Separation by First-Order Logic

Thomas Place and Marc Zeitoun

LaBRI, Université Bordeaux, CNRS

April 28, 2014
Objects we consider

Structure

Words

Trees

Descriptive Formalism

First-Order Logic (FO)
Piecewise Testable ($B\Sigma_1$)
2-Variables FO ($FO_2$)
Fragments $\Sigma_i$, $B\Sigma_i$
Locally Testable (LT)
Objects we consider

**Structure**

- Words: `ababcbbaa`
- Trees

**Descriptive Formalism**

Express Properties

- First-Order Logic (FO)
- Piecewise Testable ($B\Sigma_1$)
- 2-Variables FO (FO$_2$)
- Fragments $\Sigma_i$, $B\Sigma_i$
- Locally Testable (LT)
Objects we consider

Structure

Descriptive Formalism

Express Properties

Words

ababcbaa

Trees

First-Order Logic (FO)

Piecewise Testable (BΣ₁)

2-Variables FO (FO₂)

Fragments Σ_i, BΣ_i

Locally Testable (LT)

For this talk
Decide the following problem:

Take a regular language \( L \)
Decide the following problem:

Take a regular language $L$

Can it be defined with a \textbf{FO} formula?

Schützenberger'65, McNaughton and Papert'71

For a regular language, the following are equivalent:
$L$ is FO-definable.
The syntactic monoid of $L$ satisfies $u\omega + 1 = u\omega$. 
A Reduced Problem: Decidable Characterizations

Decide the following problem:

Take a regular language \( L \)

Can it be defined with a \textbf{FO} formula?

Schützenberger’65, McNaughton and Papert’71

For \( L \) a regular language, the following are equivalent:

- \( L \) is \textbf{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 $u^\omega$ and $u^{\omega+1}$.
- **Yet**: the logic can still express facts on $L$. 
Decide the following problem:

Take two regular languages $L_1, L_2$

Can $L_1$ be separated from $L_2$ with a $\text{FO}$-definable formula?
Decide the following problem:

Take two regular languages $L_1, L_2$

Can $L_1$ be separated from $L_2$ with a FO formula?
Separation

Decide the following problem:

Take two regular languages $L_1, L_2$

$\begin{array}{c}
L_1 \\
\text{a} \\
\text{a} \\
\text{a} \\
\text{b} \\
\text{b} \\
\end{array}$

$\begin{array}{c}
L_2 \\
\text{b} \\
\text{a} \\
\text{b} \\
\text{a} \\
\text{b} \\
\text{b} \\
\text{a} \\
\end{array}$

Can $L_1$ be separated from $L_2$ with a FO formula?
Separation

Decide the following problem:

Take two regular languages $L_1, L_2$

Can $L_1$ be separated from $L_2$ with a FO formula?

More general than getting a decidable characterization.
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.
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 $\text{Pl}$-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 $H^\omega$* defines $C^\omega(S)$, a collection of pointlike sets, in a constructive manner. $H^\omega$ 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 $S \overset{R}{\rightarrow} \text{CP}(S)$ computing $C^\omega(S)$ with $\text{CP}(S)$ aperiodic. It uses $H^\omega$, generalized to $\hat{H}^\omega$ on $\hat{C}^\omega(S)$ ‘to get rid of groups by blowing up’.
Alternate formulations

Several formulations of separation [Almeida’96]

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 $w_1 \equiv_k w_2$.
3. For all aperiodic $T$ and morphism $\beta : A^* \rightarrow T$,
   \[ \beta(L_1) \cap \beta(L_2) \neq \emptyset. \]
4. $\overline{L}_1 \cap \overline{L}_2 \neq \emptyset$ (closures taken in the pro-aperiodic semigroup).
5. $\overline{L}_1 \cap \overline{L}_2$ contains an $\omega$-term.

