Algebraic Recognizability of Languages

Pascal Weil

LaBRI, Université Bordeaux-1 and CNRS

MFCS Conference, Prague, August 2004

The general problem

- Problem: to specify and analyse infinite sets by finite means

The general problem

- Problem: to specify and analyse infinite sets by finite means
- These are sets of discrete structures, arising from the modeling of computations, data structures,...

The general problem

- Problem: to specify and analyse infinite sets by finite means
- These are sets of discrete structures, arising from the modeling of computations, data structures,...
- Since the origins of TCS, words = finite sequences of letters

The general problem

- Problem: to specify and analyse infinite sets by finite means
- These are sets of discrete structures, arising from the modeling of computations, data structures,...
- Since the origins of TCS, words = finite sequences of letters
- Need to discuss more models: trees, traces, pomsets, graphs

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata
- rational expressions

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

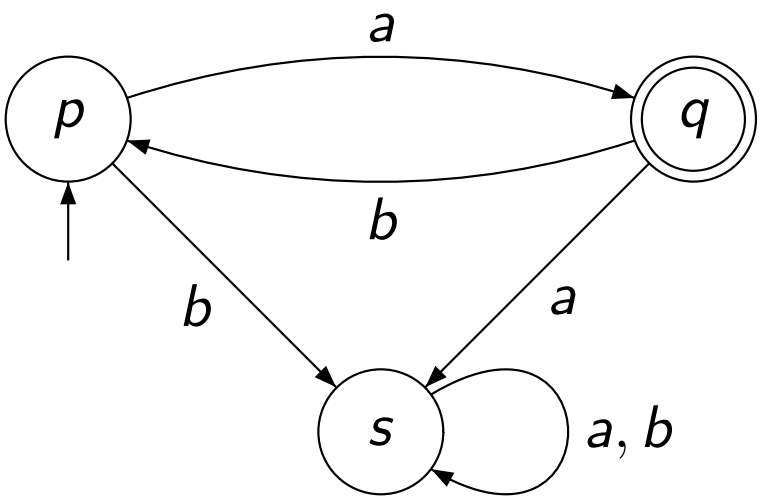
In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata
- rational expressions
- monadic second order (MSO) definability

The theory was first developped for words. The class of recognizable (regular) languages has the best algorithmic properties, and is expressive enough.

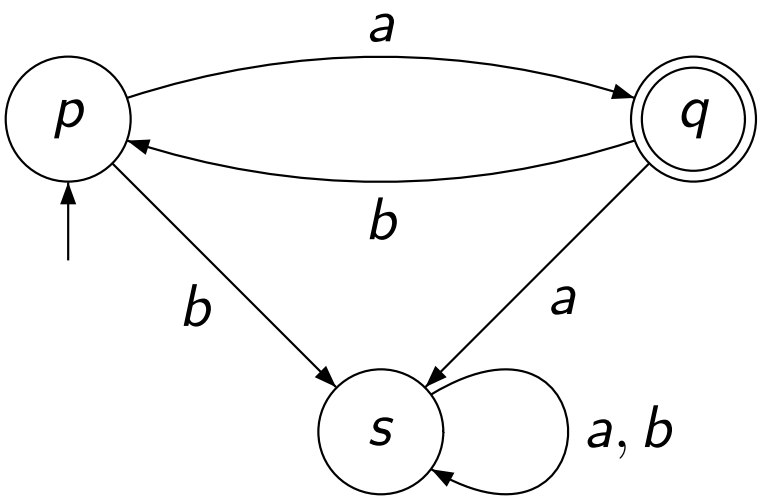
In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata
- rational expressions
- monadic second order (MSO) definability
- algebraic recognizability, by finite monoids



In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata
- rational expressions
- monadic second order (MSO) definability
- algebraic recognizability, by finite monoids



$(ab)^*a$

$\forall x\ \forall y$
$\neg(\exists z\ x < z < y) \Rightarrow (R_a x \Leftrightarrow R_b y)$
$\wedge\ R_a\mathrm{min}\ \wedge\ R_a\mathrm{max}$

In the case of finite words, the situation is *ideal*. 4 formalisms are effectively, and sometimes efficiently equivalent (Kleene, Myhill, Nerode, Büchi, Elgot, Mezei, Schützenberger,. . .)

- finite state automata
- rational expressions
- monadic second order (MSO) definability
- algebraic recognizability, by finite monoids

Our claim today: the algebraic point of view is especially relevant and useful, in the case of words and also in other settings

Our claim today: the algebraic point of view is especially relevant and useful, in the case of words and also in other settings

The rest of this talk is an attempt to substantiate this claim by

Our claim today: the algebraic point of view is especially relevant and useful, in the case of words and also in other settings

The rest of this talk is an attempt to substantiate this claim by

- analyzing the word case

Our claim today: the algebraic point of view is especially relevant and useful, in the case of words and also in other settings

The rest of this talk is an attempt to substantiate this claim by

- analyzing the word case
- discussing what we can expect in other settings

Our claim today: the algebraic point of view is especially relevant and useful, in the case of words and also in other settings

The rest of this talk is an attempt to substantiate this claim by

- analyzing the word case
- discussing what we can expect in other settings
- and reviewing a number of situations where interesting results have emerged

Syntactic monoid vs. minimal automaton

L, recognizable language: 2 *computable* invariants

Syntactic monoid vs. minimal automaton

L, recognizable language: 2 *computable* invariants

- $\mathcal{A}(L)$, the minimal automaton of L

Syntactic monoid vs. minimal automaton

L, recognizable language: 2 *computable* invariants

- $\mathcal{A}(L)$, the minimal automaton of L
- $S(L)$, the syntactic monoid of L

