# Separation and First-Order Logic

Thomas Place
Joint work with Marc Zeitoun

LaBRI Université Bordeaux 1

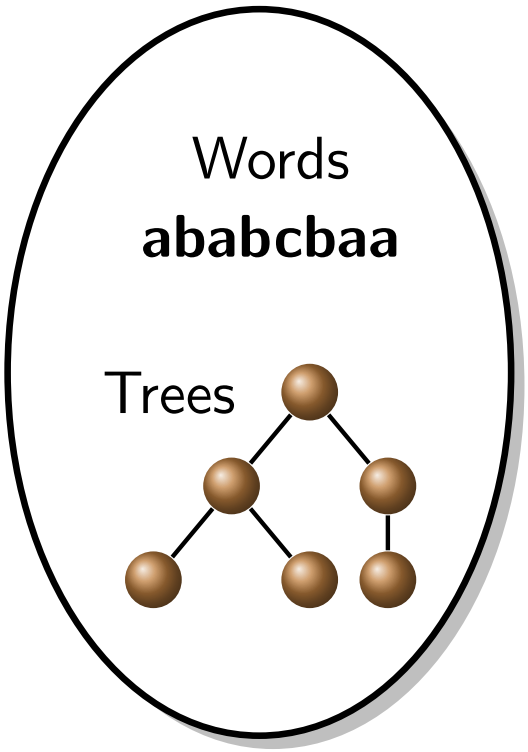
November 27, 2013

# Separation

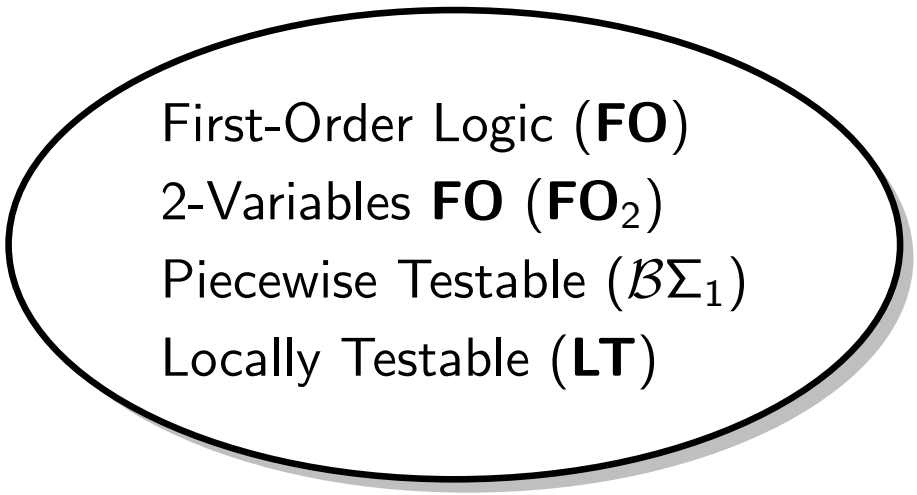
What is it ?
Why should you try it ?