Actually, 5 may be exploited to prove decidability of separation.
Quantifier rank of a formula: Nested depth of quantifiers.

$$\forall x \exists y \ (a(x) \implies \exists z \ (x < z < y \land b(y)))$$ 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.

\[ \forall x \exists y \ (a(x) \implies \exists z \ (x < z < y \land b(y))) \] has quantifier rank 3

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

\( k \)-equivalence for FO

Let \( w_1, w_2 \) be words:

\[ w_1 \cong_k w_2 \text{ iff } w_1, w_2 \text{ satisfy the same formulas of rank } k \]

All FO properties of rank \( k \) are unions of classes of \( \cong_k \).
Fixed Quantifier Rank $k$

Let's add the $\equiv_k$-classes

Let's add the $\equiv_k$-classes

Idea. Abstract $\equiv_k$ on a monoid recognizing both $L_1$ and $L_2$. 

When $k$ gets larger, $\equiv_k$ is refined but it never ends.

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

$\Rightarrow$ Compute a 'limit' for $\equiv_k$.
Let's add the $\sim_k$-classes separable with rank $k$ iff no $\sim_k$-class intersects both languages.

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

$\Rightarrow$ Compute a 'limit' for $\sim_k$. 
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 $k$

$\Rightarrow$ Compute a 'limit' for $\cong_k$.

When $k$ gets larger, $\cong_k$ is refined but it never ends
Separable with rank $k$ iff no $\equiv_k$-class intersects both languages

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

$\Rightarrow$ Compute a 'limit' for $\equiv_k$.

When $k$ gets larger, $\equiv_k$ 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 $\text{FO}$ 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
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

Idea. Abstract $\cong_k$ on a monoid recognizing both $L_1$ and $L_2$. 
“Pair” analysis

Recall from Thomas’ talk:

**FO-indistinguishable pairs for** \( \alpha : A^* \rightarrow M \)

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

\[
\exists \quad w_1 \underset{k}{\sim} w_2 \quad \alpha \quad \alpha
\]

\[
\begin{array}{c}
\alpha \\
\downarrow \\
s_1
\end{array} 
\begin{array}{c}
\alpha \\
\downarrow \\
s_2
\end{array}
\]

- 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:

\[
(s_1, s_2) \in I[\alpha] \iff \alpha^{-1}(s_1) \text{ and } \alpha^{-1}(s_2) \text{ not separable}
\]
$L_1, L_2$ recognized by $\alpha : A^* \rightarrow M$ are not separable iff there are accepting elements $s_1, s_2 \in M$ for $L_1, L_2$ s.t. $(s_1, s_2) \in I[\alpha]$. 

Separation Criterion
The Separation Criterion

Separation Criterion

$L_1, L_2$ recognized by $\alpha : A^* \to M$ are not separable
iff
there are accepting elements $s_1, s_2 \in M$ for $L_1, L_2$ s.t. $(s_1, s_2) \in I[\alpha]$. 

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

2 approaches to compute the relation $I[\alpha]$:

**Brute-force:**

- Computing $I_k[\alpha]$ easy for fixed $k$.
- $I[\alpha] = I_k[\alpha]$ for some $k$ depending on $\alpha$.
- $\Rightarrow$ Prove a bound $k = f(\alpha)$ and compute $I_k[\alpha]$.

**Algorithm:**

Find an algorithm that bypasses the bound $k$ and computes $I[\alpha]$ directly.
Two approaches

2 approaches to compute the relation $I[\alpha]$:

**Brute-force:**

- Computing $I_k[\alpha]$ easy for fixed $k$.
- $I[\alpha] = I_k[\alpha]$ for some $k$ depending on $\alpha$.
- $\Rightarrow$ Prove a bound $k = f(\alpha)$ and compute $I_k[\alpha]$.

**Algorithm:**

Find an algorithm that bypasses the bound $k$ and computes $I[\alpha]$ directly.

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 $\text{FO}$

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

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

1st Property of $\text{FO}$
$$w \simeq_k w$$

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 $\text{FO}$

$w_1 \simeq_k w_2$ and $u_1 \simeq_k u_2 \implies w_1 u_1 \simeq_k w_2 u_2$

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 $\text{FO}$

$w_1 \sim_k w_2$ and $u_1 \sim_k u_2 \Rightarrow w_1 u_1 \sim_k w_2 u_2$

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 $(t_1, t_2) \in I[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in I[\alpha]$
Idea. Start with trivial pairs and add more pairs via a fixpoint algorithm.