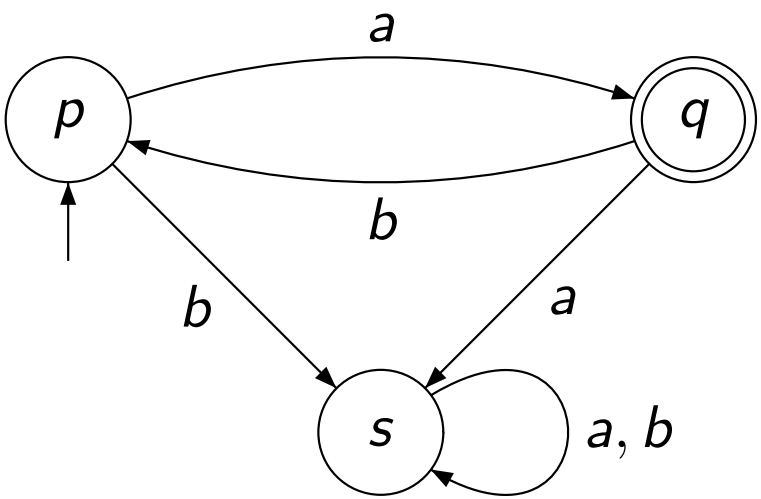
Syntactic monoid vs. minimal automaton

L, recognizable language: 2 *computable* invariants

- $\mathcal{A}(L)$, the minimal automaton of L
- $S(L)$, the syntactic monoid of L

$S(L)$ is the transition monoid of $\mathcal{A}(L)$

Syntactic monoid vs. minimal automaton



	p	q	s	
a	q	s	s	
b	s	p	s	
a^2	s	s	s	
ab	p	s	s	
ba	s	q	s	
b^2	s	s	s	$b^2 = a^2 = 0$
aba	q	s	s	$aba = a,\ bab = b$

$$S((ab)^*a) = \{a, b, ab, ba, 0\}$$

$S(L)$ is the transition monoid of $\mathcal{A}(L)$

Syntactic monoid vs. minimal automaton

L, recognizable language: 2 *computable* invariants

- $\mathcal{A}(L)$, the minimal automaton of L
- $S(L)$, the syntactic monoid of L

Classification of recognizable languages: characterize and decide significant classes

Star-free languages

L is *star-free* if it is obtained from the letters using Boolean operations and concatenations

Star-free languages

L is *star-free* if it is obtained from the letters using Boolean operations and concatenations

For instance, $(ab)^*a$ is star-free

Star-free languages

L is *star-free* if it is obtained from the letters using Boolean operations and concatenations

For instance, $(ab)^*a$ is star-free

$$(ab)^*a = aA^* \cap A^*a \cap \left(A^*aaA^* \cup A^*bbA^*\right)^c$$

Star-free languages

L is *star-free* if it is obtained from the letters using Boolean operations and concatenations

For instance, $(ab)^*a$ is star-free

$$(ab)^*a = aA^* \cap A^*a \cap \left(A^*aaA^* \cup A^*bbA^*\right)^c$$

$$A^* = a \cup a^c$$

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- *L is star-free*

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- *L is star-free*
- *L is FO-definable*

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- *L is star-free*
- *L is FO-definable*
- *L is PTL-definable*

PTL, propositional temporal logic, modal logic using modalities next, eventually, until

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- *L is star-free*
- *L is FO-definable*
- *L is PTL-definable*
- *$\mathcal{A}(L)$ is counter-free*

PTL, propositional temporal logic, modal logic using modalities next, eventually, until

$\mathcal{A}$ counter-free: if $q \cdot u^n = q$, then $q \cdot u = q$

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- L *is star-free*
- L *is FO-definable*
- L *is PTL-definable*
- $\mathcal{A}(L)$ *is counter-free*
- $S(L)$ *is aperiodic*

PTL, propositional temporal logic, modal logic using modalities next, eventually, until

$\mathcal{A}$ counter-free: if $q \cdot u^n = q$, then $q \cdot u = q$

S aperiodic: if $s^n = s$, then $s^2 = s$

Star-free languages vs. FO-definability

Theorem (Schützenberger, McNaughton, Pappert, Kamp)
TFAE

- *L is star-free*
- *L is FO-definable*
- *L is PTL-definable*
- *$\mathcal{A}(L)$ is counter-free*
- *$S(L)$ is aperiodic*

essentially the same property; provides decidability

$\mathcal{A}$ counter-free: if $q \cdot u^n = q$, then $q \cdot u = q$

S aperiodic: if $s^n = s$, then $s^2 = s$

Extending FO with modulo quantifiers

FO+MOD: allow quantification $\exists x\ \varphi(x)$ and $\exists^{\bmod q} x\ \varphi(x)$, *i.e.*, $\mathrm{card}\{x \mid \varphi(x) \text{ holds}\}$ is a multiple of q.

Extending FO with modulo quantifiers

FO+MOD: allow quantification $\exists x\ \varphi(x)$ and $\exists^{\text{mod } q} x\ \varphi(x)$, *i.e.*, $\text{card}\{x \mid \varphi(x) \text{ holds}\}$ is a multiple of q.

$\exists^{\text{mod } 2} x\ R_a x =$ all words with an even number of a (parity)

Extending FO with modulo quantifiers

FO+MOD: allow quantification $\exists x\ \varphi(x)$ and $\exists^{\text{mod } q} x\ \varphi(x)$, *i.e.*, $\text{card}\{x \mid \varphi(x) \text{ holds}\}$ is a multiple of q.

$\exists^{\text{mod } 2} x\ R_a x =$ all words with an even number of a (parity)

Theorem (Straubing, Thérien, Thomas)
L is FO+MOD-definable iff the subgroups of $S(L)$ are solvable

Extending FO with modulo quantifiers

FO+MOD: allow quantification $\exists x\ \varphi(x)$ and $\exists^{\text{mod } q} x\ \varphi(x)$, *i.e.*, $\text{card}\{x \mid \varphi(x) \text{ holds}\}$ is a multiple of q.

$\exists^{\text{mod } 2} x\ R_a x =$ all words with an even number of a (parity)

Theorem (Straubing, Thérien, Thomas)
L is FO+MOD-definable iff the subgroups of $S(L)$ are solvable

a decidable property

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*
- *L is in $\Sigma_2 \cap \Pi_2$*

Σ_2: $\exists x_1 \cdots \exists x_n \, \forall y_1 \cdots \forall y_m \, \varphi$ where φ is quantifier-free
Π_2: $\forall x_1 \cdots \forall x_n \, \exists y_1 \cdots \exists y_m \, \varphi$ where φ is quantifier-free

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*
- *L is in $\Sigma_2 \cap \Pi_2$*
- *L is PTL-definable without until*

Σ_2: $\exists x_1 \cdots \exists x_n \, \forall y_1 \cdots \forall y_m \, \varphi$ where φ is quantifier-free
Π_2: $\forall x_1 \cdots \forall x_n \, \exists y_1 \cdots \exists y_m \, \varphi$ where φ is quantifier-free

