Separation and First-Order Logic

Thomas Place
Joint work with Marc Zeitoun

LaBRI Université Bordeaux 1

November 27, 2013
Separation

What is it?
Why should you try it?
Objects we consider

Structure

Words

ababcbaa

Trees

Descriptive Formalism

First-Order Logic (\(\mathbf{FO}\))

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

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

Locally Testable (\(\mathbf{LT}\))
Objects we consider

Structure
- Words: ababcbaa
- Trees

Express Properties

Descriptive Formalism
- First-Order Logic (FO)
- 2-Variables FO (FO<sub>2</sub>)
- Piecewise Testable (BSigma<sub>1</sub>)
- Locally Testable (LT)
Objects we consider

Structure

- Words: ababcbaa
- Trees

Descriptive Formalism

- First-Order Logic (\(\text{FO}\))
- 2-Variables \(\text{FO} (\text{FO}_2)\)
- Piecewise Testable (\(\mathcal{B}\Sigma_1\))
- Locally Testable (\(\text{LT}\))

Express Properties

For the this talk (at least the first part)
Decide the following problem:

Take a regular language $L$
A Reduced Problem: Decidable Characterizations

Decide the following problem:

Take a regular language $L$

Can it be defined with a FO formula?
Why do we study these things?

Decidable characterization of a formalism \( \mathcal{F} \)

\[ = \]

Understanding Expressive Power of \( \mathcal{F} \)
Why do we study these things?

Decidable characterization of a formalism $\mathcal{F}$

$=$

Understanding Expressive Power of $\mathcal{F}$

Ideal Result Form:

$L$ definable in $\mathcal{F}$

$L$ verifies set of properties

(A Schützenberger'65, McNaughton and Papert'71): $L$ a regular language, the following are equivalent:

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

A counterexample always happen when not definable.
Why do we study these things?

Decidable characterization of a formalism $\mathcal{F}$

$= \quad$ Understanding Expressive Power of $\mathcal{F}$

Proof = Construction of a Formula,
Hypothesis = Simple Properties

⇒ Gives a standard way to write all formulas

Ideal Result Form:

$L$ definable in $\mathcal{F}$

$L$ verifies set of properties

(Schützenberger’65, McNaughton and Papert’71):

$L$ a regular language, the following are equivalent:

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

A counterexample always happen when not definable.
Why do we study these things?

Decidable characterization of a formalism $\mathcal{F}$

(Schützenberger’65, McNaughton and Papert’71):

$L$ a regular language, the following are equivalent:

- $L$ is $\text{FO}$ definable.
- The syntactic monoid of $L$ satisfies $u^{\omega+1} = u^{\omega}$.
Why do we study these things?

Decidable characterization of a formalism $F$

(Schützenberger’65, McNaughton and Papert’71):

$L$ a regular language, the following are equivalent:

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

A counterexample that always happen when not definable.
Why would we want more?

If the characterization answer is yes:
- All we did in the characterization’s proof apply.
- All subparts of the automata are actually definable definable in $\mathbf{FO}$ (provided it is minimal).

If the characterization’s answer is no:
- We know very little.
- All we know: defining $L$ requires differentiating between $u^\omega$ and $u^{\omega+1}$ at some point.

Maybe interesting things can be said even in that case.
Here comes Separation

Decide the following problem:

Take two regular languages $L_1, L_2$

$L_1$ and $L_2$ are given in the diagram.
Here comes Separation

Decide the following problem:

Take two regular languages $L_1$, $L_2$

Can $L_1$ be separated from $L_2$ with a $\text{FO}$ formula?
Here comes Separation

Decide the following problem:

Take two regular languages $L_1, L_2$

Can $L_1$ be separated from $L_2$ with a \textbf{FO} formula?
A more General Problem than Characterization

Possible to build a single automata for both languages

Problem is now: What can FO say about this automata
A more General Problem than Characterization

Possible to build a single automata for both languages

Problem is now:
What can FO say about this automata
Advantages of Separation

- We need **FO** techniques that apply to all languages, not just the **FO** definable ⇒ More general and can be used in any context.
Advantages of Separation

- We need **FO** techniques that apply to all languages, not just the **FO** definable $\Rightarrow$ More general and can be used in any context.

- In the **FO** characterization, $x^\omega = x^{\omega+1}$ is a specific counterexample that always happen. Here we need a way to describe all of them.
Advantages of Separation

- We need **FO** techniques that apply to all languages, not just the **FO** definable $\Rightarrow$ More general and can be used in any context.

- In the **FO** characterization, $x^\omega = x^{\omega+1}$ is a specific counterexample that always happen. Here we need a way to describe all of them.

- Since techniques work for all languages, maybe they can then be reused to solve harder problems.
Two Parts

- Part I: Separation with FO.
- Part II: Solving old open problems with Separation.
First-Order Logic and Separation
Goal: Separate red language from blue language with FO.

Question: What computable information is sufficient to solve that problem?
Quantifier rank of a formula: Nested depth of quantifiers.

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

If \( k \) fixed: finitely many FO properties of rank \( k \) ⇒ Separation is easy (test them all)
Quantifier rank of a formula: Nested depth of quantifiers.

\( \forall x \exists y (a(x) \land \exists z (x < z < y \land b(y))) \) 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:

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

*All FO properties of rank \( k \) are unions of classes of \( \cong_k \).*
Let's add the $\cong_k$-classes

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

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

Compute a 'limit' for $\cong_k$.

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

We have an Automata $A$.

Idea: Abstract $\cong_k$, on the states of the automata.

$\exists w_1, w_2 \in A^* \text{s.t. } w_1 \cong_k w_2$ and $q_i w_1 \rightarrow q_1 q_i w_2 \rightarrow q_2$.