3rd Property of FO
\[ \forall k \exists n \forall w_1, w_2 \in A^* \quad w_1 \equiv_k w_2 \Rightarrow (w_1)^n \equiv_k (w_2)^{n+1} \]

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 $\text{FO}$

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

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 $\omega$: $(s_1, s_2) \in I[\alpha] \Rightarrow (s_1^{\omega}, s_2^{\omega+1}) \in I[\alpha]$
A first (non complete) Algorithm computing $I[\alpha]$

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

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 $\omega$: $(s_1, s_2) \in I[\alpha] \Rightarrow (s_1^\omega, s_2^{\omega+1}) \in I[\alpha]$

Correct by definition but not complete
Why it does not work

3rd Property of FO

\[ w_1 \equiv_k w_2 \implies (w_1)^n \equiv_k (w_2)^{n+1} \]
Why it does not work

Not general enough

3rd Property of FO

\[ w_1 \preceq_k w_2 \Rightarrow (w_1)^n \preceq_k (w_2)^{n+1} \]

Needs to be replaced

\[ w_1 \preceq_k w_2 \preceq_k \cdots \preceq_k w_m \]

\[ \Downarrow \]

All large concatenations of words in \( \{w_1, \ldots, w_m\} \) are \( \preceq_k \)-equivalent.
Need for better analysis

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

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

\[
\exists w_1 \equiv_k w_2 \equiv_k \cdots \equiv_k w_n
\]

\[
\begin{array}{cccc}
\alpha & \alpha & \cdots & \alpha \\
\downarrow & \downarrow & \cdots & \downarrow \\
s_1 & s_2 & \cdots & s_n
\end{array}
\]

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

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

$$S \in I[\alpha] \text{ iff } S \in I_k[\alpha], \quad \forall k \in \mathbb{N}$$
From Pairs to Sets

New Objective

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

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

Remark

- With our new definition, we have $I[\alpha] \subseteq 2^M$.
- $2^M$ is a monoid for the operation $S_1 \cdot S_2 = \{s_1 s_2 \mid s_1 \in S_1, s_2 \in S_2\}$. 
A new (working) Algorithm

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

1st Property of \( \text{FO} \)
\[
\omega \equiv_k \omega
\]
A new (working) Algorithm

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

1st Property of FO

\[ w \equiv_k w \]

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 \( \text{FO} \)

\[
{w_1 \equiv_k w_2 \text{ and } u_1 \equiv_k u_2} \implies {w_1 u_1 \equiv_k w_2 u_2}
\]

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

\[ w_1 \sim_k w_2 \text{ and } u_1 \sim_k u_2 \implies w_1 u_1 \sim_k w_2 u_2 \]

1. Trivial sets: for all \( w \in A^* \) \( \{ \alpha(w) \} \in I[\alpha] \)
2. Operation \( \bullet \): \( S_1 \in I[\alpha] \text{ and } S_2 \in I[\alpha] \implies S_1 S_2 \in I[\alpha] \)
A new (working) Algorithm

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

3rd Property of $\mathbf{FO}$

$$w_1 \simeq_k w_2 \cdots \simeq_k w_m$$

⇒

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

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

2. Operation $\bullet$: $S_1 \in \mathbb{I}[\alpha]$ and $S_2 \in \mathbb{I}[\alpha]$ $\Rightarrow$ $S_1 S_2 \in \mathbb{I}[\alpha]$
A new (working) Algorithm

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

3rd Property of FO

\[ w_1 \equiv_k w_2 \cdots \equiv_k w_m \]

\[ \Downarrow \]

All large concatenations of words in \{w_1, \cdots, w_m\} are \( \equiv_k \)-equivalent.