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*
- *L is in $\Sigma_2 \cap \Pi_2$*
- *L is PTL-definable without until*
- *L can be obtained from the letters using only disjoint unions and unambiguous concatenations*

Σ_2: $\exists x_1 \cdots \exists x_n \, \forall y_1 \cdots \forall y_m \, \varphi$ where φ is quantifier-free
Π_2: $\forall x_1 \cdots \forall x_n \, \exists y_1 \cdots \exists y_m \, \varphi$ where φ is quantifier-free

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*
- *L is in $\Sigma_2 \cap \Pi_2$*
- *L is PTL-definable without until*
- *L can be obtained from the letters using only disjoint unions and unambiguous concatenations*
- *$S(L)$ is in DA*

S is in DA if, whenever $xyx = x$ for some y, then $x^2 = x$

Restricting FO

FO is equivalent to FO_3 (use only 3 variables) but not to FO_2

Theorem (Etessami, Pin, Schütz, Thérien, Weil, Wilke)
TFAE

- *L is in FO_2*
- *L is in $\Sigma_2 \cap \Pi_2$*
- *L is PTL-definable without until*
- *L can be obtained from the letters using only disjoint unions and unambiguous concatenations*
- *$S(L)$ is in DA*

S is in DA if, whenever $xyx = x$ for some y, then $x^2 = x$

provides decidability

Dot-depth hierarchy and quantifier alternation

Σ_n, Π_n and $\mathrm{Bool}(\Sigma_n)$ are also characterized algebraically and in terms of the complexity of a star-free expression (dot-depth),

Σ_n: FO formulas in normal prenex form, with n alternations of quantifiers, starting with $\exists$

Dot-depth hierarchy and quantifier alternation

Σ_n, Π_n and $\mathrm{Bool}(\Sigma_n)$ are also characterized algebraically and in terms of the complexity of a star-free expression (dot-depth),

... but for $n \geq 2$ or 3, we do not know any decision algorithm

Σ_n: FO formulas in normal prenex form, with n alternations of quantifiers, starting with $\exists$

Locally testable languages

Locally testable languages: determined by a Boolean combination of conditions on the presence of a finite number of prefixes, factors and suffixes

Locally testable languages

Locally testable languages: determined by a Boolean combination of conditions on the presence of a finite number of prefixes, factors and suffixes

Theorem (Simon, McNaughton, Ladner)
L is locally testable iff $S(L)$ (semigroup version) is locally idempotent and commutative

Locally testable languages

Locally testable languages: determined by a Boolean combination of conditions on the presence of a finite number of prefixes, factors and suffixes

Theorem (Simon, McNaughton, Ladner)
L is locally testable iff $S(L)$ (semigroup version) is locally idempotent and commutative

that is, if $e = e^2$, then $(ese)^2 = ese$ and $esete = etese$: a decidable property

This approach is systematized in Eilenberg's variety theorem

This approach is systematized in Eilenberg's variety theorem

- Often, decidability is possible on the syntactic monoid characterization, and in many important cases, only that way

This approach is systematized in Eilenberg's variety theorem

- Often, decidability is possible on the syntactic monoid characterization, and in many important cases, only that way
- Computing $S(L)$ from $\mathcal{A}(L)$ may lead to exponential blow-up...

This approach is systematized in Eilenberg's variety theorem

- Often, decidability is possible on the syntactic monoid characterization, and in many important cases, only that way
- Computing $S(L)$ from $\mathcal{A}(L)$ may lead to exponential blow-up...
- ... but $S(L)$ is the transition monoid of $\mathcal{A}(L)$, and the algebraic decision procedure can often be *translated* to one on the automaton

This approach is systematized in Eilenberg's variety theorem

- Often, decidability is possible on the syntactic monoid characterization, and in many important cases, only that way
- Computing $S(L)$ from $\mathcal{A}(L)$ may lead to exponential blow-up...
- ... but $S(L)$ is the transition monoid of $\mathcal{A}(L)$, and the algebraic decision procedure can often be *translated* to one on the automaton
- Sometimes with significant benefit: given $\mathcal{A}(L)$, local testability, Σ_2, FO_2 are decided in PTIME

This approach is systematized in Eilenberg's variety theorem

- Often, decidability is possible on the syntactic monoid characterization, and in many important cases, only that way
- Computing $S(L)$ from $\mathcal{A}(L)$ may lead to exponential blow-up...
- ... but $S(L)$ is the transition monoid of $\mathcal{A}(L)$, and the algebraic decision procedure can often be *translated* to one on the automaton
- Sometimes with significant benefit: given $\mathcal{A}(L)$, local testability, Σ_2, FO_2 are decided in PTIME
- but deciding FO is PSPACE-complete (Cho, Huynh)

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Theorem (Doner, Thatcher, Wright)
Let L be a language of Σ-trees. TFAE

- *L is accepted by a bottom-up tree automaton*

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Theorem (Doner, Thatcher, Wright)
Let L be a language of Σ-trees. TFAE

- *L is accepted by a bottom-up tree automaton*
- *L is MSO-definable*

MSO-formulas, interpreted on the nodes, using the parent-child predicate, the order between siblings, and the Σ-labeling

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Theorem (Doner, Thatcher, Wright)
Let L be a language of Σ-trees. TFAE

- *L is accepted by a bottom-up tree automaton*
- *L is MSO-definable*

in a Σ-algebra, each $\sigma \in \Sigma$ defines an operation of arity rank(σ).
T_Σ, the set of all Σ-trees, is a Σ-algebra

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Theorem (Doner, Thatcher, Wright)
Let L be a language of Σ-trees. TFAE

- *L is accepted by a bottom-up tree automaton*
- *L is MSO-definable*
- *L is recognizable by a finite Σ-algebra*

in a Σ-algebra, each $\sigma \in \Sigma$ defines an operation of arity rank(σ). T_Σ, the set of all Σ-trees, is a Σ-algebra

consider morphisms from T_Σ into finite Σ-algebras, or equivalently, finite-index congruences on T_Σ

Σ-labeled trees, where Σ is a ranked alphabet: a node labeled σ has rank(σ) linearly ordered children

Theorem (Doner, Thatcher, Wright)
Let L be a language of Σ-trees. TFAE

- *L is accepted by a bottom-up tree automaton*
- *L is MSO-definable*
- *L is recognizable by a finite Σ-algebra*
- *L is equational*

i.e., L is a component of a solution of a system of equations in the Σ-algebra T_Σ (can be expressed also in terms of context-free grammars)

