

The separation problem

Thomas Place and Marc Zeitoun

LaBRI, Université Bordeaux, CNRS

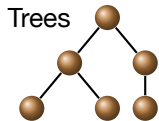
May 23, 2014

Séminaire Automates, LIAFA

Objects we consider

Structure

Words
ababcbaa



Descriptive Formalism

First-Order Logic (**FO**)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables **FO** (\mathbf{FO}_2)

Fragments $\Sigma_i, \mathcal{B}\Sigma_i$

Locally Testable (**LT**)

Objects we consider

Structure

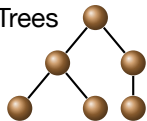
Descriptive Formalism

Express Properties

Words

ababcbaa

Trees



First-Order Logic (**FO**)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables **FO** (\mathbf{FO}_2)

Fragments $\Sigma_i, \mathcal{B}\Sigma_i$

Locally Testable (**LT**)

Objects we consider

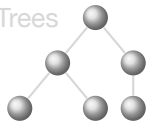
Structure

Descriptive Formalism

Express Properties

Words
ababcbaa

Trees



First-Order Logic (FO)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables FO (FO_2)

Fragments $\Sigma_i, \mathcal{B}\Sigma_i$

Locally Testable (LT)

For this talk

First-Order Logic for Words

We consider first-order logic with only the linear order '<.'

a b b b c a a a c a

First-Order Logic for Words

We consider first-order logic with only the linear order ' $<$ '.

<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>a</i>
0	1	2	3	4	5	6	7	8	9

- ▶ A word is as a sequence of labeled positions that can be quantified.
- ▶ Unary predicates $a(x), b(x), c(x), \dots$ testing the label of a position.
- ▶ One binary predicate: the linear-order $x < y$.

First-Order Logic for Words

We consider first-order logic with only the linear order ' $<$ '.

<i>a</i>	<i>b</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>a</i>	<i>c</i>	<i>a</i>
0	1	2	3	4	5	6	7	8	9

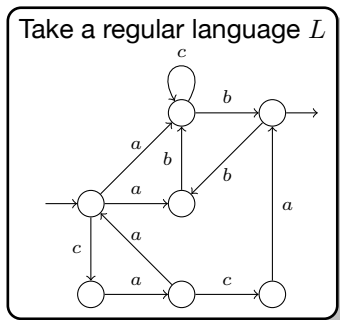
- ▶ A word is as a sequence of labeled positions that can be quantified.
- ▶ Unary predicates $a(x), b(x), c(x), \dots$ testing the label of a position.
- ▶ One binary predicate: the linear-order $x < y$.

Example: every a comes after some b

$$\forall x a(x) \Rightarrow \exists y (b(y) \wedge (y < x))$$

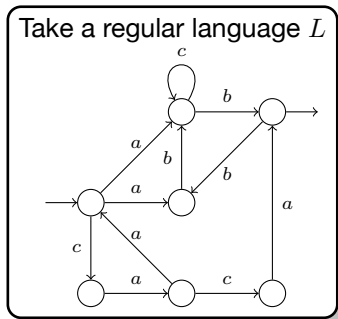
First Problem: Decidable Characterizations

Decide the following problem:



First Problem: Decidable Characterizations

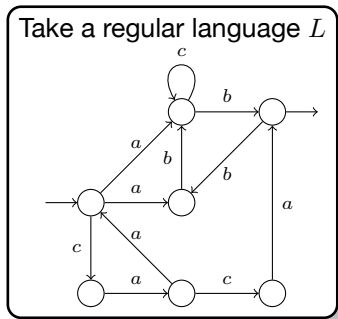
Decide the following problem:



Can it be defined with an **FO** formula?

First Problem: Decidable Characterizations

Decide the following problem:



Can it be defined
with an **FO** formula?

Schützenberger'65, McNaughton and Papert'71

For L a regular language, the following are equivalent:

- ▶ L is **FO**-definable.
- ▶ The syntactic monoid of L satisfies $u^{\omega+1} = u^{\omega}$.

Why we want more than decidable characterizations

If the characterization answer is yes for L :

- ▶ All subparts of the minimal automaton of L are **FO**-definable.

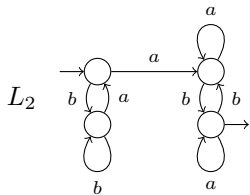
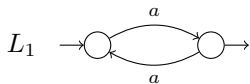
If the characterization's answer is no for L :

- ▶ We have little information.
- ▶ Defining L would require differentiating some u^ω and $u^{\omega+1}$.
- ▶ **Yet**: the logic can still express facts on L .