Separation condition for rank $k$ now on the final states.

Finitely many pairs means there is a limit.

$\forall k (q_1 \cong_k q_2) \Rightarrow$ We want to compute $\cong_k$.

Warning very bad notation (not transitive).

(We will denote it as a set of pairs).

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 a $k$

$\Rightarrow$ Compute a 'limit' for $\cong_k$. 
Fixed Quantifier Rank $k$

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

For full $\mathbf{FO}$ we want to know if there exists 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 $\mathbf{FO}$ we want to know if there exists 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 a $k$

⇒ Compute a 'limit' for $\cong_k$.

When $k$ gets larger, $\cong_k$ is refined but it never ends
Let's add the $\preceq_k$-classes separable with rank $k$ iff no $\preceq_k$-class intersects both languages.

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

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

When $k$ gets larger, $\preceq_k$ is refined but it never ends.
Fixed Quantifier Rank $k$

We have an Automata $A$
$\Rightarrow$ Idea: Abstract $\equiv_k$, on the states of the automata

- $q_1 \equiv_k q_2$ iff $\exists w_1, w_2 \in A^*$ s.t. $w_1 \equiv_k w_2$ and $q_i \xrightarrow{w_1} q_1$
- $q_i \xrightarrow{w_2} q_2$

- Separation condition for rank $k$ now on the final states

- Finitely many pairs means there is a limit.

- $q_1 \equiv q_2$ iff $\forall k\ q_1 \equiv_k q_2$, $\Rightarrow$ We want to compute $\equiv$

When $k$ gets larger, $\equiv_k$ is refined but it never ends
Fixed Quantifier Rank \( k \)

We have an Automata \( A \)
\[ \Rightarrow \text{Idea: Abstract } \cong_k, \text{ on the states of the automata} \]

- \( q_1 \cong_k q_2 \) iff \( \exists w_1, w_2 \in A^\ast \) s.t. \( w_1 \cong_k w_2 \) and \( q_i \xrightarrow{w_1} q_1 \)
  \( q_i \xrightarrow{w_2} q_2 \)

- Separation condition for rank \( k \) now on the final states

  Warning very bad notation (not transitive)
  \[ \Rightarrow (\text{We will denote it as a set of pairs}) \]

- Finitely many pairs means there is a limit.

- \( q_1 \cong q_2 \) iff \( \forall k \ q_1 \cong_k q_2 \), \( \Rightarrow \) We want to compute \( \cong \)

When \( k \) gets larger, \( \cong_k \) is refined but it never ends
We have an Automata $\mathcal{A}$
$\Rightarrow$ Idea: Abstract $\cong_k$, on the states of the automata

- $(q_1, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$ and $q_i \xrightarrow{w_1} q_1$ $q_i \xrightarrow{w_2} q_2$

- Separation condition for rank $k$ now on the final states

- Finitely many pairs means there is a limit.

- $(q_1, q_2) \in P(\mathcal{A})$ iff $\forall k (q_1, q_2) \in P_k(\mathcal{A})$, $\Rightarrow$ We want to compute $P(\mathcal{A})$

When $k$ gets larger, $\cong_k$ is refined but it never ends
**Separation Criterion**

$L_1, L_2$ recognized by $\mathcal{A}$ are **not** iff there final states $q_1, q_2$ for $L_1, L_2$ s.t. $(q_1, q_2) \in P(\mathcal{A})$. 
The Separation Criterion

Separation Criterion

$L_1, L_2$ recognized by $\mathcal{A}$ are not iff there final states $q_1, q_2$ for $L_1, L_2$ s.t. $(q_1, q_2) \in P(\mathcal{A})$.

Computing $P(\mathcal{A})$ suffices to solve separation by FO. However it gives much more.
Back to our example

$P(\mathcal{A}) = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (3, 6), (6, 3), (2, 7), (7, 2)\}$
Back to our example

\[ P(\mathcal{A}) = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (3, 6), (6, 3), (2, 7), (7, 2)\} \]

⇒ This is the information we wanted

"Everything" FO can express about \( \mathcal{A} \)
Back to our example

\[ P(A) = \{(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7), (3, 6), (6, 3), (2, 7), (7, 2)\} \]

⇒ This is the information we wanted

"Everything" FO cannot express about \( A \)
Levels of Precision, what do we mean by everything?

Final States

- Automata $\mathcal{A}$.

- $(q_1, q_2) \in P_k(\mathcal{A})$ iff
  $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$
  and $q_i \xrightarrow{w_1} q_1$
  and $q_i \xrightarrow{w_2} q_2$

- $(q_1, q_2) \in P(\mathcal{A})$ iff
  $\forall k \ (q_1, q_2) \in P_k(\mathcal{A})$
Levels of Precision, what do we mean by everything?

<table>
<thead>
<tr>
<th>Final States</th>
<th>Intial and Final States</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Automata $\mathcal{A}$.</td>
<td>• Automata $\mathcal{A}$.</td>
</tr>
<tr>
<td>• $(q_1, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^* \text{ s.t. } w_1 \cong_k w_2$ and $q_i \xrightarrow{w_1} q_1$ \quad $q_i \xrightarrow{w_2} q_2$</td>
<td>• $(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^* \text{ s.t. } w_1 \cong_k w_2$ and $r_1 \xrightarrow{w_1} q_1$ \quad $r_2 \xrightarrow{w_2} q_2$</td>
</tr>
<tr>
<td>• $(q_1, q_2) \in P(\mathcal{A})$ iff $\forall k ,(q_1, q_2) \in P_k(\mathcal{A})$</td>
<td>• $(r_1, q_1, r_2, q_2) \in P(\mathcal{A})$ iff $\forall k ,(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$</td>
</tr>
</tbody>
</table>
Levels of Precision, what do we mean by everything?

