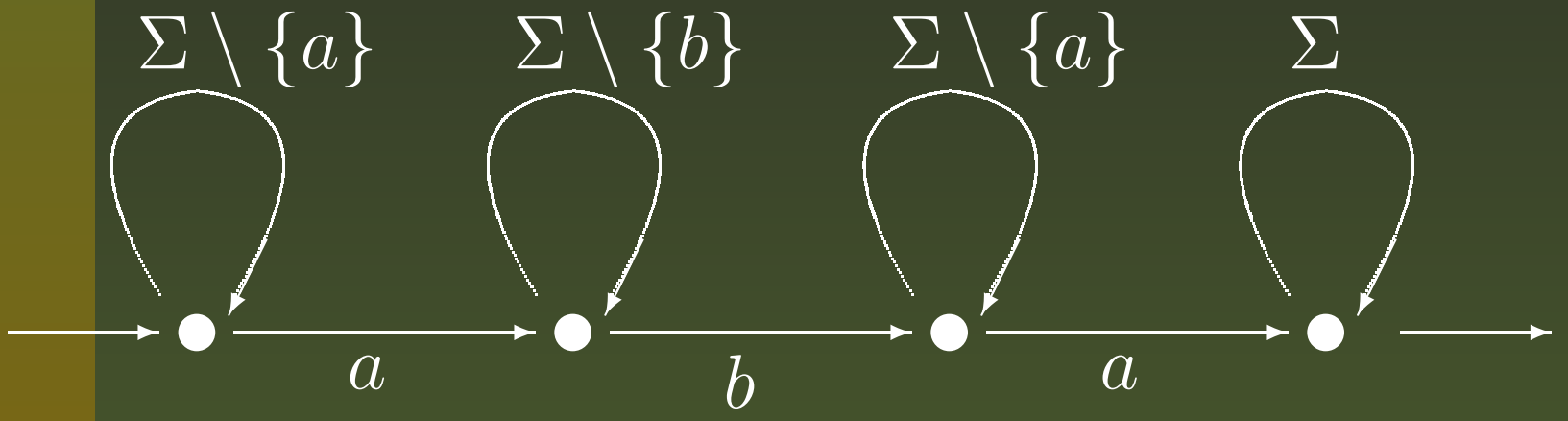
By now we have seen how identities come into the play. But where do relations and transformations come from?



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

Theorem 3: Transformations

By now we have seen how identities come into the play. But where do relations and transformations come from?



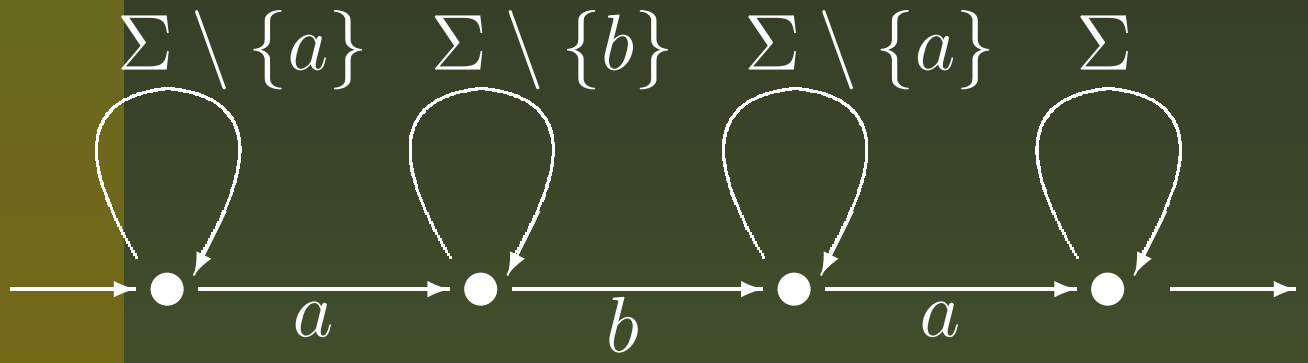
Consider $L = \Sigma^* \; a \; \Sigma^* \; b \; \Sigma^* \; a \; \Sigma^*$, quite a typical piecewise testable language, and build a deterministic finite automaton that recognizes L.