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Positive points:

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Positive points:

- there is an equivalent notion of generalized rational expressions

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Positive points:

- there is an equivalent notion of generalized rational expressions
- deterministic bottom-up tree automata provide an efficient implementation

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Disappointing points:

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Disappointing points:

- deterministic bottom-up tree automata and finite Σ-algebras are essentially the same thing

- *bottom-up tree automaton*
- *MSO-definability*
- *recognizability by a finite Σ-algebra*
- *equationality*

Disappointing points:

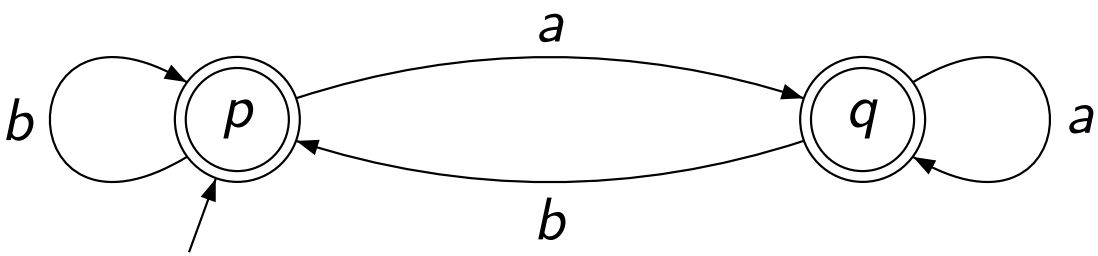
- deterministic bottom-up tree automata and finite Σ-algebras are essentially the same thing
- no classification results: in particular, we do not know how to characterize or decide FO-definable tree languages

Infinite words

Infinite words, $A^{\mathbb{N}}$ $\qquad A^\infty = A^* \cup A^{\mathbb{N}}$

Infinite words

Infinite words, $A^{\mathbb{N}}$ $\qquad A^{\infty} = A^* \cup A^{\mathbb{N}}$

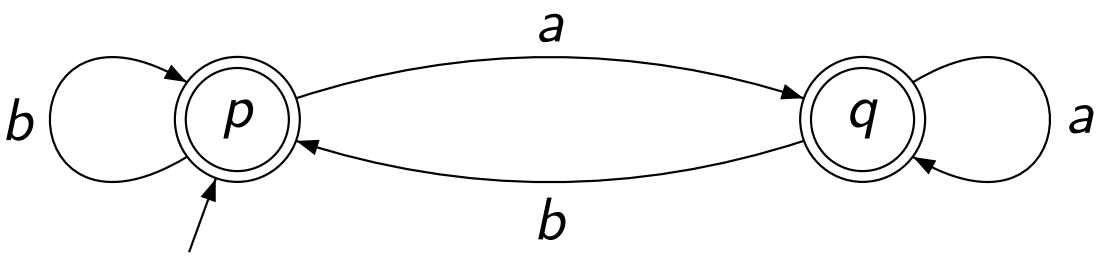


This Büchi automaton accepts all words in $A^{\mathbb{N}}$ with infinitely many a and infinitely many b

Infinite words

Infinite words, $A^{\mathbb{N}}$ $\qquad A^{\infty} = A^* \cup A^{\mathbb{N}}$

Büchi automata, Muller automata, Rabin automata, are equivalent to MSO-definability, and to ω-regular expressions, *i.e.*, finite unions of KL^{ω} (K, L recog. languages of finite words)



This Büchi automaton accepts all words in $A^{\mathbb{N}}$ with infinitely many a and infinitely many b

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations
  - binary finite product $\underline{\textit{finite}} \times \underline{\textit{finite}} \longrightarrow \underline{\textit{finite}}$

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations
  - binary finite product $\underline{\textit{finite}} \times \underline{\textit{finite}} \longrightarrow \underline{\textit{finite}}$
  - binary mixed product $\underline{\textit{finite}} \times \underline{\textit{infinite}} \longrightarrow \underline{\textit{infinite}}$

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations
  - binary finite product $\underline{\textit{finite}} \times \underline{\textit{finite}} \longrightarrow \underline{\textit{finite}}$
  - binary mixed product $\underline{\textit{finite}} \times \underline{\textit{infinite}} \longrightarrow \underline{\textit{infinite}}$
  - unary infinite power $\underline{\textit{finite}} \longrightarrow \underline{\textit{infinite}}$

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations
  - binary finite product $\underline{\textit{finite}} \times \underline{\textit{finite}} \longrightarrow \underline{\textit{finite}}$
  - binary mixed product $\underline{\textit{finite}} \times \underline{\textit{infinite}} \longrightarrow \underline{\textit{infinite}}$
  - unary infinite power $\underline{\textit{finite}} \longrightarrow \underline{\textit{infinite}}$
- Algebraic recognizability is equivalent to rational expressions, Büchi automata and MSO-definability, and

Infinite words: using many-sorted algebras

- The proper algebraic setting, called ω-*semigroups*, emerged more recently (Wilke, Pin, after work by Arnold, Pécuchet, Perrin)
- Consider A^∞ (not $A^{\mathbb{N}}$) and 2 sorts of elements: finite and infinite and 3 operations
  - binary finite product $\textit{finite} \times \textit{finite} \longrightarrow \textit{finite}$
  - binary mixed product $\textit{finite} \times \textit{infinite} \longrightarrow \textit{infinite}$
  - unary infinite power $\textit{finite} \longrightarrow \textit{infinite}$
- Algebraic recognizability is equivalent to rational expressions, Büchi automata and MSO-definability, and
- an Eilenberg-style variety theorem holds, with many interesting classes characterized (Perrin, Pin, Carton)

To study sets of structures (words, trees, posets, graphs, etc)

To study sets of structures (words, trees, posets, graphs, etc)

- we do not always have a model of automata

To study sets of structures (words, trees, posets, graphs, etc)

- we do not always have a model of automata
- MSO formulas are clearly defined once the structures are decided upon

To study sets of structures (words, trees, posets, graphs, etc)

- we do not always have a model of automata
- MSO formulas are clearly defined once the structures are decided upon
- but algebraic recognizability depends on the choice of an algebraic structure

To study sets of structures (words, trees, posets, graphs, etc)

- we do not always have a model of automata
- MSO formulas are clearly defined once the structures are decided upon
- but algebraic recognizability depends on the choice of an algebraic structure