Final States

- Automata $\mathcal{A}$.

- $(q_1, q_2) \in P_k(\mathcal{A})$ iff
  $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$
  and $q_i \xrightarrow{w_1} q_1$
  $q_i \xrightarrow{w_2} q_2$

- $(q_1, q_2) \in P(\mathcal{A})$ iff
  $\forall k \ (q_1, q_2) \in P_k(\mathcal{A})$

Initial and Final States

- Automata $\mathcal{A}$.

- $(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ iff
  $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$
  and $r_1 \xrightarrow{w_1} q_1$
  $r_2 \xrightarrow{w_2} q_2$

- $(r_1, q_1, r_2, q_2) \in P(\mathcal{A})$ iff
  $\forall k \ (r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$

Monoids

- Morphism $\alpha : A^* \rightarrow M$.

- $(s_1, s_2) \in P_k(M)$ iff
  $\exists w_1, w_2$ s.t. $w_1 \cong_k w_2$ and
  $\alpha(w_1) = s_1$
  $\alpha(w_2) = s_2$

- $(s_1, s_2) \in P(M)$ iff
  $\forall k \ (s_1, s_2) \in P_k(M)$
Levels of Precision, what do we mean by everything?

For FO, we choose monoids

Remark: For all monoids there exists a number \( \omega \) such that \( \forall s \in M, s^\omega = s^\omega s^\omega \)

Monoids

- Morphism \( \alpha : A^* \rightarrow M \).
- \((s_1, s_2) \in P_k(M)\) iff \( \exists w_1, w_2 \) s.t. \( w_1 \equiv_k w_2 \) and \( \alpha(w_1) = s_1 \), \( \alpha(w_2) = s_2 \)
- \((s_1, s_2) \in P(M)\) iff \( \forall k \ (s_1, s_2) \in P_k(M) \)
Two approaches

We want to compute the relation $P_k$, there are two approaches:

**Brute-force:**

- When $k$ fixed computing $P_k(M)$ is easy.
- $P(M) = P_k(M)$ for some fixed $k$ depending on $M$.
- $\Rightarrow$ Prove a bound $k = f(M)$ and compute $P_k$.

**Algorithm:**

Find an algorithm that bypasses the bound $k$ and computes $P$ directly.
Two approaches

We want to compute the relation $P_k$, there are two approaches:

**Brute-force:**
- When $k$ fixed computing $P_k(M)$ is easy.
- $P(M) = P_k(M)$ for some fixed $k$ depending on $M$.
- $\implies$ Prove a bound $k = f(M)$ and compute $P_k$.

**Algorithm:**
- Find an algorithm that bypasses the bound $k$ and computes $P$ directly.

We choose approach 2.
A first (non complete) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $P(M)$. Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO

$\forall k \forall w \in A^* \; w \sim_k w$
A first (non complete) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $P(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO

\[
\forall k \forall w \in A^* \ w \cong_k w
\]

- Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in P(M)$
A first (non complete) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $P(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

<table>
<thead>
<tr>
<th>Property of FO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\forall k \forall w_1, w_2, u_1, u_2 \in A^* \ w_1 \equiv_k w_2 \text{ and } u_1 \equiv_k u_2 \Rightarrow w_1 u_1 \equiv_k w_2 u_2$</td>
</tr>
</tbody>
</table>

- Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in P(M)$
A first (non complete) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $P(M)$. Idea : Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO:

$$\forall k \forall w_1, w_2, u_1, u_2 \in A^* \; w_1 \equiv_k w_2 \text{ and } u_1 \equiv_k u_2 \Rightarrow w_1 u_1 \equiv_k w_2 u_2$$

- Trivial pairs: for all $w \in A^*$, $(\alpha(w), \alpha(w)) \in P(M)$
- Operation: $(s_1, s_2) \in P(M)$ and $(t_1, t_2) \in P(M) \Rightarrow (s_1 t_1, s_2 t_2) \in P(M)$
We have $\alpha : A^* \rightarrow M$ and want to compute $P(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

**Property of FO**

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

- Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in P(M)$
- Operation: $(s_1, s_2) \in P(M)$ and $(t_1, t_2) \in P(M) \Rightarrow (s_1 t_1, s_2 t_2) \in P(M)$
A first (non complete) Algorithm

We have \( \alpha : A^* \rightarrow M \) and want to compute \( P(M) \). Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO

\[
\forall k \exists n \forall w_1, w_2 \in A^* \ w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}
\]

- Trivial pairs: for all \( w \in A^* \ (\alpha(w), \alpha(w)) \in P(M) \)
- Operation: \( (s_1, s_2) \in P(M) \) and \( (t_1, t_2) \in P(M) \) \( \Rightarrow (s_1 t_1, s_2 t_2) \in P(M) \)
- Operation: \( (s_1, s_2) \in P(M) \) \( \Rightarrow ((s_1)^{\omega}, (s_2)^{\omega+1}) \in P(M) \) and \( ((s_1)^{\omega+1}, (s_2)^{\omega}) \in P(M) \)
A first (non complete) Algorithm

We have \( \alpha : A^* \rightarrow M \) and want to compute \( P(M) \). Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