Theorem 3: transformations

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

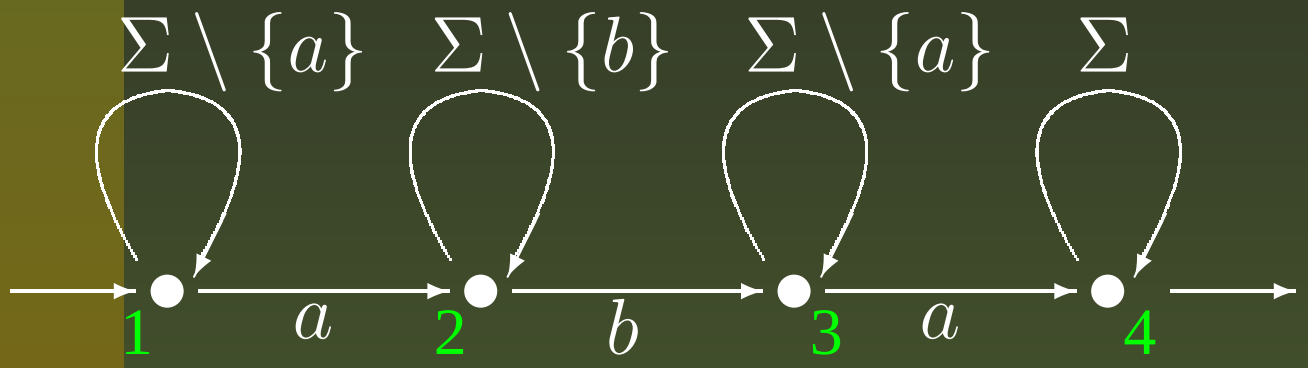
Theorem 3: transformations

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



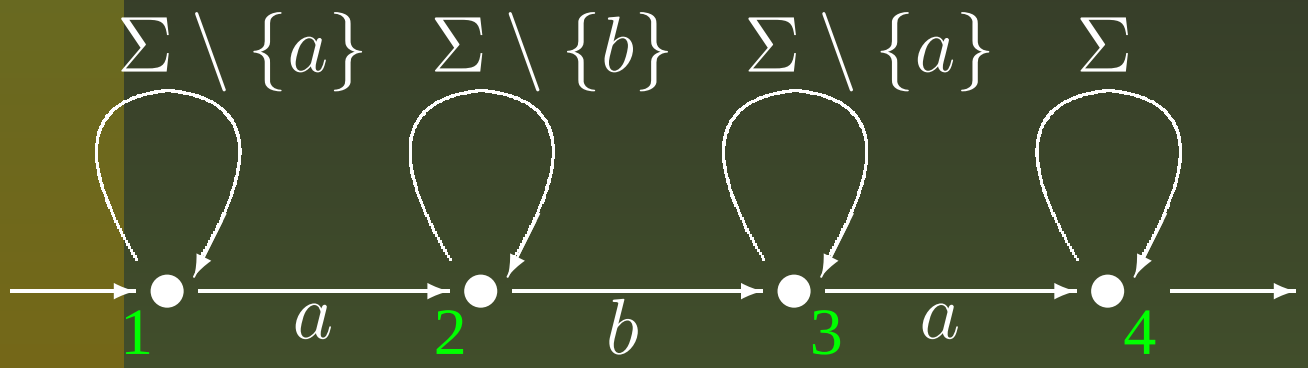
Theorem 3: transformations

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



Theorem 3: transformations

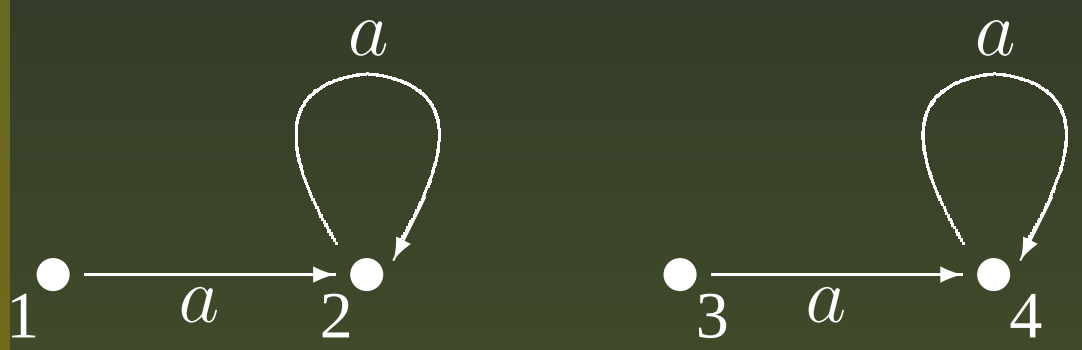
Now impose a linear order on the state set of the automaton we built:



It is easy to see that with respect to this order the transformation induced by the letters are order preserving and extensive.

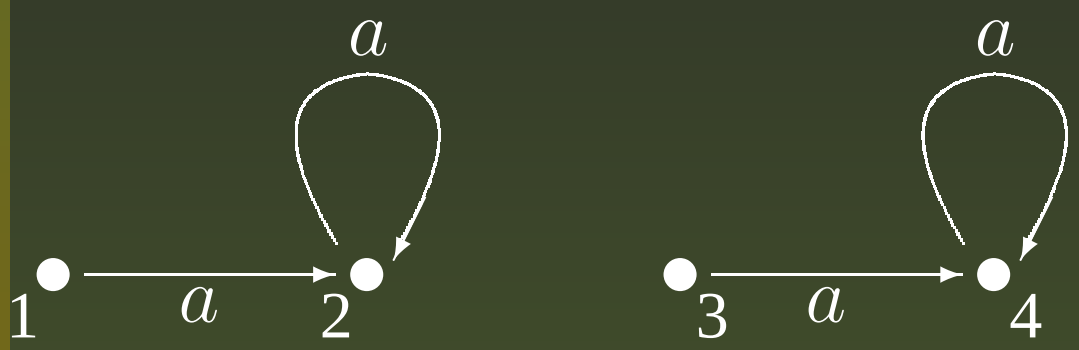
Theorem 3: Transformations

For instance, this is the one induced by a:

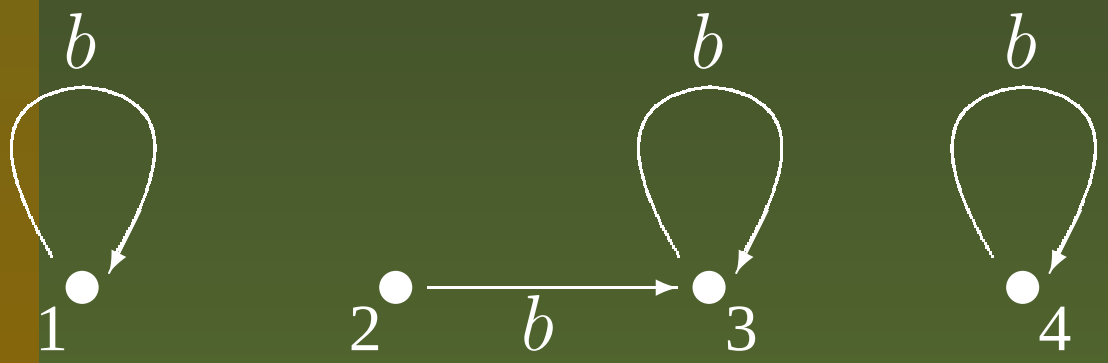


Theorem 3: Transformations

For instance, this is the one induced by a:



And this is the action of b:



Theorem 3: transformations

We see that the transition monoid of the deterministic automaton recognizing our language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of order preserving and extensive transformations of the chain $1 < 2 < 3 < 4$, i.e. it is a submonoid in $\mathscr{C}_4$.

Theorem 3: transformations

We see that the transition monoid of the deterministic automaton recognizing our language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of order preserving and extensive transformations of the chain $1 < 2 < 3 < 4$, i.e. it is a submonoid in $\mathscr{C}_4$.

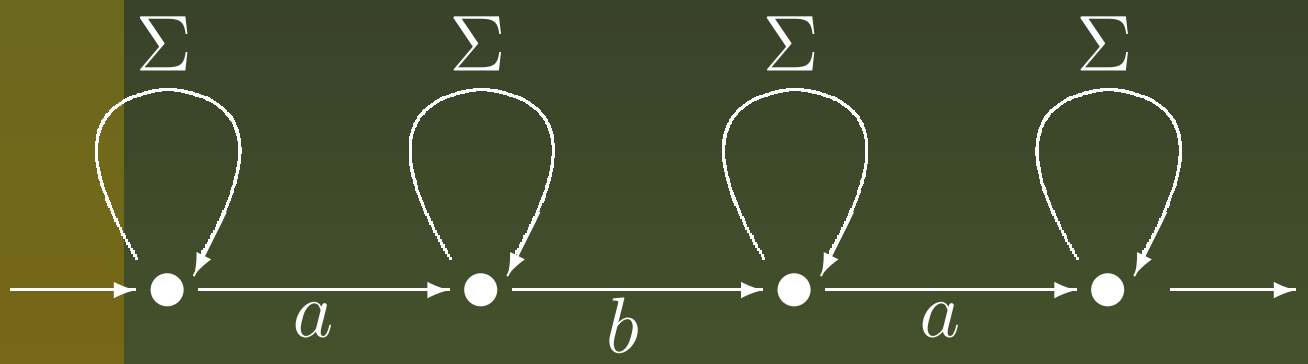
It should be clear that in general, when starting with the language $L = \Sigma^* a_1 \Sigma^* a_2 \cdots \Sigma^* a_h \Sigma^*$, we end up in the monoid $\mathscr{C}_{h+1}$. Therefore the pseudovariety $\mathbf{J}_h$ is contained in the pseudovariety $\mathrm{pvar}\, \mathscr{C}_{h+1}$.

Theorem 3: Relations

Now we want to recognize the same language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ by a non-deterministic finite automaton.

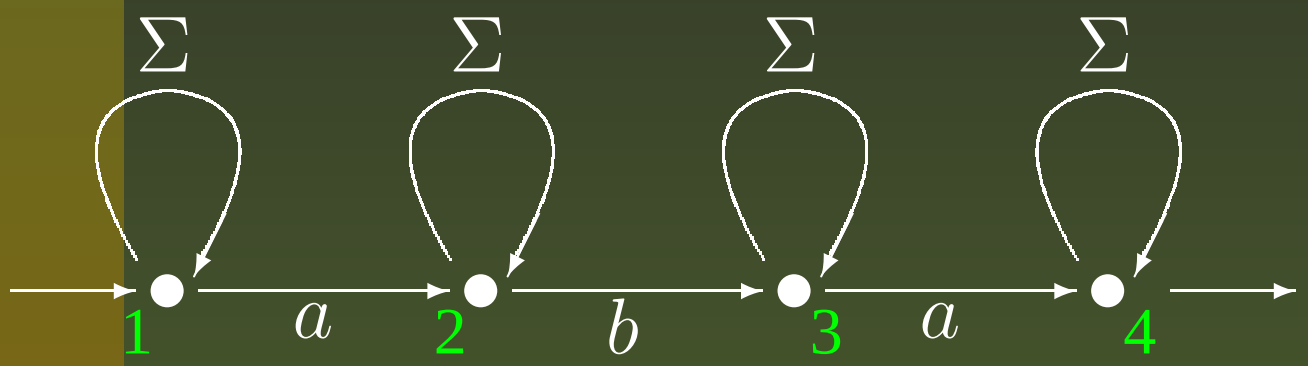
Theorem 3: Relations

Now we want to recognize the same language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ by a non-deterministic finite automaton.