Usually, the structures are given, and we need to invent the abstract algebraic structure: monoids for words, Σ-algebras for Σ-trees, ω-semigroups for A^∞, etc

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...
- allow us to characterize, and hopefully decide significant classes (typically: FO-definability)

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...
- allow us to characterize, and hopefully decide significant classes (typically: FO-definability)

We can put more or less operations in our signature, but more operations give fewer recognizable languages

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...
- allow us to characterize, and hopefully decide significant classes (typically: FO-definability)

We can put more or less operations in our signature, but more operations give fewer recognizable languages

On A^*, consider the usual concatenation product, and the unary mirror operation $a_1 \cdots a_{n-1} a_n \longmapsto a_n a_{n-1} \cdots a_1$, then the class of recognizable sets is unchanged

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...
- allow us to characterize, and hopefully decide significant classes (typically: FO-definability)

We can put more or less operations in our signature, but more operations give fewer recognizable languages

On A^*, consider the usual concatenation product, and the unary mirror operation $a_1 \cdots a_{n-1} a_n \longmapsto a_n a_{n-1} \cdots a_1$, then the class of recognizable sets is unchanged
On the other hand, if we add the unary shift operation $a_1 \cdots a_{n-1} a_n \longmapsto a_n a_1 \cdots a_{n-1}$, then $a^* b$ is not recognizable

What is a good choice of algebraic structure?

Not always clear (posets, graphs)... A *good* algebraic structure should

- match some other specification method: automata, logics,...
- allow us to characterize, and hopefully decide significant classes (typically: FO-definability)

We can put more or less operations in our signature, but more operations give fewer recognizable languages

Too few operations give too many recognizable languages: if there are no operations, then every partition is a congruence, every subset is recognizable

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\mathrm{mod}\ q} x\ \varphi(x)$ where φ is *MSO*

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\bmod q} x\ \varphi(x)$ where φ is *MSO*

For instance, consider $\mathbb{N}^A$, the free commutative semigroup on A, and a typical element, say, $3a + 5b$

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\mathrm{mod}\ q} x\ \varphi(x)$ where φ is *MSO*

For instance, consider $\mathbb{N}^A$, the free commutative semigroup on A, and a typical element, say, $3a + 5b$

- model it by a discrete graph (no edges) with 8 A-labeled vertices

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\mod q} x\ \varphi(x)$ where φ is *MSO*

For instance, consider $\mathbb{N}^A$, the free commutative semigroup on A, and a typical element, say, $3a + 5b$

- model it by a discrete graph (no edges) with 8 A-labeled vertices
  then $L \subseteq \mathbb{N}^A$ is MSO-definable iff L is finite or cofinite – and L is CMSO-definable iff L is recognizable

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\bmod q} x\ \varphi(x)$ where φ is *MSO*

For instance, consider $\mathbb{N}^A$, the free commutative semigroup on A, and a typical element, say, $3a + 5b$

- model it by a discrete graph (no edges) with 8 A-labeled vertices
  then $L \subseteq \mathbb{N}^A$ is MSO-definable iff L is finite or cofinite – and L is CMSO-definable iff L is recognizable
- model it by a pair of words (chains) (a^3, b^5) (dependence graph in trace theory)

And what is a good choice of logic?

- Obvious candidate: monadic second order logic (MSO)
- Counting MSO (CMSO) is better. CMSO = MSO extended with $\exists^{\mathrm{mod}\ q} x\ \varphi(x)$ where φ is *MSO*

For instance, consider $\mathbb{N}^A$, the free commutative semigroup on A, and a typical element, say, $3a + 5b$

- model it by a discrete graph (no edges) with 8 A-labeled vertices
  then $L \subseteq \mathbb{N}^A$ is MSO-definable iff L is finite or cofinite – and L is CMSO-definable iff L is recognizable
- model it by a pair of words (chains) (a^3, b^5) (dependence graph in trace theory)
  then L is MSO-definable iff L is recognizable

Very often, CMSO-definability implies recognizability

Once all the ingredients are in place,

Very often, CMSO-definability implies recognizability

Once all the ingredients are in place,

Theorem (Courcelle)
Under appropriate conditions, CMSO-def $\Longrightarrow$ recognizability

Very often, CMSO-definability implies recognizability

Once all the ingredients are in place,

Theorem (Courcelle)
Under appropriate conditions, CMSO-def $\Longrightarrow$ recognizability

Appropriate conditions:
the algebra S must be *finitely generated*, so there is a finite Σ s.t. every $s \in S$ is described by a Σ-term (Σ-tree)
and the valuation map val: Terms $\longrightarrow S$ must be *MS-definable*

Very often, CMSO-definability implies recognizability

Once all the ingredients are in place,

Theorem (Courcelle)
Under appropriate conditions, CMSO-def $\Longrightarrow$ recognizability

Appropriate conditions:
the algebra S must be *finitely generated*, so there is a finite Σ s.t. every $s \in S$ is described by a Σ-term (Σ-tree)
and the valuation map val: Terms $\to S$ must be *MS-definable*

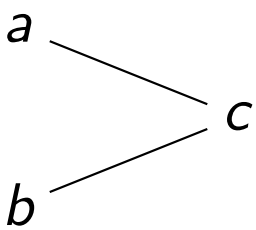
Converse:
if we have a *MS-definable parsing* mapping, parse: $S \to$ Terms such that val(parse(x)) $= x$, then recognizability implies CMSO-definability

Traces

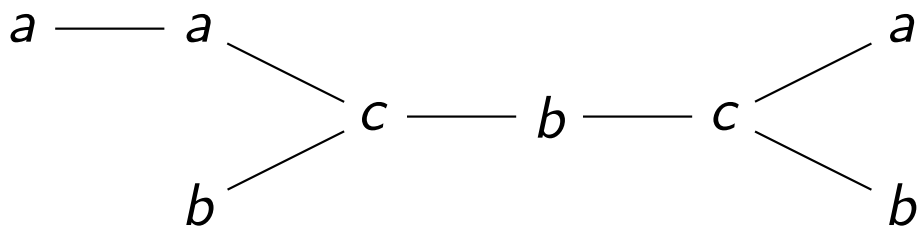
- The alphabet is equipped with a dependence relation, and a trace is a dependence graph