# Objects we consider

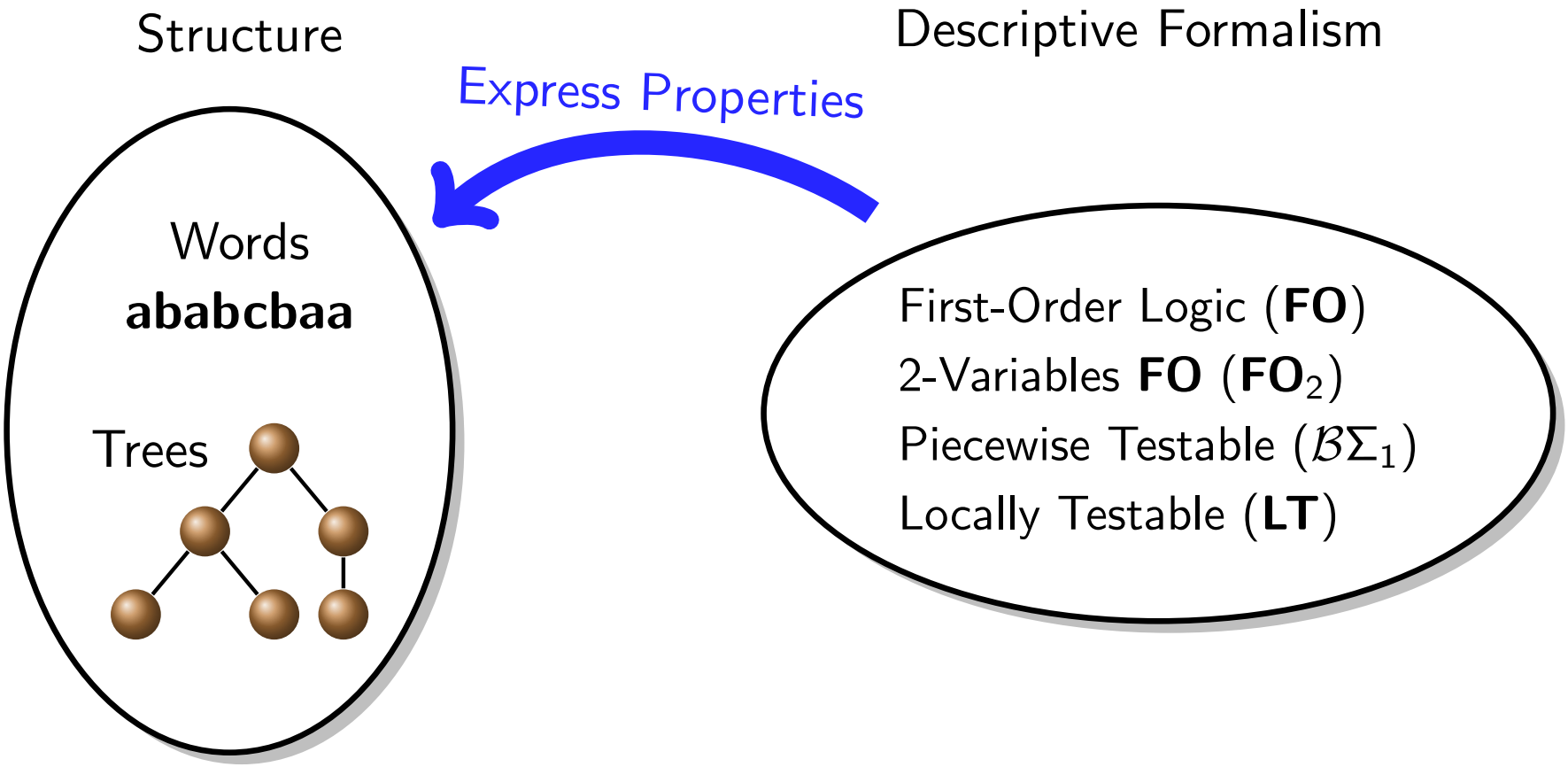
Structure




Descriptive Formalism



# Objects we consider



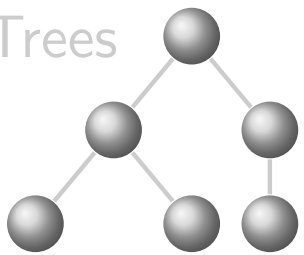
# Objects we consider

Structure

Words

**ababcbaa**

Trees



Express Properties

Descriptive Formalism

First-Order Logic (**FO**)

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

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

Locally Testable (**LT**)

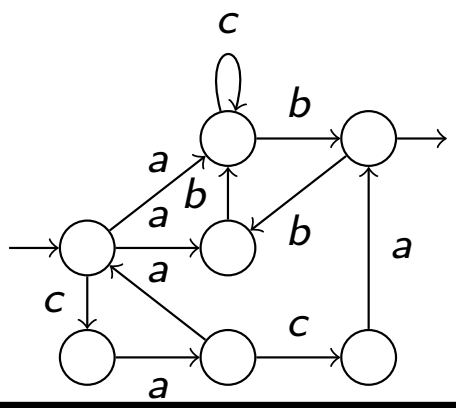
For the this talk

(at least the first part)

# A Reduced Problem: Decidable Characterizations

Decide the following problem:

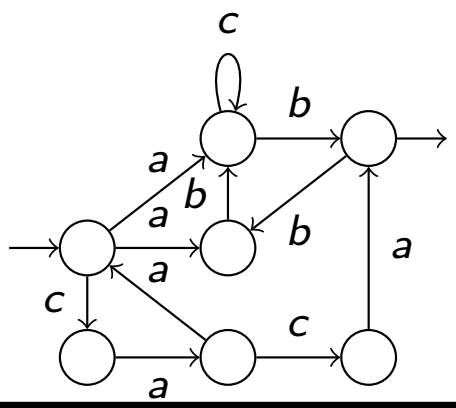
Take a regular language $L$



# A Reduced Problem: Decidable Characterizations

Decide the following problem:

Take a regular language $L$




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

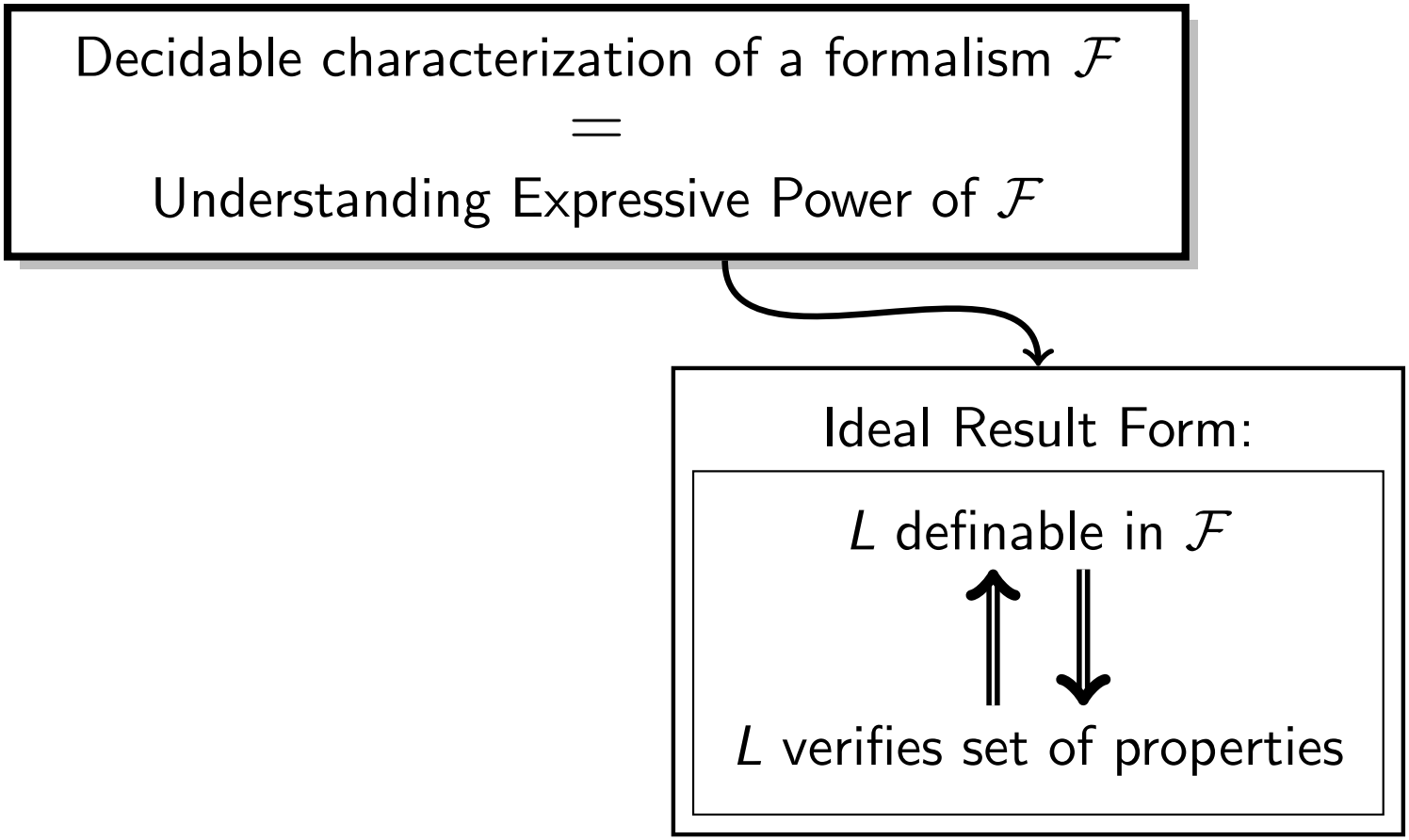
# Why do we study these things ?

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

$=$

Understanding Expressive Power of $\mathcal{F}$

# Why do we study these things ?



# Why do we study these things ?

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

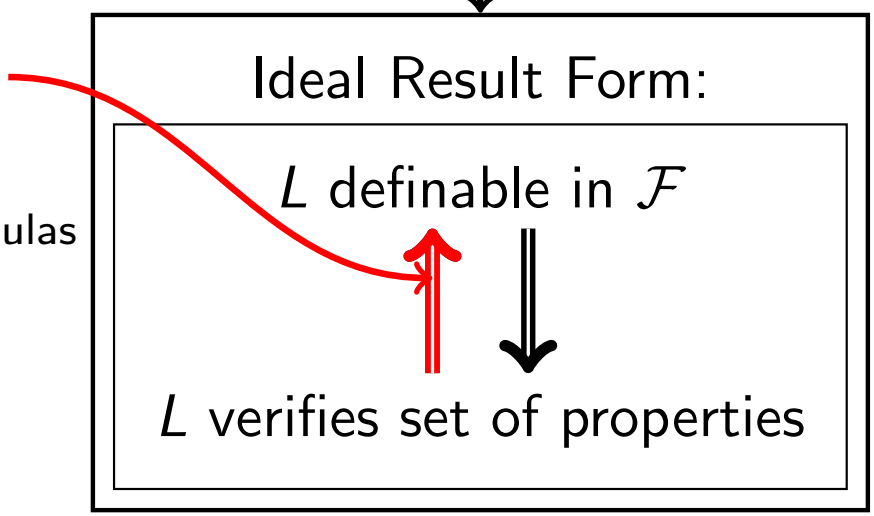
$=$

Understanding Expressive Power of $\mathcal{F}$

Proof = Construction of a Formula,

Hypothesis = Simple Properties

$\Rightarrow$ Gives a standard way to write all formulas



# 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 **FO** definable.

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

$\Rightarrow$

es

# 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:

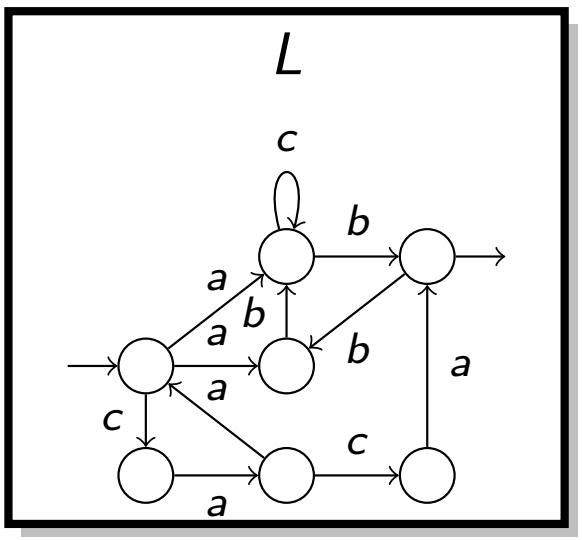
- $L$ is **FO** definable.
- The syntactic monoid of $L$ satisfies $u^{\omega+1} = u^{\omega}$.