1. Trivial sets: for all \( w \in A^* \) \( \{\alpha(w)\} \in I[\alpha] \)
2. Operation \( \bullet \): \( S_1 \in I[\alpha] \) and \( S_2 \in I[\alpha] \) \( \Rightarrow \) \( S_1 S_2 \in I[\alpha] \)
3. Operation \( \omega \): \( S \in I[\alpha] \) \( \Rightarrow \) \( (S^\omega \cup S^{\omega+1}) \in I[\alpha] \)
A new (working) Algorithm

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

<table>
<thead>
<tr>
<th>3rd Property of FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_1 \equiv_k w_2 \cdots \equiv_k w_m )</td>
</tr>
<tr>
<td>↓</td>
</tr>
<tr>
<td>All large concatenations of words in ( {w_1, \cdots, w_m} ) are ( \equiv_k )-equivalent.</td>
</tr>
</tbody>
</table>

1. Trivial sets: For all \( w \in A^* \) \( \{\alpha(w)\} \in I[\alpha] \)
2. Operation \( \bullet \): \( S_1 \in I[\alpha] \) and \( S_2 \in I[\alpha] \) \( \Rightarrow S_1 S_2 \in I[\alpha] \)
3. Operation \( \omega \): \( S \in I[\alpha] \) \( \Rightarrow (S^\omega \cup S^{\omega+1}) \in I[\alpha] \)

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 $\bullet$: $S_1 \in I[\alpha]$ and $S_2 \in I \Rightarrow S_1 S_2 \in I[\alpha]$
  - Operation: $\mathcal{G}$ a subgroup of $I[\alpha] \Rightarrow (\bigcup_{S \in \mathcal{G}} S) \in I[\alpha]$
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 $H$-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 $\bullet$: $S_1 \in I[\alpha]$ and $S_2 \in I \Rightarrow S_1 S_2 \in I[\alpha]$
- Operation: $G$ a subgroup of $I[\alpha] \Rightarrow (\bigcup_{S \in G} S) \in I[\alpha]$

Works also for $H$-classes (similar to Henckell’s algorithm).
Alternate algorithms

3 variations of the 3rd operation.

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

Theorem

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

3 variations of the 3rd operation.

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

Theorem

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

- From $G$ to $\omega$: if $\mathcal{G} = \{T_1, \ldots, T_n\}$, then

$$T_1 \cup \cdots \cup T_n \subseteq (T_1^\omega \cup T_1^{\omega+1}) \cdots (T_n^\omega \cup T_n^{\omega+1})$$
Alternate algorithms

3 variations of the 3rd operation.

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

Theorem

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

- From $G$ to $\omega$: if $\mathcal{G} = \{ T_1, \ldots, T_n \}$, then
  
  $$T_1 \cup \cdots \cup T_n \subseteq (T_1^\omega \cup T_1^{\omega+1}) \cdots (T_n^\omega \cup T_n^{\omega+1})$$

- From $\omega$ to $H$: $S^\omega$ and $S^{\omega+1}$ are $\mathcal{H}$-equivalent.
Alternate algorithms

3 variations of the 3rd operation.

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

**Theorem**

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

- From $G$ to $\omega$: if $G = \{T_1, \ldots, T_n\}$, then
  
  \[ T_1 \cup \cdots \cup T_n \subseteq (T_1^\omega \cup T_1^{\omega+1}) \cdots (T_n^\omega \cup T_n^{\omega+1}) \]

- From $\omega$ to $H$: $S^\omega$ and $S^{\omega+1}$ are $\mathcal{H}$-equivalent.
- From $H$ to $G$: every $\mathcal{H}$-class is of the form $R.G$ with $G$ a group.
Completeness

1 How to prove completeness?

2 Does this work for all logics?
Completeness

1. How to prove completeness?
   ⇒ Generalizing well-known characterization proof by Wilke

2. Does this work for all logics?
Completeness

1. How to prove completeness?
   ⇒ Generalizing well-known characterization proof by Wilke

2. Does this work for all logics?
   ⇒ No, this works only for FO (deeply linked to the proof)
Completeness: What we need to prove

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

Completeness: What we need to prove

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 $\ell = |M|(2^{|M|})!$, 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 $Gen_k(w) \in I_k[\alpha]$:

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

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 Proof of the FO characterization

We have \( \alpha : A^* \rightarrow M \) with \( M \) satisfying \( x^\omega = x^{\omega+1} \).
Let \( w \in A^* \), how does FO proceeds to detect \( \alpha(w) \)?

\[
\begin{align*}
\text{Two Cases:} \\
\bullet & \text{For all } a \in A, \quad \alpha(a)_M = M \text{ and } M_{\alpha(a)} = M \\
\bullet & \text{There exists } a \in A \text{ such that } \alpha(a)_M \subsetneq M \text{ or } M_{\alpha(a)} \subsetneq M
\end{align*}
\]