Separation

Decide the following problem:

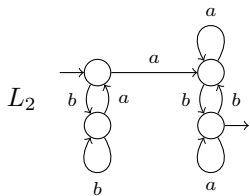
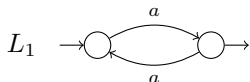
Take two regular languages L_1, L_2



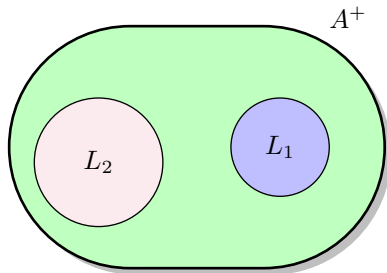
Separation

Decide the following problem:

Take two regular languages L_1, L_2



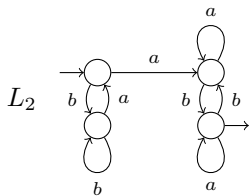
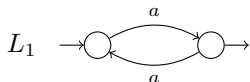
Can L_1 be separated from L_2 with an **FO** formula?



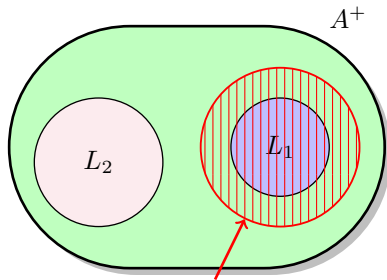
Separation

Decide the following problem:

Take two regular languages L_1, L_2



Can L_1 be separated from L_2 with an **FO** formula?

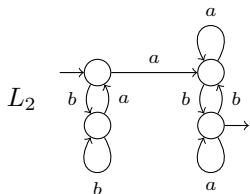
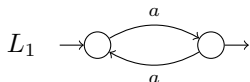


FO-definable

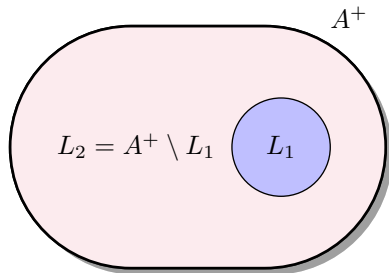
Separation

Characterization can be formally reduced to separation

Take two regular languages L_1, L_2



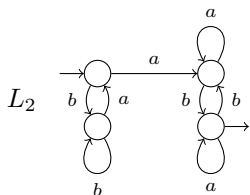
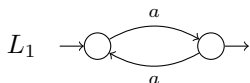
Can L_1 be separated from L_2 with an **FO** formula?



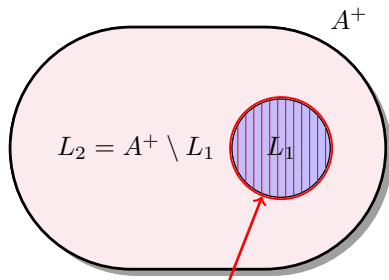
Separation

Characterization can be formally reduced to separation

Take two regular languages L_1, L_2



Can L_1 be separated from L_2 with an **FO** formula?



FO-separable from complement

\Leftrightarrow

FO-definable

Motivations for Separation

- ▶ More general: need **FO** techniques applying to **all** languages.

Motivations for Separation

- ▶ More general: need **FO** techniques applying to **all** languages.
- ▶ Therefore, may give information to solve harder problems.

Motivations for Separation

- ▶ More general: need **FO** techniques applying to **all** languages.
- ▶ Therefore, may give information to solve harder problems.
- ▶ For FO, already solved with such motivations by Henckell '88.

Pointlike sets: the finest aperiodic cover of a finite semigroup

The research in this paper is motivated by the open question: "Is the complexity of a finite semigroup S decidable?"

Motivations for Separation

- ▶ More general: need **FO** techniques applying to **all** languages.
- ▶ Therefore, may give information to solve harder problems.
- ▶ For FO, already solved with such motivations by Henckell '88.

Pointlike sets: the finest aperiodic cover of a finite semigroup

The research in this paper is motivated by the open question: "Is the complexity of a finite semigroup S decidable?"

- ▶ **Difficult algebraic techniques.** Following the lead of the Presentation Lemma (Rhodes), we describe the finest cover on S that can be computed using an aperiodic semigroup and give an explicit relation. The central idea of the proof is that an aperiodic computation can be described by a new 'blow-up operator' HW. The proof also relies on the Rhodes expansion of S and on Zeiger coding.