$\Rightarrow$

A counterexample that always happen when not definable.

es

# 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 **FO** (provided it is minimal).

If the characterization's answer is no:

-We know very little.

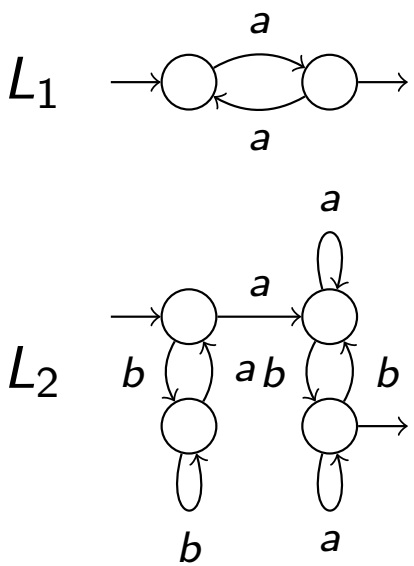
-All we know: defining $L$ requires differentiating between $u^\omega$ and $u^{\omega+1}$ at some point.

Maybe interesting things can be said even in that case.

# Here comes Separation

Decide the following problem:

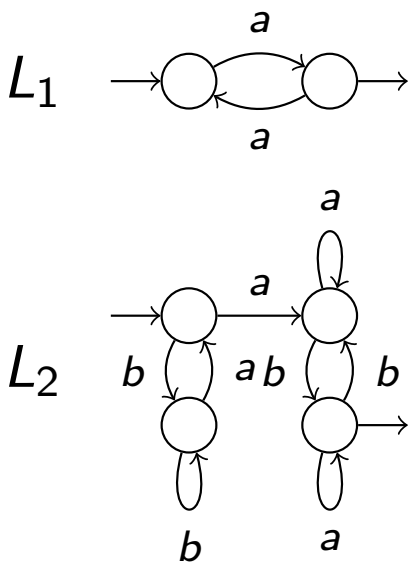
Take two regular languages $L_1, L_2$



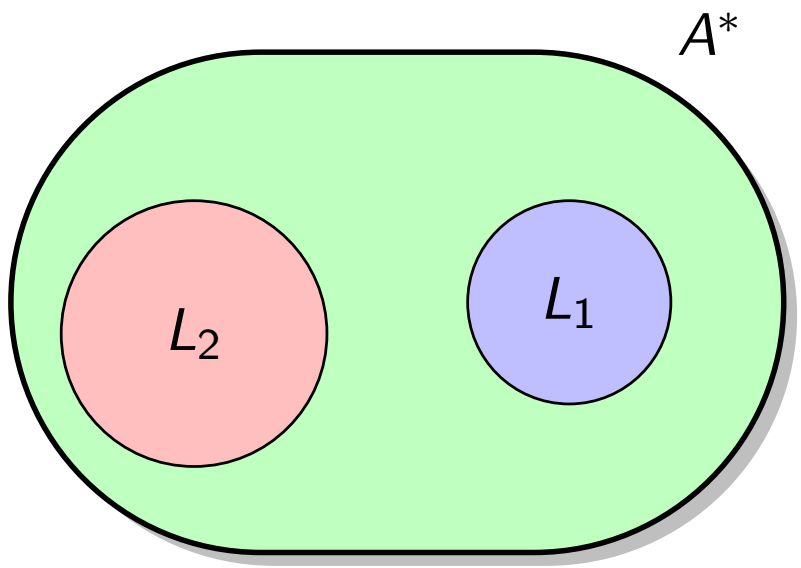
# Here comes Separation

Decide the following problem:

Take two regular languages $L_1, L_2$



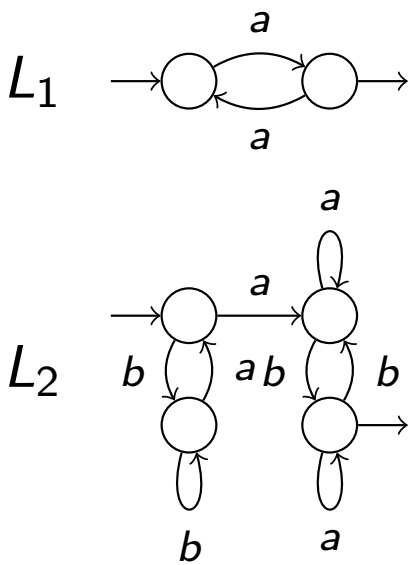

Can $L_1$ be separated from $L_2$ with a **FO** formula ?