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph



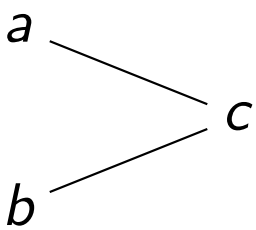
alphabet



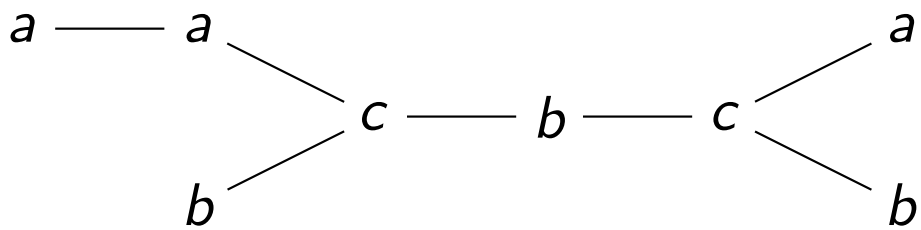
the trace $a^2bcbcab = abacbcba$

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph
- the algebraic structure is the so-called partially commutative monoid



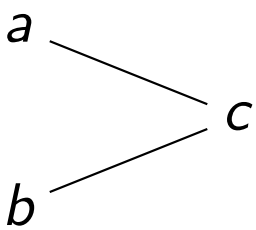
alphabet



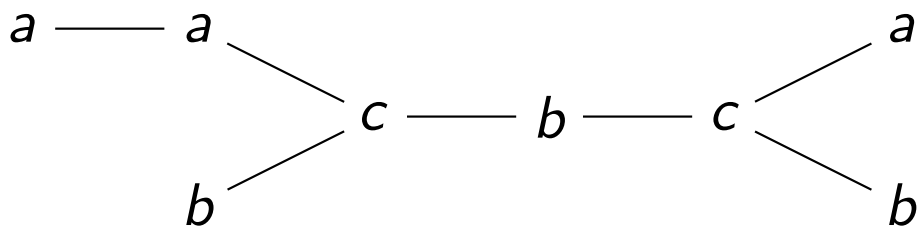
the trace $a^2bcbcab = abacbcba$

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph
- the algebraic structure is the so-called partially commutative monoid
- Recognizability is equivalent to MSO-definability, but not to rationality ($(ab)^*$ is not recognizable)



alphabet



the trace $a^2bcbcab = abacbcba$

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph
- the algebraic structure is the so-called partially commutative monoid
- Recognizability is equivalent to MSO-definability, but not to rationality ($(ab)^*$ is not recognizable)
- Recognizability is equivalent to Zielonka's automata

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph
- the algebraic structure is the so-called partially commutative monoid
- Recognizability is equivalent to MSO-definability, but not to rationality ($(ab)^*$ is not recognizable)
- Recognizability is equivalent to Zielonka's automata
- FO-definability is characterized by algebraic means, and decidable (Guaiana, Restivo, Salemi, Ebinger, Muscholl)

Traces

- The alphabet is equipped with a dependence relation, and a trace is a dependence graph
- the algebraic structure is the so-called partially commutative monoid
- Recognizability is equivalent to MSO-definability, but not to rationality ($(ab)^*$ is not recognizable)
- Recognizability is equivalent to Zielonka's automata
- FO-definability is characterized by algebraic means, and decidable (Guaiana, Restivo, Salemi, Ebinger, Muscholl)
- but few other classes of recognizable trace languages have been characterized that way...

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the N-free posets

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the *N*-free posets
- natural algebraic structure: 2 binary associative operations, one of which is commutative

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the N-free posets
- natural algebraic structure: 2 binary associative operations, one of which is commutative
- Recognizability is equivalent to CMSO-definability (Kuske)

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the N-free posets
- natural algebraic structure: 2 binary associative operations, one of which is commutative
- Recognizability is equivalent to CMSO-definability (Kuske)
- FO-definability is characterized algebraically, but only for bounded-width languages (Kuske)

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the N-free posets
- natural algebraic structure: 2 binary associative operations, one of which is commutative
- Recognizability is equivalent to CMSO-definability (Kuske)
- FO-definability is characterized algebraically, but only for bounded-width languages (Kuske)
- an automaton model is known to match recognizability under the same hypothesis (Lodaya, Weil)

Series-parallel pomsets and Message Sequence Charts

- *sp*-posets are obtained from singletons by using parallel and sequential products – characterized as the N-free posets
- natural algebraic structure: 2 binary associative operations, one of which is commutative
- Recognizability is equivalent to CMSO-definability (Kuske)

- For MSCs, recognizability has been considered only with reference to a monoid structure (i.e. considering linearizations), and several logics have been proposed

An algebraic structure for the set of finite graphs?

No obvious choice!

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars

- HR and VR are infinite signatures, defining an infinitely-sorted algebra.
- HR and VR admit many technical variants, but the corresponding notions of recognizability are stable

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars
- The modular signature $\mathcal{F}_\infty$ comes from the modular decomposition of graphs.

- HR and VR are infinite signatures, defining an infinitely-sorted algebra.
- HR and VR admit many technical variants, but the corresponding notions of recognizability are stable

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars
- The modular signature $\mathcal{F}_\infty$ comes from the modular decomposition of graphs.

- HR and VR are infinite signatures, defining an infinitely-sorted algebra.
- HR and VR admit many technical variants, but the corresponding notions of recognizability are stable
- $\mathcal{F}_\infty$ is also an infinite signature, and it defines a one-sorted algebra

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars
- The modular signature $\mathcal{F}_\infty$ comes from the modular decomposition of graphs.

2 options for logic:

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars
- The modular signature $\mathcal{F}_\infty$ comes from the modular decomposition of graphs.

2 options for logic:

- CMSO[E] – the domain is V (vertices), and E is the binary edge predicate

An algebraic structure for the set of finite graphs?

No obvious choice!