An already known result: Henckell '88

Guide to the paper

Chapter 1. Elementary definitions and notation should be omitted on first reading and used as a reference as needed.

Chapter 2. The Pl-functor defines pointlike sets in a general setting and shows by an abstract compactness argument that $\text{Pl}(S)$ can be computed by an aperiodic semigroup.

Chapter 3. Definition of $C^\omega(S)$ and H^ω defines $C^\omega(S)$, a collection of pointlike sets, in a constructive manner. H^ω is the ‘blow-up-operator’ that we will use in Chapter 5 to show $C^\omega(S) = \text{Pl}(S)$. It has some examples in the end.

Chapter 4. The Rhodes-expansion defines the tools needed in Chapter 5.

Chapter 5. $C^\omega(S) = \text{Pl}(S)$ shows the main result by actually constructing a relation $S \xrightarrow{R} \text{CP}(S)$ computing $C^\omega(S)$ with $\text{CP}(S)$ aperiodic. It uses H^ω , generalized to \hat{H}^ω on $\hat{C}^\omega(S)$ ‘to get rid of groups by blowing up’.

Alternate formulations of separation

Formulations of separation [Almeida'96] [AlmeidaCostaZ13]

The following are equivalent:

1. L_1 and L_2 are not FO-separable.
2. For all k , there exist $w_1 \in L_1, w_2 \in L_2$ with $w_1 \cong_k w_2$.
3. For all aperiodic T and morphism $\beta : A^+ \rightarrow T$,

$$\beta(L_1) \cap \beta(L_2) \neq \emptyset.$$

4. $\bar{L}_1 \cap \bar{L}_2 \neq \emptyset$ (closures taken in the pro-aperiodic semigroup).
5. $\bar{L}_1 \cap \bar{L}_2$ contains an ω -term.

Actually, 5 may be exploited to prove decidability of separation.

What was already known

Separation **already solved** for interesting classes:

- ▶ First-order definable,
- ▶ Piecewise testable,
- ▶ Locally (threshold) testable.

Drawbacks:

- ▶ Difficult algebraic proofs.
- ▶ No insight of an actual separator.

Contributions

- ▶ New, **elementary** proofs for already known cases.
- ▶ **Single** "proof canvas".
- ▶ Extension to other logics (FO^2 , quantifier alternation within FO^2).
- ▶ Generalization to logics not closed under negation: Σ_2 .
- ▶ Transfer theorems: **separation for $\Sigma_2 \Rightarrow$ characterization for Σ_3** .
- ▶ Results can be lifted to the profinite interpretation.

Canvas: given $\alpha : A^+ \rightarrow S$, fixpoint algorithm answering separability
for all input languages recognized by α .

FO is hard, let's make it easy: Quantifier Rank

Quantifier rank of a formula: Nested depth of quantifiers.

$\forall x \exists y (a(x) \implies \exists z (x < z < y \wedge b(y)))$ has quantifier rank 3

If k fixed: finitely many **FO** properties of rank k
 \implies Separation is easy (test them all).

FO is hard, let's make it easy: Quantifier Rank

Quantifier rank of a formula: Nested depth of quantifiers.

$\forall x \exists y (a(x) \implies \exists z (x < z < y \wedge b(y)))$ has quantifier rank 3

If k fixed: finitely many **FO** properties of rank k
 \implies Separation is easy (test them all).

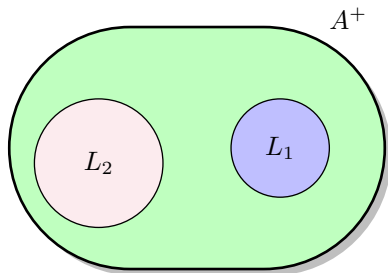
k -equivalence for **FO**

Let w_1, w_2 be words:

$w_1 \cong_k w_2$ iff w_1, w_2 satisfy the same formulas of rank k

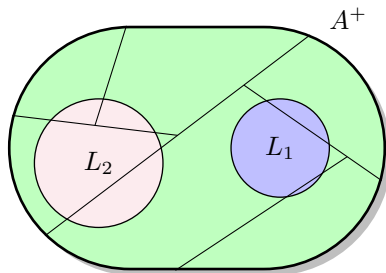
All **FO** properties of rank k are unions of classes of \cong_k .