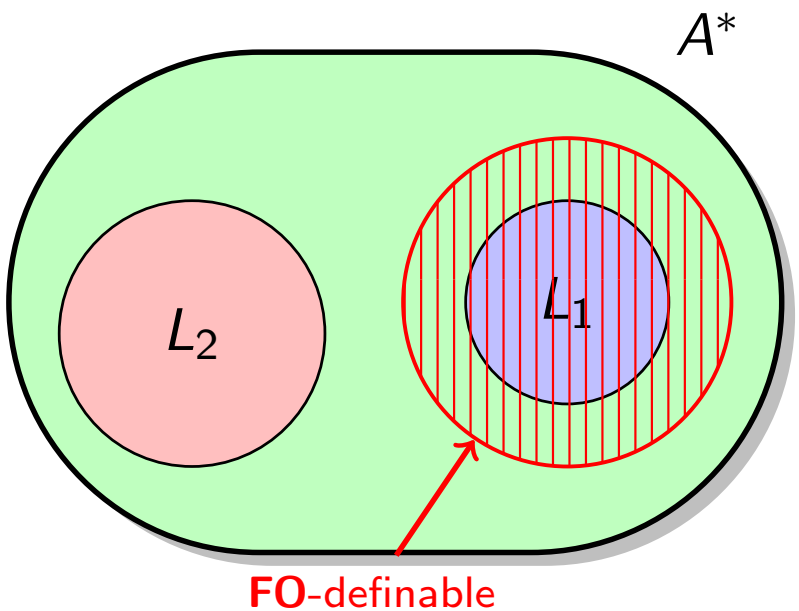
# Here comes Separation

Decide the following problem:

Take two regular languages $L_1, L_2$

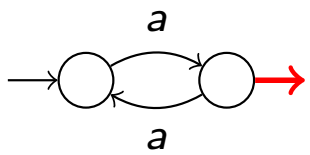



Can $L_1$ be separated from $L_2$ with a **FO** formula ?

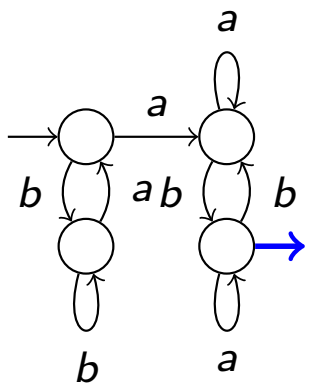


# A more General Problem than Characterization

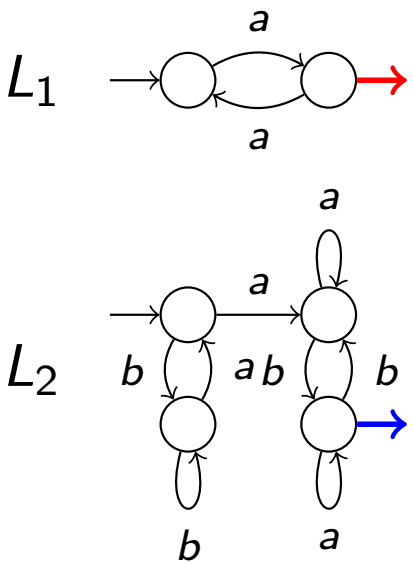
$L_1$



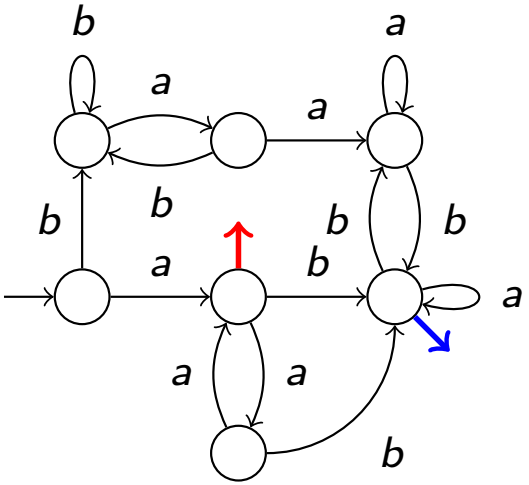

$L_2$



# 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 $\Rightarrow$ 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

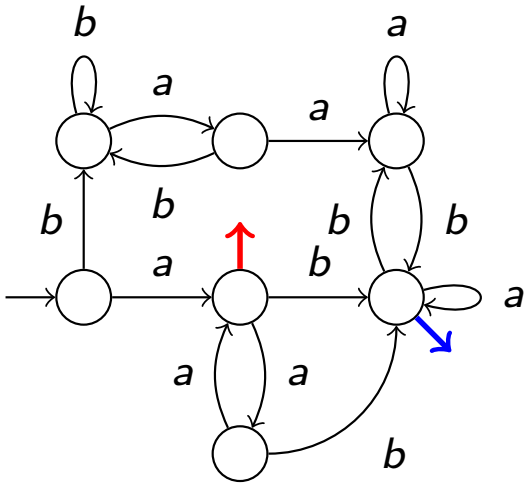
- We need **FO** techniques that apply to all languages, not just the **FO** definable $\Rightarrow$ More general and can be used in any context.
- In the **FO** characterization, $x^\omega = x^{\omega+1}$ is a specific counterexample that always happen. Here we need a way to describe all of them.
- 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

# First-example




Goal: Separate red language from blue language with **FO**.

Question: What computable information is sufficient to solve that problem ?

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

Quantifier rank of a formula: Nested depth of quantifiers.