Theorem 3: Relations

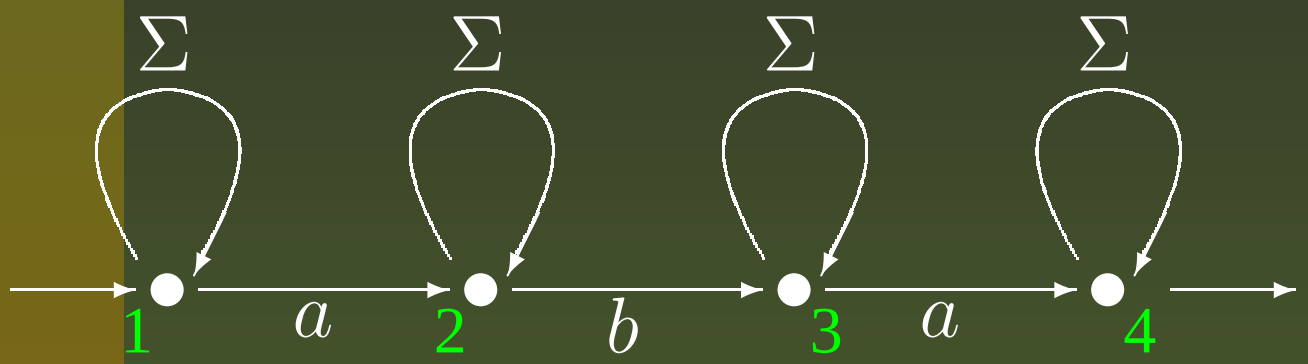
Now we want to recognize the same language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ by a non-deterministic finite automaton.



We index the states

Theorem 3: Relations

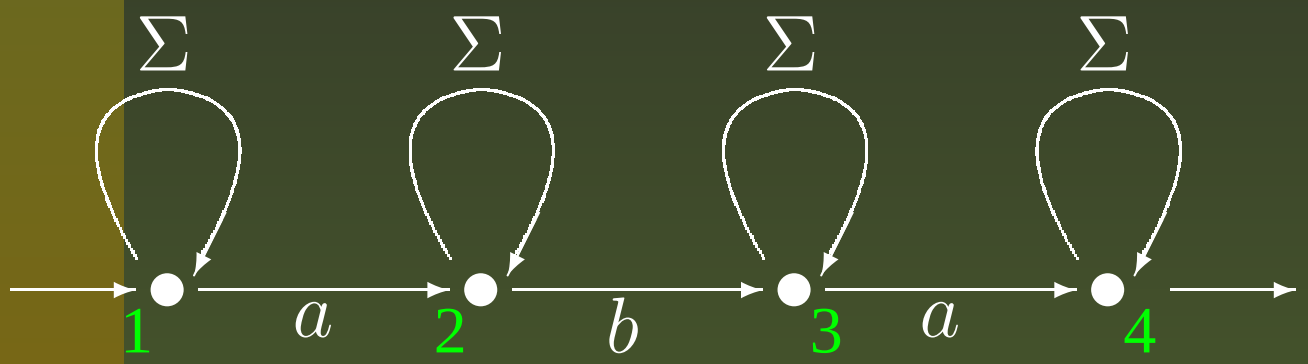
Now we want to recognize the same language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ by a non-deterministic finite automaton.



We index the states and consider the corresponding relations on $\{1, 2, 3, 4\}$.

Theorem 3: Relations

Now we want to recognize the same language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ by a non-deterministic finite automaton.



We index the states and consider the corresponding relations on $\{1, 2, 3, 4\}$. One readily sees that these relations will be reflexive and upper triangular.

Theorem 3: Relations

For instance, this matrix represents the relation induced by a:

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

Theorem 3: Relations

For instance, this matrix represents the relation induced by a:

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

And this is the relation induced by b:

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

Theorem 3: Relations

We see that the transition monoid of the non-deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of reflexive and upper triangular relations on the set $\{1, 2, 3, 4\}$, i.e. it is a submonoid in $\mathscr{U}_4$.

Theorem 3: Relations

We see that the transition monoid of the non-deterministic automaton recognizing the language $L = \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^*$ consists of reflexive and upper triangular relations on the set $\{1, 2, 3, 4\}$, i.e. it is a submonoid in $\mathscr{U}_4$.