Fixed Quantifier Rank k



Let's add the \cong_k -classes

Fixed Quantifier Rank k

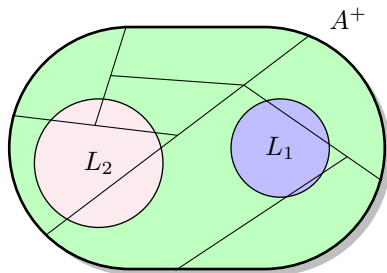


Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

\Rightarrow Compute a 'limit' for \cong_k .

Fixed Quantifier Rank k



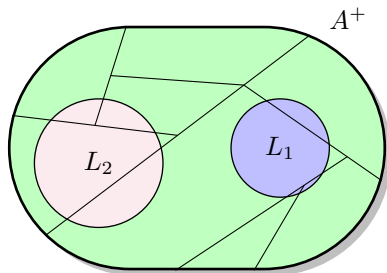
Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

⇒ Compute a 'limit' for \cong_k .

When k gets larger, \cong_k is refined but it never ends

Fixed Quantifier Rank k



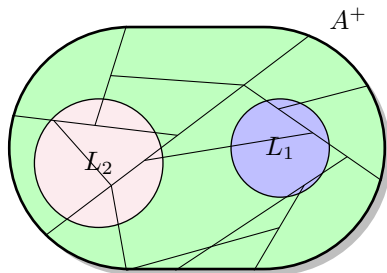
Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

\Rightarrow Compute a 'limit' for \cong_k .

When k gets larger, \cong_k is refined but it never ends

Fixed Quantifier Rank k



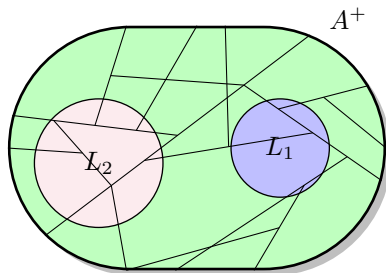
Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

⇒ Compute a 'limit' for \cong_k .

When k gets larger, \cong_k is refined but it never ends

Fixed Quantifier Rank k



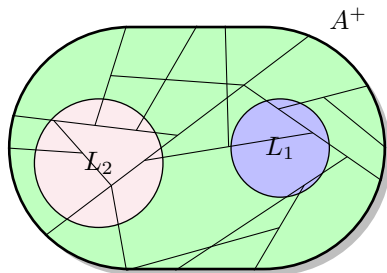
Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

\Rightarrow Compute a 'limit' for \cong_k .

When k gets larger, \cong_k is refined but it never ends

Fixed Quantifier Rank k



Separable with rank k iff no \cong_k -class intersects both languages

For full **FO** we want to know if there exists such a k

⇒ Compute a 'limit' for \cong_k .

When k gets larger, \cong_k is refined but it never ends

Idea. Abstract \cong_k on a **finite** semigroup recognizing both L_1 and L_2 .

"Pair" analysis

Fix $\alpha : A^+ \rightarrow S$. Compute $I_k[\alpha]$, k -indistinguishable pairs.

FO-indistinguishable **pairs** for $\alpha : A^+ \rightarrow S$

$(s_1, s_2) \in I_k[\alpha]$ if

$$\exists \begin{array}{ccc} w_1 & \cong_k & w_2 \\ \alpha \downarrow & & \alpha \downarrow \\ s_1 & & s_2 \end{array}$$

- ▶ Smaller and smaller sets: $I_{k+1}[\alpha] \subseteq I_k[\alpha]$.
- ▶ Limit set: $I[\alpha] = \bigcap_k I_k[\alpha]$.
- ▶ Computing these pairs solves separation:

$$(s_1, s_2) \in I[\alpha] \iff \alpha^{-1}(s_1) \text{ and } \alpha^{-1}(s_2) \text{ not separable}$$

"Pair" analysis

- ▶ Smaller and smaller sets: $I_{k+1}[\alpha] \subseteq I_k[\alpha]$
- ▶ Limit set: $I[\alpha] = \bigcap_k I_k[\alpha]$.

What have we gained?


We work with **finite semigroups** \Rightarrow the refinement stabilizes.

"Pair" analysis

- ▶ Smaller and smaller sets: $I_{k+1}[\alpha] \subseteq I_k[\alpha]$
- ▶ Limit set: $I[\alpha] = \bigcap_k I_k[\alpha]$.

What have we gained?

We work with **finite semigroups** \Rightarrow the refinement stabilizes.

 It may happen that $I_{k+1}[\alpha] = I_k[\alpha]$ **before stabilization**.