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

If $k$ fixed: finitely many **FO** properties of rank $k \Rightarrow$ 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.

$$\forall x \exists y\ (a(x) \wedge \exists z\ (x < z < y \wedge b(y))) \quad \text{has quantifier rank 3}$$

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

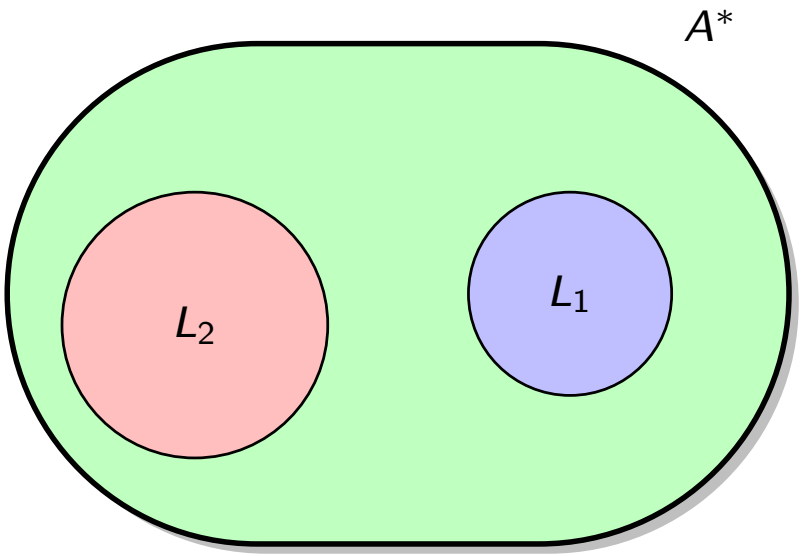
**$k$-equivalence for FO**

Let $w_1, w_2$ be words:

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

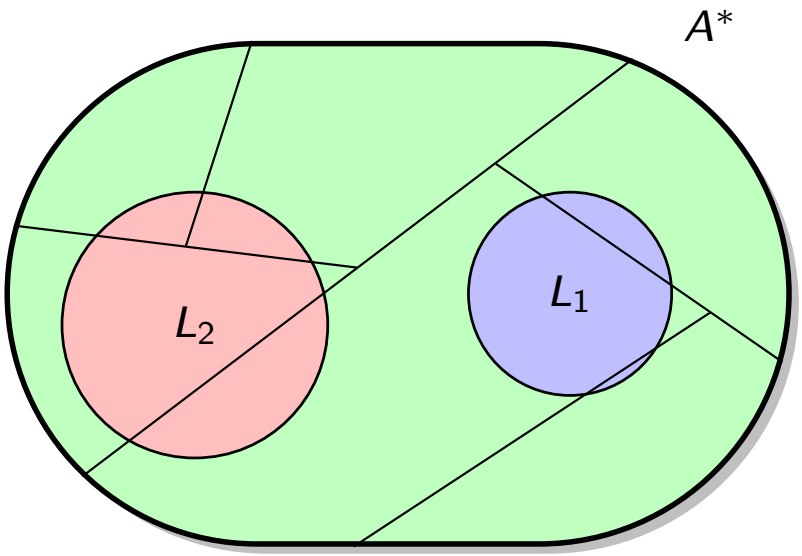
*All* **FO** *properties of rank k are unions of classes of* $\cong_k$.