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

- Trivial pairs: for all \( w \in A^* \) \( (\alpha(w), \alpha(w)) \in P(M) \)
- Operation: \((s_1, s_2) \in P(M) \) and \((t_1, t_2) \in P(M) \) \( \Rightarrow (s_1 t_1, s_2 t_2) \in P(M) \)
- Operation: \((s_1, s_2) \in P(M) \) \( \Rightarrow ((s_1)^\omega, (s_2)^{\omega+1}) \in P(M) \) and \(( (s_1)^{\omega+1}, (s_2)^\omega ) \in P(M) \)

Correct by definition but not complete
We now explain why
Why it does not work

Property of FO
\[ \forall k \ \exists n \ \forall w_1, w_2 \in A^* \ w_1 \sim_k w_2 \Rightarrow (w_1)^n \sim_k (w_2)^{n+1} \]
Why it does not work

Not general enough

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

Needs to be replaced

\[ \forall k \forall w_1, \ldots, w_m \in A^*, \; w_1 \sim_k w_2 \cdots \sim_k w_m \]
\[ \Downarrow \]
All large concatenations of words in \( \{w_1, \ldots, w_m\} \) are \( \sim_k \)-equivalent.
From Pairs to Sets

For using this more general property, pairs are not general enough ⇒ Use sets.

For all \( k \), we set \( S_k(M) \subseteq 2^M \) the set s.t.:

\[
\{s_1, s_2, s_3\} \in S_k \text{ iff } \exists w_1, w_2, w_3 \in A^* \text{ s.t. } w_1 \cong_k w_2 \cong_k w_3 \text{ and } \alpha(w_1) = s_1, \alpha(w_2) = s_2, \alpha(w_3) = s_3
\]

New Objective

We want to compute the set \( S(M) \subseteq 2^M \) such that:

\[
S \in S(M) \text{ iff } \forall k \in \mathbb{N} \ S \in S_k(M)
\]
From Pairs to Sets

For using this more general property, pairs are not general enough ⇒ Use sets.

For all $k$, we set $S_k(M) \subseteq 2^M$ the set s.t.:

$$\{s_1, s_2, s_3\} \in S_k \text{ iff } \exists w_1, w_2, w_3 \in A^* \text{ s.t. } w_1 \equiv_k w_2 \equiv_k w_3 \text{ and } \alpha(w_1) = s_1, \alpha(w_2) = s_2, \alpha(w_3) = s_3$$

New Objective

We want to compute the set $S(M) \subseteq 2^M$ such that:

$$S \in S(M) \text{ iff } \forall k \in \mathbb{N} \ S \in S_k(M)$$

Remark

$2^M$ is still a monoid for the operation $S_1 \cdot S_2 = \{s_1s_2 \mid s_1 \in S_1, s_2 \in S_2\}$. 
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO
\[ \forall k \forall w \in A^* \ w \approx_k w \]
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO
\[ \forall k \forall w \in A^* \ w \sim_k w \]

- Trivial sets: for all $w \in A^*$ \( \{\alpha(w)\} \in S(M) \)
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO
$$\forall k \forall w_1, w_2, u_1, u_2 \in A^* \ w_1 \equiv_k w_2 \text{ and } u_1 \equiv_k u_2 \Rightarrow w_1 u_1 \equiv_k w_2 u_2$$

- Trivial sets: for all $w \in A^*$ \{\alpha(w)\} $\in S(M)$
A new (working) Algorithm

We have $\alpha : A^* \to M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

**Property of FO**

$\forall k \forall w_1, w_2, u_1, u_2 \in A^* \ w_1 \equiv_k w_2 \text{ and } u_1 \equiv_k u_2 \Rightarrow w_1 u_1 \equiv_k w_2 u_2$

- Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in S(M)$
- Operation: $S_1 \in S(M)$ and $S_2 \in S(M) \Rightarrow S_1 S_2 \in S(M)$
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

\[
\forall k \forall w_1, \ldots, w_m \in A^*, w_1 \equiv_k w_2 \cdots \equiv_k w_m \\
\Rightarrow \text{All large concatenations of words in } \{w_1, \cdots, w_m\} \text{ are } \equiv_k \text{-equivalent.}
\]

- Trivial sets: for all $w \in A^*$ \{\(\alpha(w)\}\} $\in S(M)$
- Operation: $S_1 \in S(M)$ and $S_2 \in S(M) \Rightarrow S_1 S_2 \in S(M)$
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

Property of FO

$\forall k \forall w_1, \ldots, w_m \in A^*, w_1 \cong_k w_2 \cdots \cong_k w_m$

$\Downarrow$

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

- Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in S(M)$
- Operation: $S_1 \in S(M)$ and $S_2 \in S(M) \Rightarrow S_1 S_2 \in S(M)$
- Operation: $S \in S(M) \Rightarrow (S^\omega \cup S^{\omega+1}) \in S(M)$
A new (working) Algorithm

We have $\alpha : A^* \rightarrow M$ and want to compute $S(M)$. Idea: Start with trivial pairs and add more pairs via a fixpoint algorithm.

**Property of FO**
\[ \forall k \forall w_1, \ldots, w_m \in A^*, w_1 \equiv_k w_2 \cdots \equiv_k w_m \]
\[ \Downarrow \]
All large concatenations of words in $\{w_1, \ldots, w_m\}$ are $\equiv_k$-equivalent.

- Trivial sets: for all $w \in A^*$ \{\(\alpha(w)\)\} $\in S(M)$
- Operation: $S_1 \in S(M)$ and $S_2 \in S(M) \Rightarrow S_1S_2 \in S(M)$
- Operation: $S \in S(M) \Rightarrow (S^\omega \cup S^\omega+1) \in S(M)$

Correct by definition
Can be proved to be complete
An alternate algorithm

The algorithm reflects the equation $x^\omega = x^{\omega+1}$ in the well-known $\text{FO}$ characterization. There are other ways to state this characterization, can the algorithm be modified to reflect them too?
An alternate algorithm

The algorithm reflects the equation $x^\omega = x^\omega + 1$ in the well-known FO characterization. There are other ways to state this characterization, can the algorithm be modified to reflect them too?