It may happen that

- ▶ $(r, s) \in I[\alpha]$,
- ▶ $(s, t) \in I[\alpha]$,
- ▶ but $(r, t) \notin I[\alpha]$ (no transitivity).

The Separation Criterion

Separation Criterion

L_1, L_2 recognized by $\alpha : A^+ \rightarrow S$ are **not** separable
iff

there are accepting elements $s_1, s_2 \in S$ for L_1, L_2 s.t. $(s_1, s_2) \in I[\alpha]$.

The Separation Criterion

Separation Criterion

L_1, L_2 recognized by $\alpha : A^+ \rightarrow S$ are **not** separable
iff

there are accepting elements $s_1, s_2 \in S$ for L_1, L_2 s.t. $(s_1, s_2) \in I[\alpha]$.

Computing $I[\alpha]$ suffices to solve separation.

Two approaches to compute $I[\alpha]$

Brute-force

- ▶ Computing $I_k[\alpha]$ easy for fixed k .
- ▶ $I[\alpha] = I_k[\alpha]$ for k depending on α .
- ▶ \Rightarrow Prove a bound $k = f(\alpha)$,
Compute $I_k[\alpha]$.

Algorithm

Algorithm bypassing the bound k :
Direct fixpoint computation of $I[\alpha]$.

Two approaches to compute $I[\alpha]$

Brute-force

- ▶ Computing $I_k[\alpha]$ easy for fixed k .
- ▶ $I[\alpha] = I_k[\alpha]$ for k depending on α .
- ▶ \Rightarrow Prove a bound $k = f(\alpha)$,
Compute $I_k[\alpha]$.

Algorithm

Algorithm bypassing the bound k :
Direct fixpoint computation of $I[\alpha]$.

We use approach 2.

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

1st Property of **FO**

$$w \cong_k w$$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

1st Property of **FO**

$$w \cong_k w$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

2nd Property of **FO**

$$w_1 \cong_k w_2 \text{ and } u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

2nd Property of **FO**

$$w_1 \cong_k w_2 \text{ and } u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$
2. Operation \bullet : $(s_1, s_2) \in I[\alpha]$ and $(t_1, t_2) \in I[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in I[\alpha]$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$\forall k \exists n \forall w_1, w_2 \in A^+ \quad w_1 \cong_k w_2 \Rightarrow (w_1)^n \cong_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$
2. Operation \bullet : $(s_1, s_2) \in I[\alpha]$ and $(t_1, t_2) \in I[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in I[\alpha]$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$\forall k \exists n \forall w_1, w_2 \in A^+ \quad w_1 \cong_k w_2 \Rightarrow (w_1)^n \cong_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$
2. Operation \bullet : $(s_1, s_2) \in I[\alpha]$ and $(t_1, t_2) \in I[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in I[\alpha]$
3. Operation ω : $(s_1, s_2) \in I[\alpha] \Rightarrow (s_1^\omega, s_2^{\omega+1}) \in I[\alpha]$

A first (non complete) Algorithm computing $I[\alpha]$

Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$\forall k \exists n \forall w_1, w_2 \in A^+ \quad w_1 \cong_k w_2 \Rightarrow (w_1)^n \cong_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^+$ $(\alpha(w), \alpha(w)) \in I[\alpha]$
2. Operation \bullet : $(s_1, s_2) \in I[\alpha]$ and $(t_1, t_2) \in I[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in I[\alpha]$
3. Operation ω : $(s_1, s_2) \in I[\alpha] \Rightarrow (s_1^\omega, s_2^{\omega+1}) \in I[\alpha]$

Correct by definition but not complete

Why it does not work

3rd Property of **FO**

$$w_1 \cong_k w_2 \Rightarrow (w_1)^n \cong_k (w_2)^{n+1}$$

Why it does not work

Not general enough

3rd Property of **FO**

$$w_1 \cong_k w_2 \Rightarrow (w_1)^n \cong_k (w_2)^{n+1}$$

Needs to be replaced

$$w_1 \cong_k w_2 \cong_k \cdots \cong_k w_m$$

All large concatenations of words in $\{w_1, \dots, w_m\}$ are \cong_k -equivalent.