# Fixed Quantifier Rank $k$




Let's add the $\cong_k$-classes

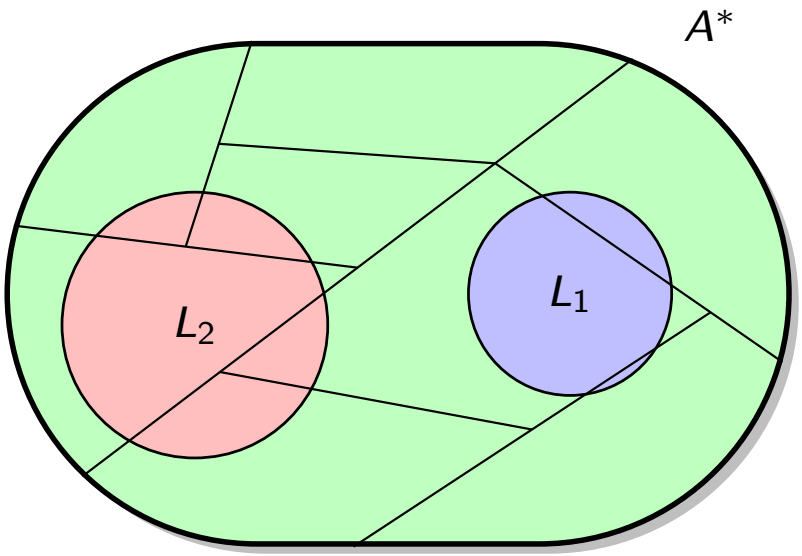
# Fixed Quantifier Rank $k$




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 $k$

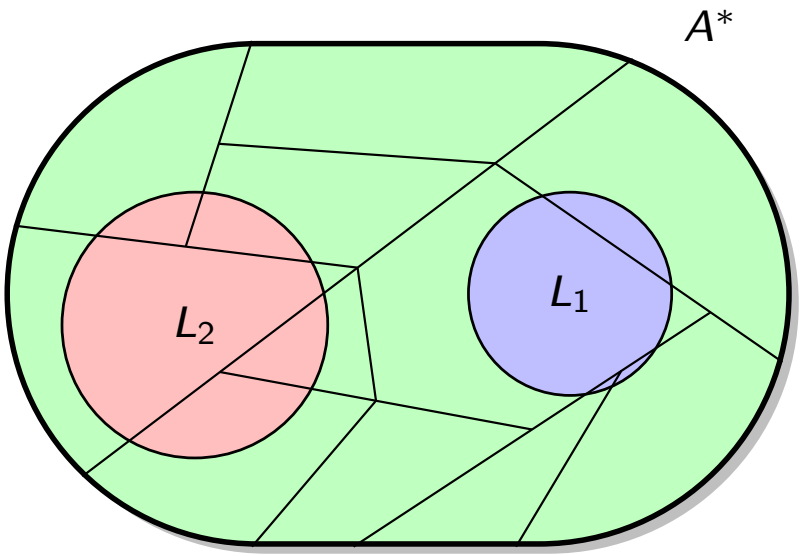



Separable with rank $k$ iff no $\cong_k$-class intersects both languages

For full **FO** we want to know if there exists a $k$
$\Rightarrow$ Compute a 'limit' for $\cong_k$.

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

# Fixed Quantifier Rank $k$

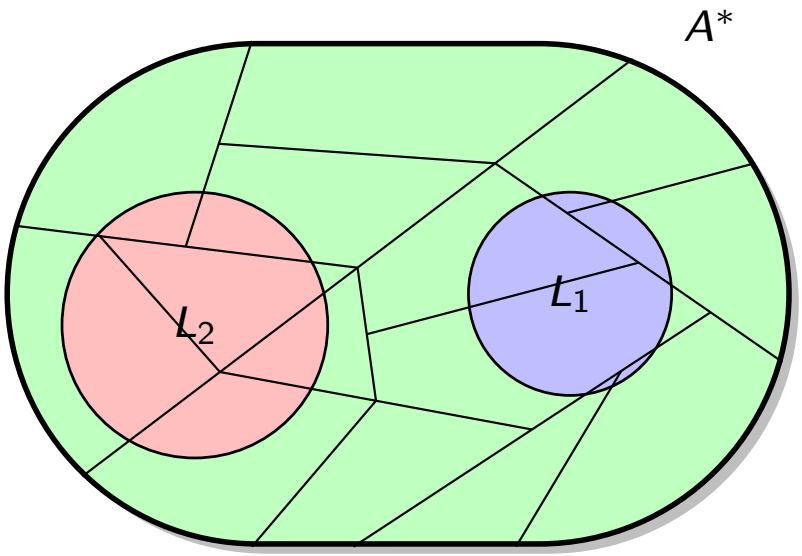



Separable with rank $k$ iff no $\cong_k$-class intersects both languages

For full **FO** we want to know if there exists a $k$
$\Rightarrow$ Compute a 'limit' for $\cong_k$.

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

# Fixed Quantifier Rank $k$

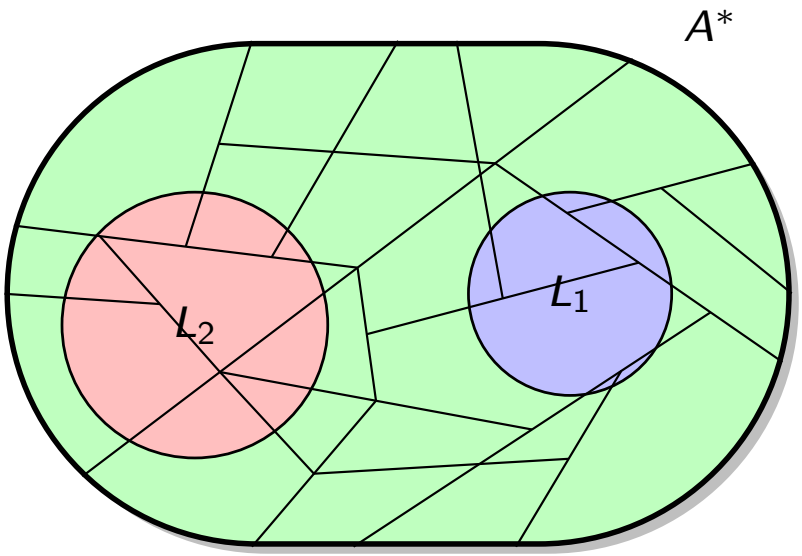



Separable with rank $k$ iff no $\cong_k$-class intersects both languages

For full **FO** we want to know if there exists a $k$
$\Rightarrow$ 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 $k$
$\Rightarrow$ Compute a 'limit' for $\cong_k$.

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

# Fixed Quantifier Rank $k$

We have an Automata $\mathcal{A}$

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

- $q_1 \cong_k q_2$ 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.
- $q_1 \cong q_2$ 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

# Fixed Quantifier Rank $k$

We have an Automata $\mathcal{A}$

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

- $q_1 \cong_k q_2$ 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

Warning very bad notation (not transitive)