New algorithm:

- Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in S(M)$
- Operation: $S_1 \in S(M)$ and $S_2 \in S \Rightarrow S_1 S_2 \in S(M)$
- Operation: $G$ a subgroup of $S(M) \Rightarrow (\bigcup_{S \in G} S) \in S(M)$
An alternate algorithm

The algorithm reflects the equation $x^\omega = x^{\omega + 1}$ in the well-known $\text{FO}$ characterization. There are other ways to state this characterization, can the algorithm be modified to reflect them too?

**New algorithm:**

- **Trivial sets:** for all $w \in A^*$ \( \{ \alpha(w) \} \in S(M) \)
- **Operation:** $S_1 \in S(M)$ and $S_2 \in S$ $\Rightarrow S_1 S_2 \in S(M)$
- **Operation:** $G$ a subgroup of $S(M)$ $\Rightarrow (\bigcup_{S \in G} S) \in S(M)$

This also works
Questions that remain

1. How do you prove completeness?

2. Does this work for all logics?

3. All of this is very nice but what if I want to actually compute an FO separator (and finish within my lifespan)?
Questions that remain

1. How do you prove completeness?
   ⇒ By generalizing a well-known proof of the FO characterization by Wilke

2. Does this work for all logics?

3. All of this is very nice but what if I want to actually compute an FO separator (and finish within my lifespan)?
Questions that remain

1. How do you prove completeness?
   ⇒ By generalizing a well-known proof of the FO characterization by Wilke

2. Does this work for all logics?
   ⇒ No, this works only for FO (deeply linked to the proof)

3. All of this is very nice but what if I want to actually compute an FO separator (and finish within my lifespan)?
Questions that remain

1. How do you prove completeness?
   ⇒ By generalizing a well-known proof of the $\text{FO}$ characterization by Wilke

2. Does this work for all logics?
   ⇒ No, this works only for $\text{FO}$ (deeply linked to the proof)

3. All of this is very nice but what if I want to actually compute an $\text{FO}$ separator (and finish within my lifespan)?
   ⇒ You can, but you will still have to be patient...
Completeness
Completeness: What we need to prove

Reminder

$S(M) = \bigcap_{k \in \mathbb{N}} S_k(M)$. In particular, for all $k$, $S(M) \subseteq S_k(M)$.
Completeness: What we need to prove

Reminder

\[ S(M) = \bigcap_{k \in \mathbb{N}} S_k(M). \]
In particular, for all \( k \), \( S(M) \subseteq S_k(M) \).

What we prove

For \( k = |M|(2^{|M|})! \), the algorithm computes \( S_k(M) \). \( \Rightarrow \) In particular, we get the bound of the brute-force approach for free.

How do we proceed.

To every \( w \in A^* \), one can associate \( Gen_k(w) \in S_k \):

\[ Gen_k(w) = \{ s \in M \mid \exists w' \cong_k w \text{ s.t. } \alpha(w') = s \} \]

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

\[ w = \ldots \]
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 $M\alpha(a) \subsetneq M$
Wilke Proof of the FO characterization

We have $\alpha : A^* \to M$ 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 $M\alpha(a) \subset M$

In that Case:
$x^\omega = x^{\omega+1} \Rightarrow M = \{1_M\}$
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:

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

Two Cases:

\[ w = w_0 \, a w_1 \, a w_2 \, a w_3 \, a w_4 \ldots \ldots \ldots \ldots a w_m \]
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 $\alpha(w)$?

\[ \alpha(w_0) \text{ detectable} \]
(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 $M\alpha(a) \subsetneq M$

\[ w = [w_0] aw_1 aw_2 aw_3 aw_4 \cdots \cdots \cdots aw_m \]
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 $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

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

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

New meta-word on alphabet $\alpha(a)M$
Use a morphism $\beta : (\alpha(a)M)^* \rightarrow \alpha(a)M \subsetneq M$
Detectable by induction on $|M|$
Wilke Proof of the FO characterization

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

For separation this proof has both a big advantage and a big disadvantage

Disadvantage:

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

For separation this proof has both a big advantage and a big disadvantage.

Disadvantage: Induction use $|M|$ as a parameter.

Advantage: Aperiodicity only used in the base case: not needed for induction.
Wilke Proof of the $\text{FO}$ characterization

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

For separation this proof has both a big advantage and a big disadvantage.

Disadvantage: Induction use $|M|$ as a parameter.

Advantage: Aperiodicity only used in the base case: not needed for induction.

For all $a \in A$ and $M \in \mathcal{M}$,

- $\alpha(a)M = M$
- There exists $a \in A$ such that $\alpha(a)M \subsetneq M$
Let’s summarize what we have

1. An algorithm for computing $S(M)$. Therefore, we can answer yes or no to the separation problem for $FO$.

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

Question: Can Item 2 be done in a better way?
Answer is Yes: Computing a Separator

Assume $S(M) = \{S_1, \ldots, S_n\}$. By reversing the completeness proof, it is possible to compute $n \textbf{FO}$ 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 $S(M)$ is a union of these formulas.
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.
Separation and the Quantifier Alternation Hierarchy
Most well-known classes are solved:

- Piecewise Testable Languages ($B\Sigma_1(\prec)$). [Czerwinski, Martens, and Masopust] and independently [P., Rooijen and Zeitoun] (2013).
Most well-known classes are solved:

- Piecewise Testable Languages ($B\Sigma_1(<)$). [Czerwinski, Martens, and Masopust] and independently [P., Rooijen and Zeitoun] (2013).