It should be clear that in general, when departing from the language $L = \Sigma^* a_1 \Sigma^* a_2 \cdots \Sigma^* a_h \Sigma^*$, we end up in the monoid $\mathscr{U}_{h+1}$. Therefore the pseudovariety $\mathbf{J}_h$ is contained in the pseudovariety $\operatorname{pvar} \mathscr{U}_{h+1}$ and hence also in the pseudovariety $\operatorname{pvar} \mathscr{R}_{h+1}$.

Recognizing Height

Is this solution to the problem of recognizing height efficient?

Recognizing Height

Is this solution to the problem of recognizing height efficient? This doesn't follow from any general result — 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 solution to the problem of recognizing height efficient? This doesn't follow from any general result — 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 solution to the problem of recognizing height efficient? This doesn't follow from any general result — 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 $|B|$).

Moreover, Ralph McKenzie constructed a monoid M (of size 8009) such that the problem of whether or not a given monoid N belongs to $\mathrm{pvar}\, M$ is NP-hard.

Recognizing Height

Is this solution to the problem of recognizing height efficient? This doesn't follow from any general result — 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 $|B|$).

Moreover, Ralph McKenzie constructed a monoid M (of size 8009) such that the problem of whether or not a given monoid N belongs to $\mathrm{pvar}\, M$ is NP-hard. Marcel Jackson has reduced the size of the example to 55.

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 $\mathscr{U}_2$ of all Boolean upper unitriangular 2×2-matrices.

Recognizing Height via Identities

Example: The pseudovariety $\mathbf{J}_1$ of all $\mathscr{J}$-trivial monoids of height 1 is generated by the monoid $\mathscr{U}_2$ of all Boolean upper unitriangular 2×2-matrices. Such a matrix has only one "free" entry: $\begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix}$. Therefore $\mathscr{U}_2$ 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 $\mathscr{U}_2$ of all Boolean upper unitriangular 2×2-matrices. Such a matrix has only one "free" entry: $\begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix}$. Therefore $\mathscr{U}_2$ 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 $\mathscr{U}_2$ of all Boolean upper unitriangular 2×2-matrices. Such a matrix has only one "free" entry: $\begin{pmatrix} 1 & * \\ 0 & 1 \end{pmatrix}$. Therefore $\mathscr{U}_2$ 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 4. ($\sim$, 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 4. ($\sim$, 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 4. ($\sim$, 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 4. ($\sim$, 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 words lead to certain classes of transformation monoids of finite 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 words lead to certain classes of transformation monoids of finite 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 words lead to certain classes of transformation monoids of finite 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 $\{1, 2, \ldots, n\}$:

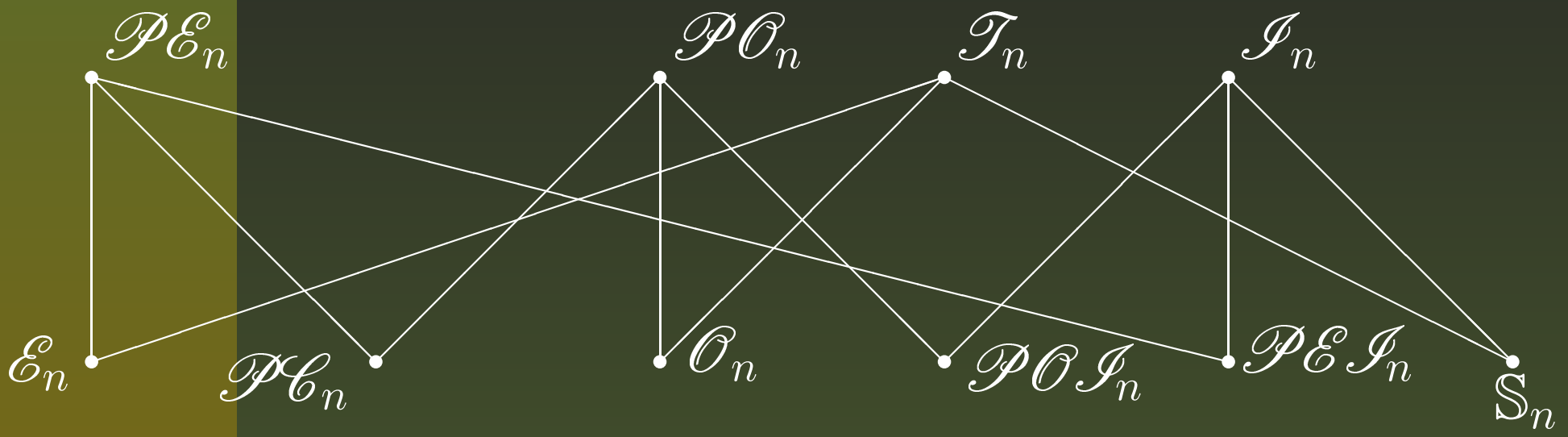
- being *total* (everywhere defined);
- being *injective*;
- being *order preserving*, that is, endomorphic;
- being *extensive*.