In that Case:
\[
\begin{align*}
\omega = x^\omega = x^{\omega+1} \Rightarrow M = \{1\}
\end{align*}
\]
Wilke Proof of the FO characterization

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

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

Two Cases:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$
- There exists $a \in A$ such that $\alpha(a)M \subset M$ or $M\alpha(a) \subset M$

\[ w = \ldots \]
Wilke Proof of the **FO** characterization

We have $\alpha : A^* \rightarrow M$ with $M$ satisfying $x^\omega = x^{\omega+1}$. Let $w \in A^*$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$
- There exists $a \in A$ such that $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

**In that Case:**

$x^{\omega} = x^{\omega+1} \Rightarrow M = \{1_M\}$

**w = ........................................**
Wilke Proof of the **FO** characterization

We have $\alpha : A^* \rightarrow M$ with $M$ satisfying $x^\omega = x^{\omega+1}$. Let $w \in A^*$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$

- There exists $a \in A$ such that $\alpha(a)M \subset M$ or $M\alpha(a) \subset M$

$$w = w_0 aw_1 aw_2 aw_3 aw_4 \ldots \ldots \ldots \ldots aw_m$$
Wilke Proof of the FO characterization

We have $\alpha : A^* \rightarrow M$ with $M$ 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:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$
- There exists $a \in A$ such that $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

$w = w_0 aw_1 aw_2 aw_3 aw_4 \ldots \ldots \ldots \ldots aw_m$
Wilke Proof of the **FO** characterization

We have $\alpha : A^* \rightarrow M$ with $M$ satisfying $x^\omega = x^{\omega+1}$. Let $w \in A^*$, how does **FO** proceeds to detect $\alpha(w)$?

Two Cases:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$
- There exists $a \in A$ such that $\alpha(a)M \subset M$ or $M\alpha(a) \subset M$

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

$w = w_0 aw_1 aw_2 aw_3 aw_4 \ldots \ldots \ldots \ldots aw_m$

$\alpha(aw_1) \alpha(aw_2) \alpha(aw_3) \alpha(aw_4) \ldots \ldots \ldots \ldots \alpha(aw_m)$

New meta-word on alphabet $\alpha(a)M$

Use a morphism $\beta : (\alpha(a)M)^* \rightarrow \alpha(a)M \subset M$

Detectable by induction on $|M|$
Wilke Proof of the FO characterization

We have $\alpha : A^* \rightarrow M$ with $M$ 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:

- For all $a \in A$, $\alpha(a)M = M$ and $M\alpha(a) = M$
- There exists $a \in A$ such that $\alpha(a)M \subset M$ or $M\alpha(a) \subset M$

New meta-word on alphabet $\alpha(a)M$

Use a morphism $\beta : (\alpha(a)M)^* \rightarrow \alpha(a)M \subset M$
Detectable by induction on $|M|$

Adapt induction: the algorithm works for smaller alphabets and smaller semigroups.

Aperiodicity used only in the base case.
We have

1. An algorithm for computing $I[\alpha]$. Therefore, we can answer yes/no to the separation problem for $\text{FO}$.

2. A bound on the size of the separator: it is possible to compute a separator (in a very non-efficient way).
We have the following results:

- Separation by $\text{FO}$ is decidable (in $\text{EXPTIME}$).
- Computing an actual separator formula can be done in an elementary way (but still with high complexity).
- Results can be (easily) generalized to infinite words.
Thank you!
Assume $I[\alpha] = \{S_1, \ldots, S_n\}$. By reversing the completeness proof, it is possible to compute $n$ FO formulas $\varphi_1, \ldots, \varphi_n$ of rank $k = |M|(2^{|M|})!$ such that:

- The associated languages are covering: $\{w \mid w \models \varphi_1 \lor \cdots \lor \models \varphi_n\} = A^*$.
- For all $i$, $w \models \varphi_i \Rightarrow \alpha(w_i) \in S_i$.
- The computation is inductive and elementary.

$\Rightarrow$ All information that can be expressed with FO as stated in $I[\alpha]$ is a union of these formulas.