One class was left for which characterization was known while nothing was known on separation: $\Sigma_2(<)$. 
FO Quantifier Alternation Hierarchy

\[ \Sigma_1(\prec) \subseteq \Sigma_2(\prec) \subseteq \Sigma_3(\prec) \subseteq \ldots \]

\[ \Pi_1(\prec) \subseteq \Pi_2(\prec) \subseteq \Pi_3(\prec) \subseteq \ldots \]

\[ B\Sigma_1(\prec) \subseteq \Delta_2(\prec) \subseteq B\Sigma_2(\prec) \subseteq \Delta_3(\prec) \subseteq B\Sigma_3(\prec) \subseteq \ldots \]

Boolean Combinations

Separation Knowledge

Characterization Knowledge

\[ \Sigma_2(\prec) \text{ and } \Pi_2(\prec) \text{ were open for separation (Pin, Weil)'97} \]

New Separation Knowledge

New Characterization Knowledge

Using the separation solution for \( \Sigma_2(\prec) \) and \( \Pi_2(\prec) \), one can prove decidable characterizations for \( B\Sigma_2(\prec) \), \( \Delta_3(\prec) \), \( \Sigma_3(\prec) \) and \( \Pi_3(\prec) \).
**FO Quantifier Alternation Hierarchy**

$\exists \Sigma_1(\prec)$

$\forall \Pi_1(\prec)$

$\exists \mathcal{B} \Sigma_1(\prec)$

$\forall \Delta_2(\prec)$

$\exists \Sigma_2(\prec)$

$\forall \Pi_2(\prec)$

$\exists \mathcal{B} \Sigma_2(\prec)$

$\forall \Delta_3(\prec)$

$\exists \Sigma_3(\prec)$

$\forall \Pi_3(\prec)$

$\exists \mathcal{B} \Sigma_3(\prec)$

$\forall \Delta_3(\prec) \cdots$
FO Quantifier Alternation Hierarchy

\[\Sigma_1(\prec) \subseteq B\Sigma_1(\prec) \subseteq \Delta_2(\prec) \subseteq \Sigma_2(\prec) \cap \Pi_2(\prec) \subseteq B\Sigma_2(\prec) \subseteq \Delta_3(\prec) \subseteq \Sigma_3(\prec) \cap \Pi_3(\prec) \subseteq B\Sigma_3(\prec) \subseteq \cdots\]

\[\exists \Sigma_2(\prec) \cap \Pi_2(\prec) \subseteq \Sigma_3(\prec) \cap \Pi_3(\prec) \subseteq \exists \exists \Sigma_3(\prec) \cap \Pi_3(\prec) \subseteq \exists \exists \exists \Sigma_3(\prec) \cap \Pi_3(\prec) \subseteq \cdots\]
FO Quantifier Alternation Hierarchy

Boolean Combinations

Σ₂(⟨⟩) ∩ Π₂(⟨⟩) → Σ₂(⟨⟩) → Π₂(⟨⟩) → Σ₃(⟨⟩) ∩ Π₃(⟨⟩) → Σ₃(⟨⟩) → Π₃(⟨⟩) → Σ₄(⟨⟩) → ...
FO Quantifier Alternation Hierarchy

Separation Knowledge

\[ \Sigma_1(\prec) \to B\Sigma_1(\prec) \to \Delta_2(\prec) \]

\[ \Pi_1(\prec) \]

\[ \Sigma_2(\prec) \]

\[ \Pi_2(\prec) \]

\[ \Sigma_3(\prec) \]

\[ \Pi_3(\prec) \]

\[ B\Sigma_2(\prec) \to \Delta_3(\prec) \]

\[ B\Sigma_3(\prec) \ldots \]

Boolean Combinations

Separation Knowledge

Characterization Knowledge

\[ \Sigma_2(\prec) \] and \[ \Pi_2(\prec) \] were open for separation (Pin, Weil)'97

New Separation Knowledge

New Characterization Knowledge

Using the separation solution for \[ \Sigma_2(\prec) \] and \[ \Pi_2(\prec) \],

one can prove decidable characterizations for

\[ B\Sigma_2(\prec) \]

\[ \Delta_3(\prec) \]

\[ B\Sigma_3(\prec) \ldots \]
Separation Knowledge

Characterization Knowledge

$\Sigma_2(\prec)$ and $\Pi_2(\prec)$ were open for separation

(Pin, Weil)'97

$\Sigma_2(\prec)$ and $\Pi_2(\prec)$ were open for separation

New Separation Knowledge

New Characterization Knowledge

Using the separation solution for $\Sigma_2(\prec)$ and $\Pi_2(\prec)$, one can prove decidable characterizations for $B\Sigma_2(\prec)$, $\Delta_3(\prec)$, $\Sigma_3(\prec)$ and $\Pi_3(\prec)$. 

$\Sigma_3(\prec)$ and $\Pi_3(\prec)$ were open for separation
FO Quantifier Alternation Hierarchy

New Separation Knowledge
Using the separation solution for $\Sigma_2(\prec)$ and $\Pi_2(\prec)$, one can prove decidable characterizations for $B\Sigma_2(\prec), \Delta_3(\prec), \Sigma_3(\prec)$ and $\Pi_3(\prec)$. 