3 signatures arise from the literature: modular, HR, VR

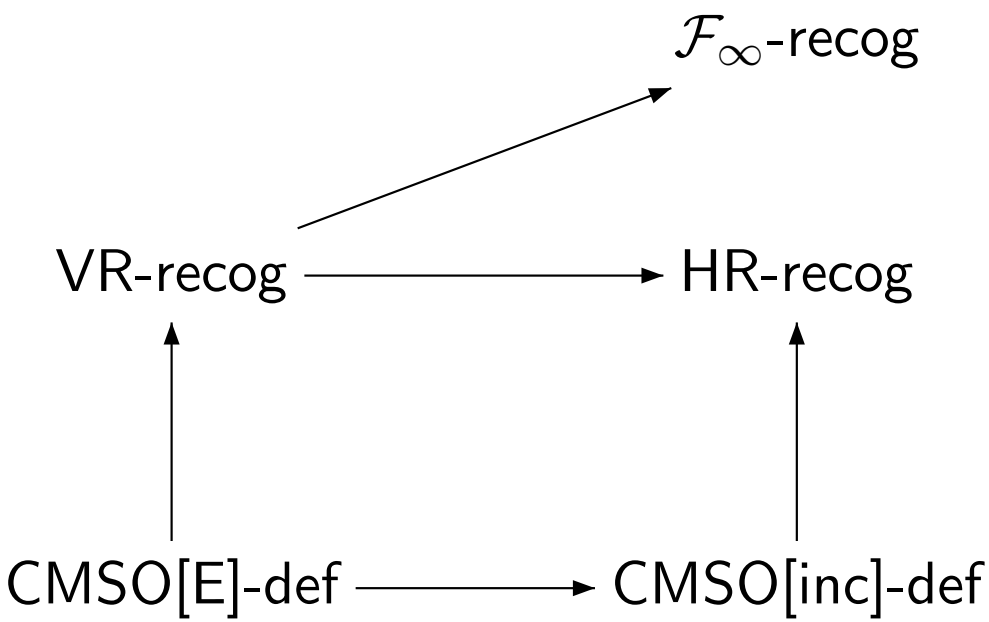
- VR comes from vertex-replacement grammars
- HR comes from hyperedge replacement grammars
- The modular signature $\mathcal{F}_\infty$ comes from the modular decomposition of graphs.

2 options for logic:

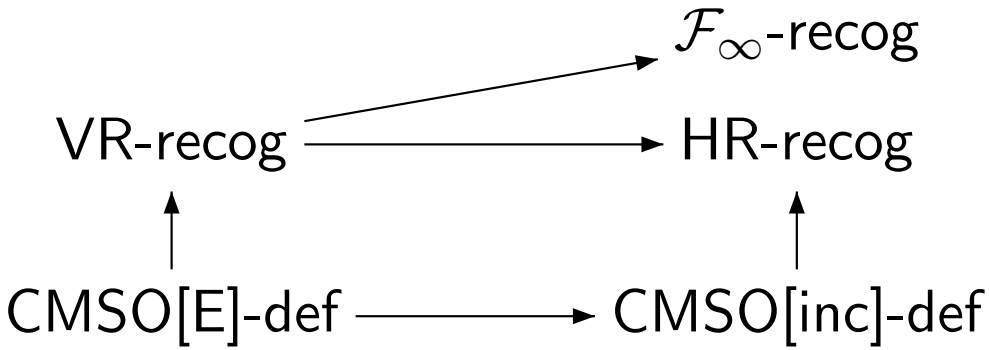
- CMSO[E] – the domain is V (vertices), and E is the binary edge predicate
- CMSO[inc] – the domain is 2-sorted (V, E), and inc is the incidence predicate

General implications

The following implications are known, and strict (Courcelle, Weil)

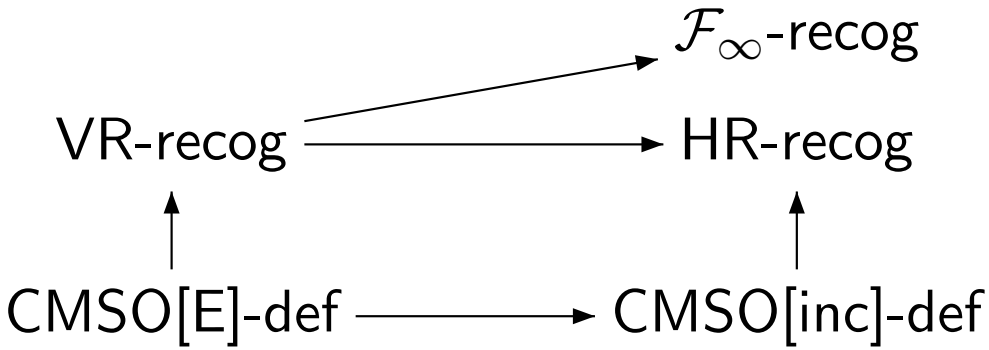


Finiteness conditions yield equivalence results



Under certain finiteness conditions, which can be interpreted in terms of finitely generated algebras, we get equivalence results

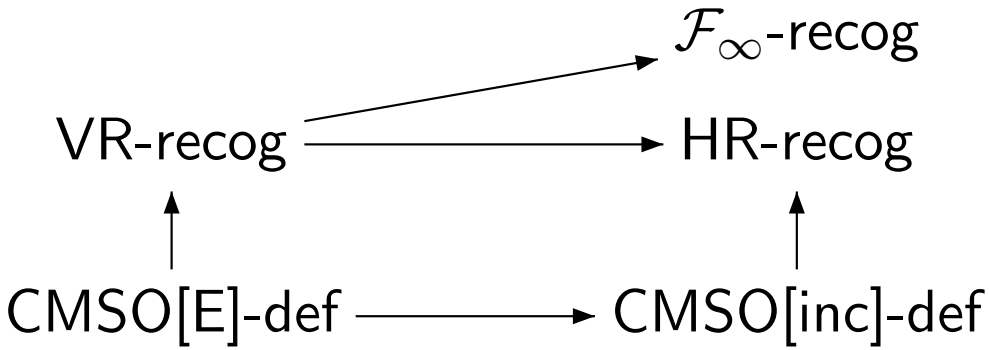
Finiteness conditions yield equivalence results



- For bounded tree-width, CMSO[inc] $\Longleftrightarrow$ HR-recog (Lapoire)

Under certain finiteness conditions, which can be interpreted in terms of finitely generated algebras, we get equivalence results