Need for better analysis

A Generalization: FO-indistinguishable **Sets** for $\alpha : A^+ \rightarrow S$:

- ▶ $\{s_1, s_2, \dots, s_n\} \in I_k[\alpha]$ if

$$\begin{array}{ccccccc} \exists & & w_1 & \cong_k & w_2 & \dots & \cong_k & w_n \\ & & \downarrow \alpha & & \downarrow \alpha & & \downarrow \alpha & \\ & & s_1 & & s_2 & \dots & & s_n \end{array}$$

- ▶ Limit set: $I[\alpha] = \bigcap_k I_k[\alpha]$.
- ▶ Computing these **sets** is more general than computing pairs.
⇒ also solves separation (and gives much more).

From Pairs to Sets

New Objective

We want to compute the set $I[\alpha] \subseteq 2^S$ such that:

$$T \in I[\alpha] \text{ iff } T \in I_k[\alpha], \forall k \in \mathbb{N}$$

From Pairs to Sets

New Objective

We want to compute the set $I[\alpha] \subseteq 2^S$ such that:

$$T \in I[\alpha] \text{ iff } T \in I_k[\alpha], \forall k \in \mathbb{N}$$

Remark

- ▶ With our new definition, we have $I[\alpha] \subseteq 2^S$.
- ▶ 2^S is a semigroup for the operation $T_1 \cdot T_2 = \{t_1 t_2 \mid t_1 \in T_1, t_2 \in T_2\}$.

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

1st Property of **FO**

$$w \cong_k w$$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

1st Property of **FO**

$$w \cong_k w$$

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

2nd Property of **FO**

$$w_1 \cong_k w_2 \text{ and } u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$$

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

2nd Property of **FO**

$$w_1 \cong_k w_2 \text{ and } u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$$

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$
2. Operation \bullet : $T_1 \in I[\alpha]$ and $T_2 \in I[\alpha] \Rightarrow T_1 T_2 \in I[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \cong_k w_2 \cdots \cong_k w_m$$



All large concatenations of words in $\{w_1, \dots, w_m\}$ are \cong_k -equivalent.

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$
2. Operation \bullet : $T_1 \in I[\alpha]$ and $T_2 \in I[\alpha] \Rightarrow T_1 T_2 \in I[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \cong_k w_2 \cdots \cong_k w_m$$



All large concatenations of words in $\{w_1, \dots, w_m\}$ are \cong_k -equivalent.

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in \mathbb{I}[\alpha]$
2. Operation \bullet : $T_1 \in \mathbb{I}[\alpha]$ and $T_2 \in \mathbb{I}[\alpha] \Rightarrow T_1 T_2 \in \mathbb{I}[\alpha]$
3. Operation ω : $T \in \mathbb{I}[\alpha] \Rightarrow (T^\omega \cup T^{\omega+1}) \in \mathbb{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \cong_k w_2 \cdots \cong_k w_m$$



All large concatenations of words in $\{w_1, \dots, w_m\}$ are \cong_k -equivalent.

1. Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$
2. Operation \bullet : $T_1 \in I[\alpha]$ and $T_2 \in I[\alpha] \Rightarrow T_1 T_2 \in I[\alpha]$
3. Operation ω : $T \in I[\alpha] \Rightarrow (T^\omega \cup T^{\omega+1}) \in I[\alpha]$

Correct by definition (e.g., use EF games)

Can be proved to be complete

Alternate algorithms

- ▶ The algorithm reflects the equation $x^\omega = x^{\omega+1}$
- ▶ 2 other equivalent ways to characterize **FO**-definability:
 - ▶ All groups are trivial.
 - ▶ All \mathcal{H} -classes are trivial.
- ▶ The algorithm can be modified to reflect them too:

Alternate algorithms

- ▶ The algorithm reflects the equation $x^\omega = x^{\omega+1}$
- ▶ 2 other equivalent ways to characterize **FO**-definability:
 - ▶ All groups are trivial.
 - ▶ All \mathcal{H} -classes are trivial.
- ▶ The algorithm can be modified to reflect them too:

New algorithm:

- Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$
- Operation \bullet : $T_1 \in I[\alpha]$ and $T_2 \in I \Rightarrow T_1 T_2 \in I[\alpha]$
- Operation: \mathcal{G} a subgroup of $I[\alpha] \Rightarrow (\bigcup_{T \in \mathcal{G}} T) \in I[\alpha]$

Alternate algorithms

- ▶ The algorithm reflects the equation $x^\omega = x^{\omega+1}$
- ▶ 2 other equivalent ways to characterize **FO**-definability:
 - ▶ All groups are trivial.
 - ▶ All \mathcal{H} -classes are trivial.
- ▶ The algorithm can be modified to reflect them too:

New algorithm:

- Trivial sets: for all $w \in A^+$ $\{\alpha(w)\} \in I[\alpha]$
 - Operation \bullet : $T_1 \in I[\alpha]$ and $T_2 \in I \Rightarrow T_1 T_2 \in I[\alpha]$
 - Operation: \mathcal{G} a subgroup of $I[\alpha] \Rightarrow (\bigcup_{T \in \mathcal{G}} T) \in I[\alpha]$
- ▶ Works also for \mathcal{H} -classes (similar to Henckell's algorithm).