New Separation Knowledge

New Characterization Knowledge
Three Main Results

1. Separation by $\Sigma_2(\prec)$.
2. Characterization for $\Sigma_3(\prec)$.
3. Characterization for $B\Sigma_2(\prec)$. 
Three Main Results

1. Separation by $\Sigma_2(<)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(<)$.
3. Characterization for $B\Sigma_2(<)$. 
Three Main Results

1. Separation by $\Sigma_2(\prec)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(\prec)$. $\Rightarrow$ Not that Difficult.
3. Characterization for $B\Sigma_2(\prec)$. 
Three Main Results

1. Separation by $\Sigma_2(\prec)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(\prec)$. $\Rightarrow$ Not that Difficult.
3. Characterization for $B\Sigma_2(\prec)$. $\Rightarrow$ Very Difficult.
Separation and $\Sigma_2(<)$, what we compute

A $\Sigma_2(<)$ formula is this:

$$\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m \varphi(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

Quantifier rank still applies, here rank is $n + m$. $\Sigma_2(<)$ not closed under complement $\Rightarrow$ no $k$-equivalence, a $k$-preorder.

**$k$-preorder for $\Sigma_2(<)$**

Let $w_1, w_2$ be words:

$$w_1 \lesssim_k^{\Sigma_2} w_2 \text{ iff for any } \Sigma_2(<) \text{ formula } \Psi \text{ rank } k, (w_1 \models \Psi \Rightarrow w_2 \models \Psi)$$

$L$ is $\Sigma_2(<)$ definable with rank $k$ iff $L = \{w \mid \exists w' \in L \text{ s.t. } w' \lesssim_k^{\Sigma_2} w\}$
Separation and $\Sigma_2(\prec)$, what we compute (2)

We have a monoid morphism $\alpha : A^* \rightarrow M$, for all $k \in \mathbb{N}$ set

$$C_k^2(M) = \{(s_1, s_2) | \exists w_1, w_2 \in A^* \text{ s.t. } w_1 \prec_k^2 w_2 \text{ and } \alpha(w_1) = s_1, \alpha(w_2) = s_2\}$$

What we want to compute (and can compute) is

$$C^2(M) = \{(s_1, s_2) | \forall k (s_1, s_2) \in C_k^2(M)\}$$

This solves separation:

**Separation Solution**

$L_1, L_2$ recognized by $M$. $L_1$ is not separable from $L_2$ iff there exists $(s_1, s_2) \in C^2(M)$ such that $s_1 \in \alpha^{-1}(L_1)$ and $s_2 \in \alpha^{-1}(L_2)$. 
Separation and $\Sigma_2(\prec)$, what we compute (2)

We have a monoid morphism $\alpha : A^* \to M$, for all $k \in \mathbb{N}$ set

$$C^2_k(M) = \{(s_1, s_2) \mid \exists w_1, w_2 \in A^* \text{ s.t. } w_1 \triangleleft_k w_2 \text{ and } \alpha(w_1) = s_1, \alpha(w_2) = s_2\}$$

What we want to compute (and can compute) is

$$C^2(M) = \{(s_1, s_2) \mid \forall k \ (s_1, s_2) \in C^2_k(M)\}$$

This solves separation:

**Separation Solution**

$L_1, L_2$ recognized by $M$. $L_1$ is not separable from $L_2$ iff there exists $(s_1, s_2) \in C^2(M)$ such that $s_1 \in \alpha^{-1}(L_1)$ and $s_2 \in \alpha^{-1}(L_2)$.

However, computing $C^2(M)$ gives much more.
A Decidable Characterization for $\Sigma_3(<)$. 
Deriving a characterization for $\Sigma_3(<)$.

**Theorem**

Let $L$ be a regular language and $\alpha : A^* \rightarrow M$ be its syntactic morphism. $L$ is definable in $\Sigma_3(<)$ iff $\alpha$ satisfies:

$$s^\omega \leq s^\omega ts^\omega \text{ for } (t, s) \in C^2$$
Deriving a characterization for $\Sigma_3(\prec)$.

**Theorem**

Let $L$ be a regular language and $\alpha : A^* \rightarrow M$ be its syntactic morphism. $L$ is definable in $\Sigma_3(\prec)$ iff $\alpha$ satisfies:

$$s^\omega \leq s^\omega ts^\omega \text{ for } (t, s) \in C^2$$

What does $\leq$ mean? The syntactic monoid comes fromes

Myhill-Nerode Equivalence:

$$w \equiv w' \text{ iff } \forall u, v \ uvw \in L \iff u\!w'\!v \in L$$

The preorder $\leq$ comes from a Myhill-Nerode preorder:

$$w \leq w' \text{ iff } \forall u, v \ uvw \in L \Rightarrow u\!w'\!v \in L$$
What of $\mathcal{B}\Sigma_2(\prec)$?
A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(<)$

We defined $C(M)$ as pairs, we can be more general. For all $k \in \mathbb{N}$ we $C^2_k(M) \subseteq M^*$ such that:

$$(s_1, \ldots, s_n) \in C^2_k(M) \text{ iff } \exists w_1, \ldots, w_n \in A^* \text{ s.t. } w_1 \lessapprox_k^2 \cdots \lessapprox_k^2 w_n \text{ and }$$

$$\alpha(w_1) = s_1 \quad \vdots \quad \alpha(w_n) = s_n$$

We set

$$C^2(M) = \bigcap_{k \in \mathbb{N}} C_k(M)$$

**Fun Fact**

$C^2(M)$ is a regular language over the alphabet $M$. 
A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(<)$

We defined $\mathcal{C}(M)$ as pairs, we can be more general. For all $k \in \mathbb{N}$ we $C_k^2(M) \subseteq M^*$ such that:

$$(s_1, \ldots, s_n) \in C_k^2(M) \text{ iff } \exists w_1, \ldots, w_n \in A^* \text{ s.t. } w_1 \preceq_k^2 \cdots \preceq_k^2 w_n \text{ and } \alpha(w_1) = s_1 \cdots \alpha(w_n) = s_n$$

We set

$$C^2(M) = \cap_{k \in \mathbb{N}} C_k(M)$$

Fun Fact

$C^2(M)$ is a regular language over the alphabet $M$.