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

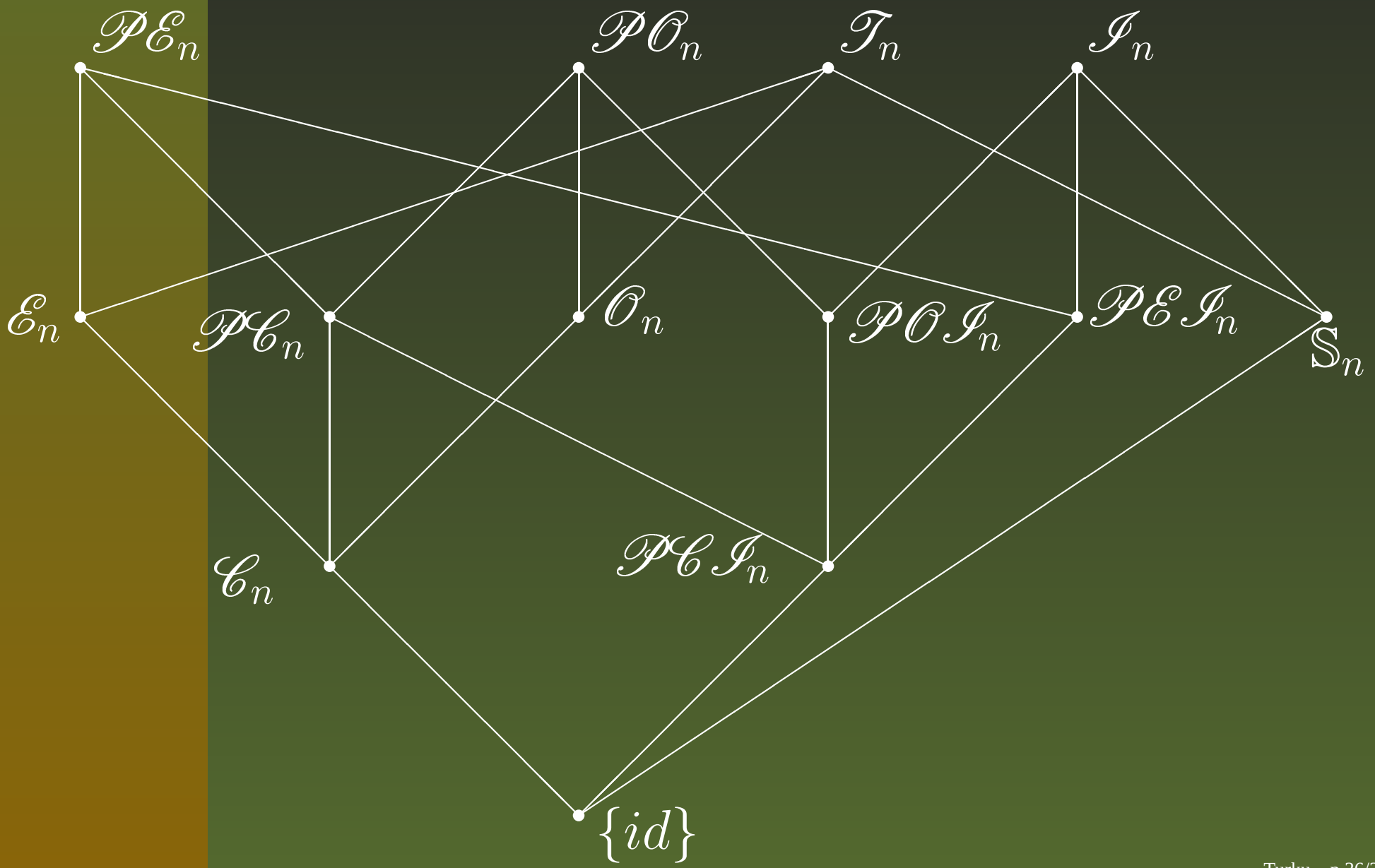
Conclusion

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