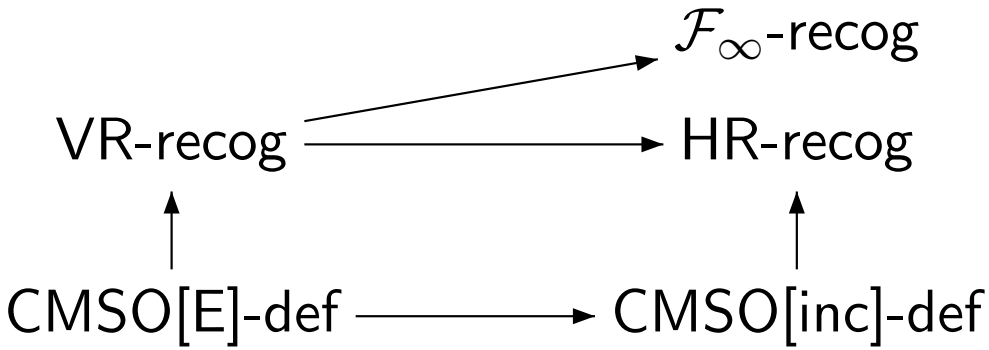
Finiteness conditions yield equivalence results



- For bounded tree-width, CMSO[inc] $\Longleftrightarrow$ HR-recog (Lapoire)
- For graphs without $\vec{K}_{n,n}$, VR-recog $\Longleftrightarrow$ HR-recog (Courcelle, Weil)

Under certain finiteness conditions, which can be interpreted in terms of finitely generated algebras, we get equivalence results

Finiteness conditions yield equivalence results

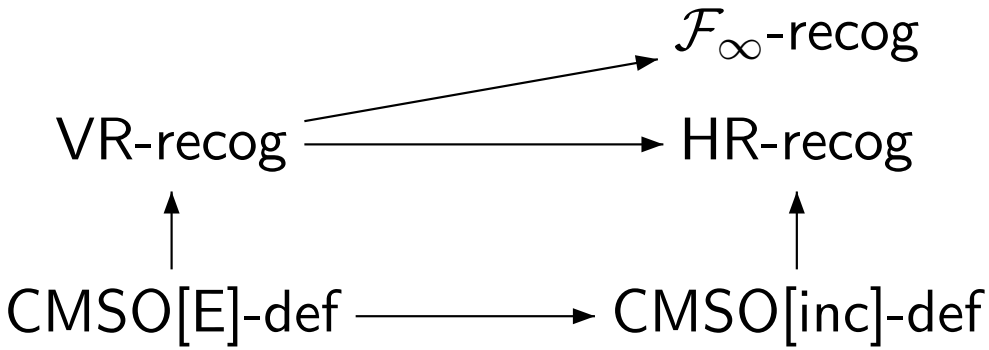


- For bounded tree-width, CMSO[inc] $\Longleftrightarrow$ HR-recog (Lapoire)
- For graphs without $\vec{K}_{n,n}$, VR-recog $\Longleftrightarrow$ HR-recog (Courcelle, Weil)

Let $\mathcal{F}$ be a finite subset of $\mathcal{F}_\infty$. For $\mathcal{F}$-graphs,

- VR-recog $\Longleftrightarrow$ $\mathcal{F}$-recog (Courcelle, Weil)

Finiteness conditions yield equivalence results



- For bounded tree-width, CMSO[inc] $\Longleftrightarrow$ HR-recog (Lapoire)
- For graphs without $\vec{K}_{n,n}$, VR-recog $\Longleftrightarrow$ HR-recog (Courcelle, Weil)

Let $\mathcal{F}$ be a finite subset of $\mathcal{F}_\infty$. For $\mathcal{F}$-graphs,

- VR-recog $\Longleftrightarrow$ $\mathcal{F}$-recog (Courcelle, Weil)
- and if $\mathcal{F}$ is weakly rigid, this is equivalent to CMSO[E]-definability (Weil)

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so
- with Z. Esik, we proposed a new algebraic structure to discuss Σ-trees, called preclones

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so
- with Z. Esik, we proposed a new algebraic structure to discuss Σ-trees, called preclones
- Unfortunately more complex than the Σ-algebras, infinitely sorted (and trees represent just one sort of elements)

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so
- with Z. Esik, we proposed a new algebraic structure to discuss Σ-trees, called preclones
- Unfortunately more complex than the Σ-algebras, infinitely sorted (and trees represent just one sort of elements)
- but the class of recognizable tree languages is not modified (no cheating, we are talking of the same tree languages)

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so
- with Z. Esik, we proposed a new algebraic structure to discuss Σ-trees, called preclones
- Unfortunately more complex than the Σ-algebras, infinitely sorted (and trees represent just one sort of elements)
- but the class of recognizable tree languages is not modified (no cheating, we are talking of the same tree languages)
- FO-definable sets of trees are now characterized algebraically, by their syntactic preclone

Revisiting trees

- Σ-algebras have not given a tool to characterize or decide FO-definability for languages of Σ-trees, so
- with Z. Esik, we proposed a new algebraic structure to discuss Σ-trees, called preclones
- Unfortunately more complex than the Σ-algebras, infinitely sorted (and trees represent just one sort of elements)
- but the class of recognizable tree languages is not modified (no cheating, we are talking of the same tree languages)
- FO-definable sets of trees are now characterized algebraically, by their syntactic preclone
- the characterization does not yet prove decidability. . .

Conclusion

- In the case of word languages, monoid theory is a great tool for classification and decision

Conclusion

- In the case of word languages, monoid theory is a great tool for classification and decision
- For languages of other discrete objects, automata are not always available, it is a tool to systematically discuss the decidability of logically defined and other classes.

Conclusion

- In the case of word languages, monoid theory is a great tool for classification and decision
- For languages of other discrete objects, automata are not always available, it is a tool to systematically discuss the decidability of logically defined and other classes.
- Problem: to identify the appropriate algebraic structure; must match a natural logic or automaton model, and allow the characterization of significant subclasses

Conclusion

- In the case of word languages, monoid theory is a great tool for classification and decision
- For languages of other discrete objects, automata are not always available, it is a tool to systematically discuss the decidability of logically defined and other classes.
- Problem: to identify the appropriate algebraic structure; must match a natural logic or automaton model, and allow the characterization of significant subclasses
- Wherever it works (e.g. infinite words), it provides a clarification of complex algorithms or difficult results

Conclusion

- In the case of word languages, monoid theory is a great tool for classification and decision
- For languages of other discrete objects, automata are not always available, it is a tool to systematically discuss the decidability of logically defined and other classes.
- Problem: to identify the appropriate algebraic structure; must match a natural logic or automaton model, and allow the characterization of significant subclasses
- Wherever it works (e.g. infinite words), it provides a clarification of complex algorithms or difficult results
- Unavoidably very complex if we want to discuss graph or poset languages without restrictions, but adding finiteness conditions often helps

Thank you for your attention!