Separation Theorem

$L_1, L_2$ recognized by $M$ are not $\mathcal{B}\Sigma_2(<)$-separable iff there exists $s_1 \in \alpha^{-1}(L_1)$ and $s_2 \in \alpha^{-1}(L_2)$ such that $(s_1 s_2)^* \subseteq C^2(M)$. 
A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(\prec)$

Deciding separation for $\mathcal{B}\Sigma_2(\prec)$ requires computing $C^2(M)$.

- We do not know how to do it.
Deciding separation for $\mathcal{B}\Sigma_2(\prec)$ requires computing $\mathcal{C}^2(M)$.

- We do not know how to do it.
- The best we can do: for any fixed $n \in \mathbb{N}$: compute the words of length $n$ in $\mathcal{C}^2(M)$. 
Deciding separation for $\mathcal{B}\Sigma_2(\prec)$ requires computing $C^2(M)$.

- We do not know how to do it.
- The best we can do: for any fixed $n \in \mathbb{N}$: compute the words of length $n$ in $C^2(M)$.
- What if we only want a decidable characterization?
Deriving a characterization for $\mathcal{B}\Sigma_2(\prec)$. (half of it)

**Theorem**

Let $L$ be a regular language and $\alpha : A^* \rightarrow M$ be its syntactic morphism. $L$ is definable in $\mathcal{B}\Sigma_2(\prec)$ iff $\alpha$ satisfies the two following equations:

$$
\begin{align*}
  s_1^\omega s_3^\omega &= s_1^\omega s_2 s_3^\omega \\
  s_3^\omega s_1^\omega &= s_3^\omega s_2 s_1^\omega
\end{align*}
$$

for $(s_1, s_2, s_3) \in C^2(M)$.

**Decidability**

It suffices to compute words of length 2 and 3 in $C^2(M)$ which we can do. $\Rightarrow$ the characterization is decidable.

**Fun Fact**

The first equation generalizes the $\mathcal{B}\Sigma_1(\prec)$ characterization in a nice way.
Theorem

Let \( L \) be a regular language and \( \alpha : A^* \to M \) be its syntactic morphism. \( L \) is definable in \( \mathcal{B}\Sigma_2(\prec) \) iff \( \alpha \) satisfies the two following equations:

\[
\begin{align*}
    s_1^\omega s_3^\omega &= s_1^\omega s_2 s_3^\omega \\
    s_3^\omega s_1^\omega &= s_3^\omega s_2 s_1^\omega
\end{align*}
\]

for \((s_1, s_2, s_3) \in C^2(M)\)

Equation 2 (requires a slight generalization of words of length 2 in \( C^2(M) \))
Theorem

Let $L$ be a regular language and $\alpha : A^* \rightarrow M$ be its syntactic morphism. $L$ is definable in $\mathcal{B}\Sigma_2(\prec)$ iff $\alpha$ satisfies the two following equations:

\[
\begin{align*}
{s_1}s_3 \omega &= s_1 s_2 s_3 \omega & \text{for } (s_1, s_2, s_3) \in C^2(M) \\
{s_3} s_1 \omega &= s_3 s_2 s_1 \omega 
\end{align*}
\]

Equation 2 (requires a slight generalization of words of length 2 in $C^2(M)$)

Decidability

It suffices to compute words of length 2 and 3 in $C^2(M)$ which we can do. $\Rightarrow$ the characterization is decidable.
Deriving a characterization for $\mathcal{B}\Sigma_2(<)$. (half of it)

**Theorem**

Let $L$ be a regular language and $\alpha : A^* \rightarrow M$ be its syntactic morphism. $L$ is definable in $\mathcal{B}\Sigma_2(<)$ iff $\alpha$ satisfies the two following equations:

\[
\begin{align*}
    s_1^\omega s_3^\omega & = s_1^\omega s_2 s_3^\omega \\
    s_3^\omega s_1^\omega & = s_3^\omega s_2 s_1^\omega
\end{align*}
\]

for $(s_1, s_2, s_3) \in C^2(M)$

*Equation 2 (requires a slight generalization of words of length 2 in $C^2(M)$)*

**Decidability**

It suffices to compute words of length 2 and 3 in $C^2(M)$ which we can do. $\Rightarrow$ the characterization is decidable.

**Fun Fact**

The first equation generalizes the $\mathcal{B}\Sigma_1(<)$ characterization in a nice way.
Conclusion

We have the following results:

- Separation for FO(\(<\)).
- Separation for \(\Sigma_2(\,<\,).\)
- Decidable Characterization for \(B\Sigma_2(\,<\,).\)
- Decidbale Characterizations for \(\Sigma_3(\,<\,), \Pi_3(\,<\,)\) and \(\Delta_3(\,<\,).\)
Conclusion

We have the following results:

- Separation for $\text{FO}(\prec)$.
- Separation for $\Sigma_2(\prec)$.
- Decidable Characterization for $\mathcal{B}\Sigma_2(\prec)$.
- Decidbale Characterizations for $\Sigma_3(\prec)$, $\Pi_3(\prec)$ and $\Delta_3(\prec)$.

What is left:

- Separation for $\mathcal{B}\Sigma_2(\prec)$.
- Going Higher in the Hierarchy: Separation for $\Sigma_3(\prec)$.
- Trees.
Thank You