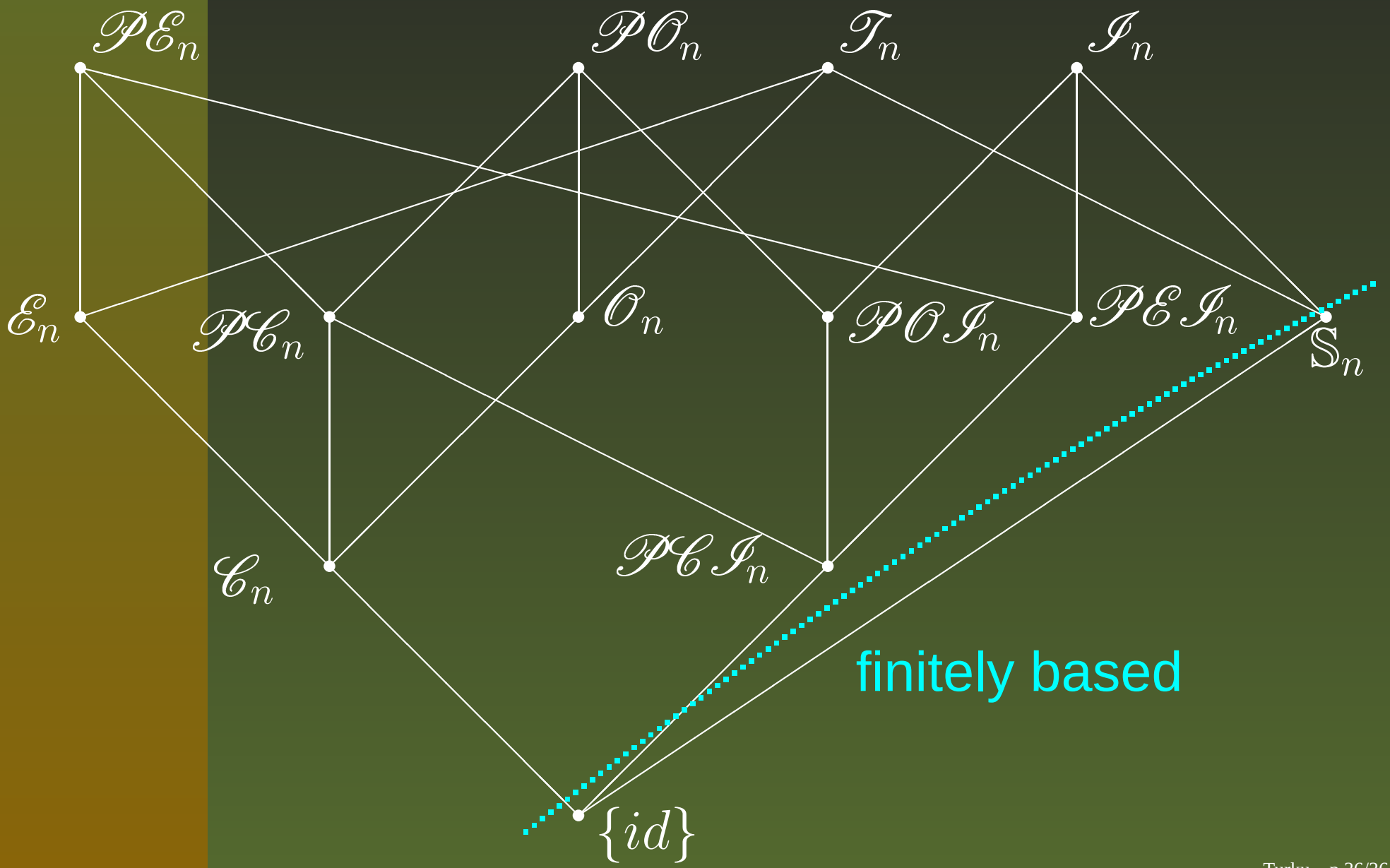
Conclusion


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

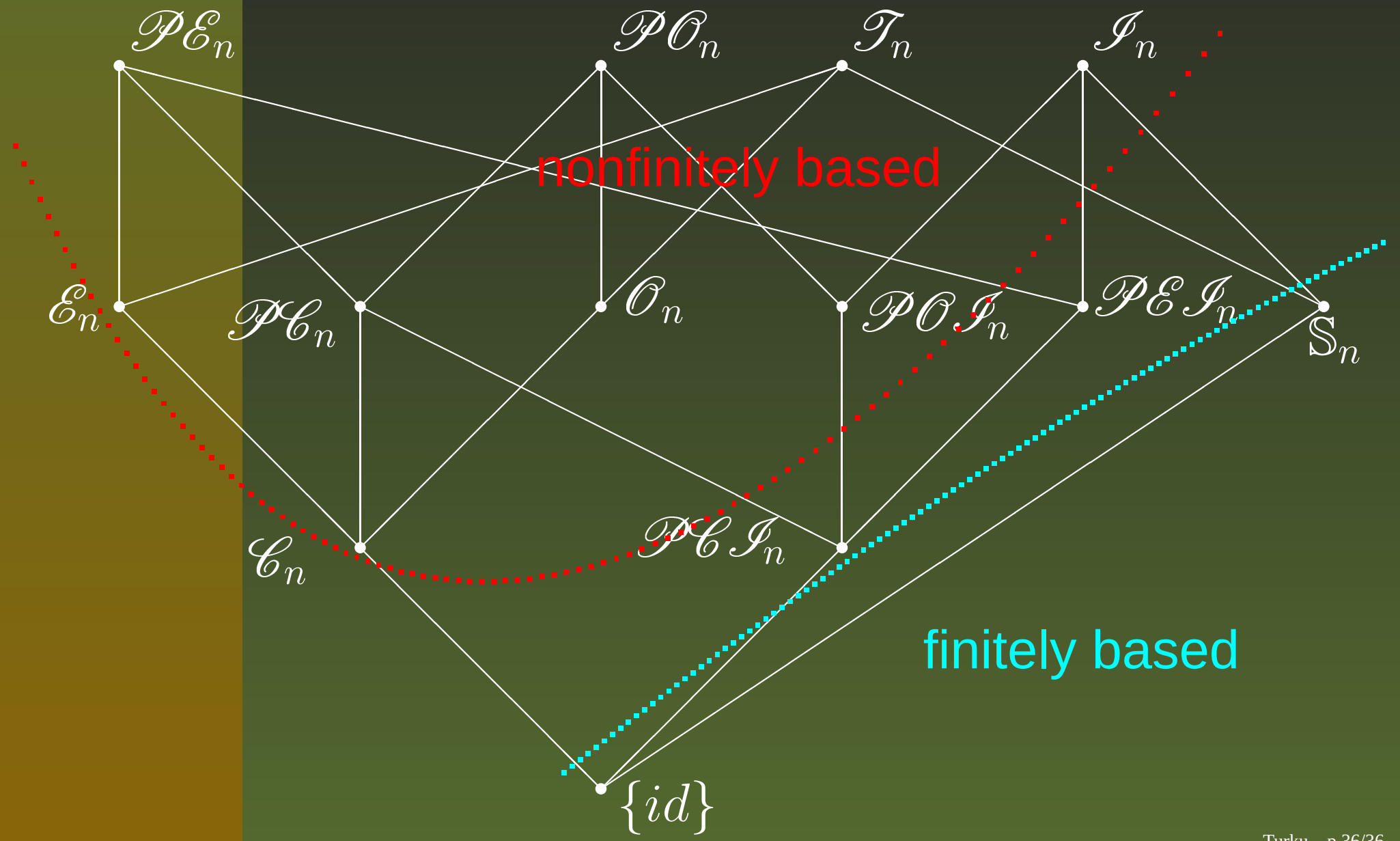
Conclusion