Alternate algorithms

3 variations of the 3rd operation.

- ▶ Operation ω : $T \in I[\alpha] \Rightarrow (T^\omega \cup T^{\omega+1}) \in I[\alpha]$.
- ▶ Operation H : \mathcal{H} an \mathcal{H} -class of $I[\alpha] \Rightarrow (\bigcup_{T \in \mathcal{H}} T) \in I[\alpha]$.
- ▶ Operation G : \mathcal{G} a subgroup of $I[\alpha] \Rightarrow (\bigcup_{T \in \mathcal{G}} T) \in I[\alpha]$.

Theorem

All variations compute FO-indistinguishable sets, and **all maximal ones**.

Completeness: Generalizing Wilke's proof

Reminder: $I[\alpha] = \bigcap_{k \in \mathbb{N}} I_k[\alpha]$. In particular, for all k , $I[\alpha] \subseteq I_k[\alpha]$.

Completeness: Generalizing Wilke's proof

Reminder: $I[\alpha] = \bigcap_{k \in \mathbb{N}} I_k[\alpha]$. In particular, for all k , $I[\alpha] \subseteq I_k[\alpha]$.

What we prove

For $\ell = |A|2^{|S|^2}$, the algorithm computes **all maximal subsets of $I_\ell[\alpha]$** .
In particular, we get the bound of the "brute-force" approach for free.

Proof technique

To every $w \in A^+$, one can associate $Gen_k(w) \in I_k[\alpha]$:

$$Gen_k(w) = \{s \in S \mid \exists w' \cong_k w \text{ s.t. } \alpha(w') = s\}$$

We prove that for all $w \in A^+$, $Gen_\ell(w)$ is computed by the algorithm.

\Rightarrow We start with a $w \in A^+$, we need a way to decompose it in a way that respects the operations of our algorithm.

Wilke's Proof for characterization of **FO**

We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

$w = \dots\dots\dots$

Wilke's Proof for characterization of **FO**

We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$

$w =$

Wilke's Proof for characterization of FO

We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$

$w =$

In that Case:

$$x^\omega = x^{\omega+1} \Rightarrow S = \{1_S\}$$

Wilke's Proof for characterization of **FO**

We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$

$$w = w_0 a w_1 a w_2 a w_3 a w_4 \dots a w_m$$

Wilke's Proof for characterization of FO

We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does FO proceed to detect $\alpha(w)$?

$\alpha(w_0)$ detectable
(Induction on $|A|$)

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$

$w = \boxed{w_0} aw_1 aw_2 aw_3 aw_4 \dots aw_m$

Wilke's Proof for characterization of FO

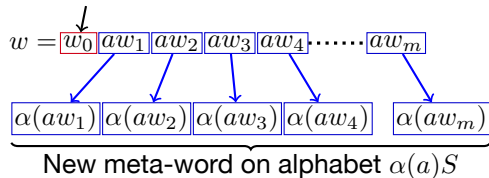
We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

$\alpha(w_0)$ detectable
(Induction on $|A|$)

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$



New morphism $\beta : (\alpha(a)S)^* \rightarrow \alpha(a)S \subsetneq S$
Detectable by induction on $|S|$

Wilke's Proof for characterization of FO

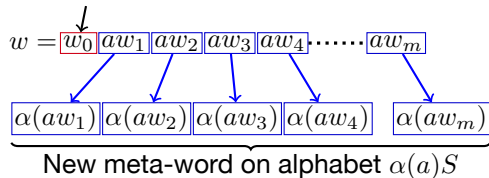
We have $\alpha : A^+ \rightarrow S$ with S satisfying $x^\omega = x^{\omega+1}$.

Let $w \in A^+$, how does **FO** proceeds to detect $\alpha(w)$?