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

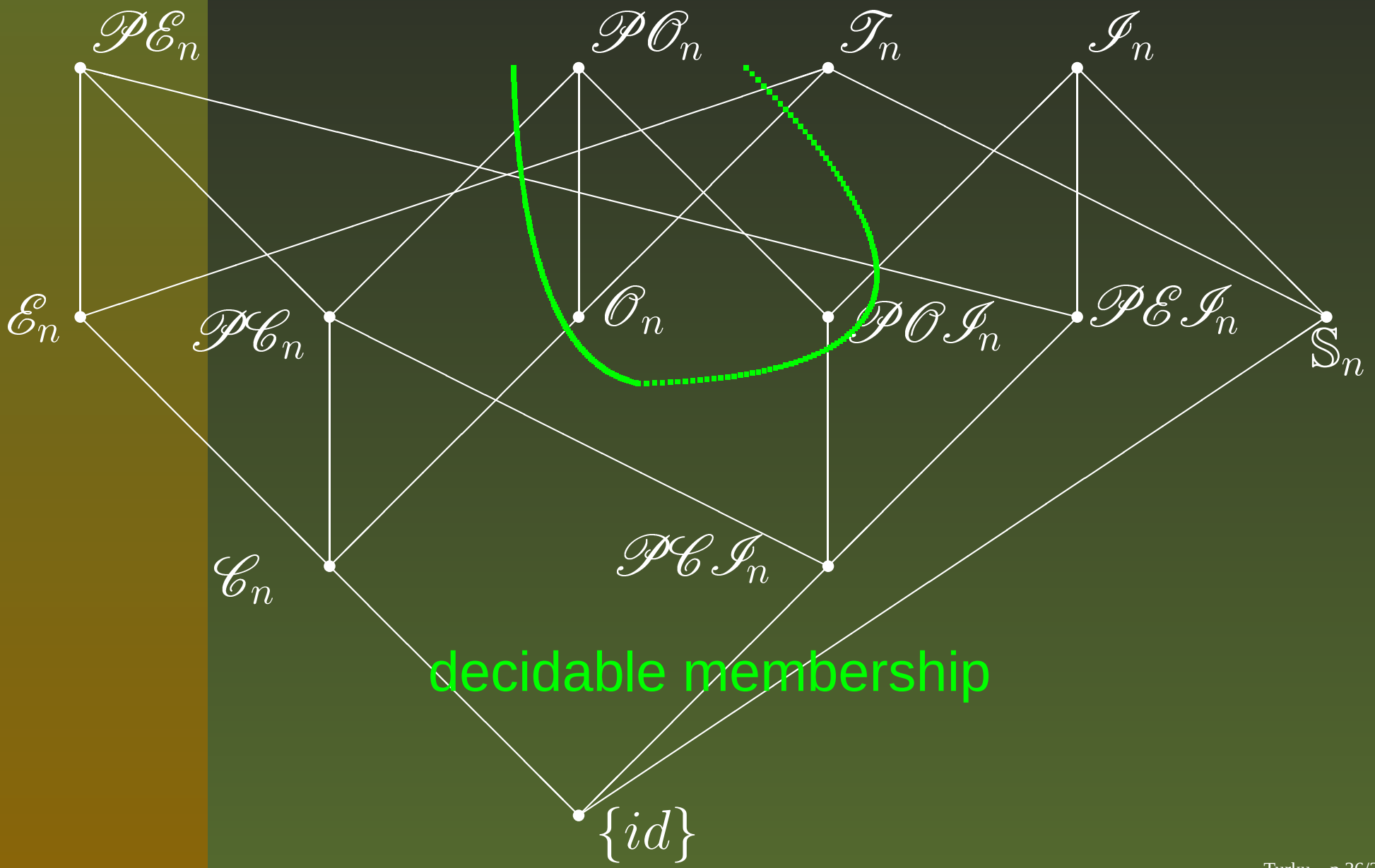
Conclusion


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