$\alpha(w_0)$ detectable
(Induction on $|A|$)

Two Cases:

1. $\forall a \in A,$
 $\alpha(a)S = S$ and $S\alpha(a) = S$
2. $\exists a \in A,$
 $\alpha(a)S \subsetneq S$ or $S\alpha(a) \subsetneq S$



New morphism $\beta : (\alpha(a)S)^* \rightarrow \alpha(a)S \subsetneq S$
Detectable by induction on $|S|$

- ▶ One can adapt induction in our case.
- ▶ **Aperiodicity used only in the base case.**

Completeness proof: induction

- ▶ $\alpha : A^+ \rightarrow S$ surjective, \mathcal{S} : subsemigroup of 2^S .
- ▶ $\text{Sat}(\mathcal{S})$: add all $T^\omega \cup T^{\omega+1}$, and close by subsemigroup.
- ▶ $\mathcal{S}_0 = \{\{s\} \mid s \in S\}$: singletons.

$$\mathcal{S}_0 \subset \text{Sat}(\mathcal{S}_0) \subset \cdots \subset \text{Sat}^K(\mathcal{S}_0) = \text{Sat}^{K+1}(\mathcal{S}_0) \stackrel{\text{def}}{=} \text{Sat}^*(\mathcal{S}_0)$$

Goal: show that $\downarrow \text{Sat}^*(\mathcal{S}) = I[\alpha]$.

Induction on

- ▶ the number of the elements of the base semigroup:

$$\|\mathcal{S}\| = \bigcup_{X \in \mathcal{S}} X.$$

- ▶ the size $|A|$ of the alphabet.

Completeness proof: induction

Main proposition

- ▶ \mathcal{S} : subsemigroup of 2^S . and $\beta : B^+ \rightarrow \mathcal{S}$ surjective morphism.
- ▶ Then, one can compute FO-formulas $\{\varphi_1, \dots, \varphi_m\}$ such that
 - ▶ The languages $L(\varphi_i)$ form a **partition** of B^+ ,
 - ▶ For all i ,

$$\llbracket \beta(L(\varphi_i)) \rrbracket \in \downarrow \text{Sat}^*(\mathcal{S})$$

Consequences

1. Entails that $I[\alpha] \subseteq \downarrow \text{Sat}^*(\alpha(A^+))$, ie, **completeness**.
2. Separators for languages recognized by α can be chosen as $\bigcup_I L(\varphi_i)$.
3. Constructive: formulas of controlled rank.

Recap for **FO**-separation

We have

1. **Algorithm** computing $I[\alpha] \Rightarrow$ **yes/no** answer for **FO**-separation.
2. **Bound** on the size of the separator.
3. Inductive **computation** of a separator.

More general analysis

- ▶ Goal: handle logics which are not closed under complement.
- ▶ Eg, Σ_2


$$\exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_k \varphi, \quad \text{where } \varphi \text{ quantifier-free}$$

- ▶ Now use a non-symmetrical relation:

$$s \lesssim_k t$$

if

$$\forall \psi \in \Sigma_2 \text{ of rank } k \\ \alpha^{-1}(s) \subseteq L(\psi) \implies \alpha^{-1}(t) \cap L(\psi) \neq \emptyset$$

 \lesssim_k is not transitive.

To compute "chains", one needs more precise information.

- ▶ With **FO**, we worked in a finite universe, 2^S .

Here we work with **infinite** set S^+ .

More general analysis

- ▶ For Σ_1 , one can compute all chains.
- ▶ For Σ_2 , one can compute **all chains fixed length**.
Fixpoint algorithm, by induction on length.

More general analysis

- ▶ For Σ_1 , one can compute all chains.
- ▶ For Σ_2 , one can compute **all chains fixed length**.
Fixpoint algorithm, by induction on length.

Theorem

Let L be a regular language and $\alpha : A^+ \rightarrow S$ be its syntactic morphism. Then, L is definable in Σ_3 iff α satisfies:

$$s^\omega \leq s^\omega t s^\omega \quad \text{for all } (t, s) \in I^{\Sigma_2}[\alpha]$$

- ▶ Actually **generic for all levels**.
- ▶ We have an algorithm for computing $I^{\Sigma_2}[\alpha]$,
 \Rightarrow **decidable characterization of Σ_3** .

Conclusion

We have the following results:

- ▶ Separation by **FO** is decidable (in EXPTIME).
- ▶ Computing an actual separator formula done in an elementary way.
- ▶ Results can be (easily) generalized to infinite words.
- ▶ Can be extended to other classes, e.g., Σ_2 .
- ▶ Transfer results: separation for $\Sigma_2 \implies$ characterization for Σ_3 .

Question: can we compute chains for other levels of the alternation hierarchy?

Thank you!