$\Rightarrow$ (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

# Fixed Quantifier Rank $k$

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

# 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. $(q_1, q_2) \in P(\mathcal{A})$.

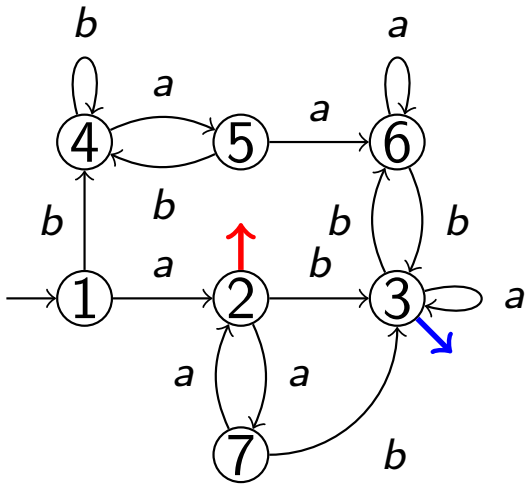
# The Separation Criterion

**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. $(q_1, q_2) \in P(\mathcal{A})$.

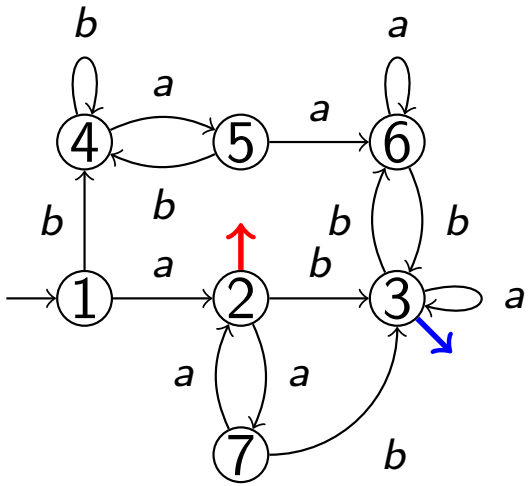
Computing $P(\mathcal{A})$ suffices to solve separation by **FO**. However it gives much more.

# 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

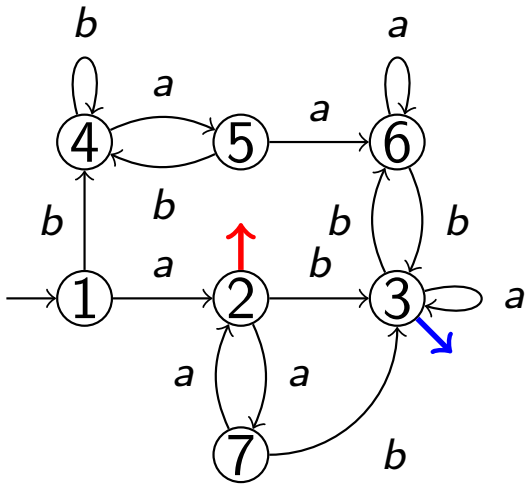



$P(\mathcal{A}) = \{(1,1),(2,2),(3,3),(4,4),$
$(5,5),(6,6),(7,7),(3,6),(6,3),(2,7),(7,2)\}$

$\Rightarrow$ This is the information we wanted

"Everything" **FO** can express about $\mathcal{A}$

# Back to our example




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

$\Rightarrow$ This is the information we wanted

"Everything" **FO** cannot express about $\mathcal{A}$

# 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 $q_i \xrightarrow{w_1} q_1$
$q_i \xrightarrow{w_2} q_2$
- $(q_1, q_2) \in P(\mathcal{A})$ iff
$\forall k\ (q_1, q_2) \in P_k(\mathcal{A})$

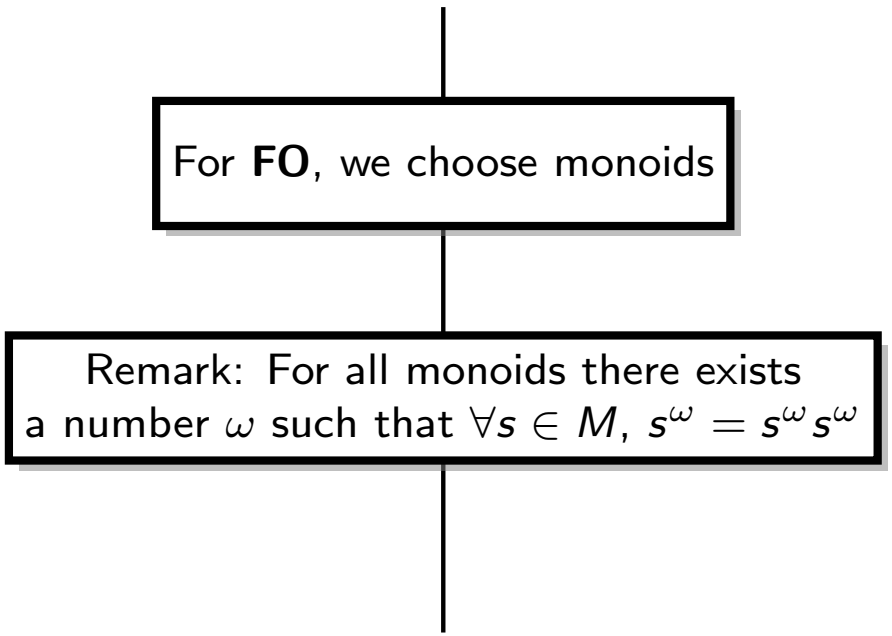
# Levels of Precision, what do we mean by everything ?

| Final States | Intial and Final States |
| --- | --- |
| • Automata $\mathcal{A}$. | • 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 $q_i \xrightarrow{w_1} q_1$, $q_i \xrightarrow{w_2} q_2$ | • $(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$ and $r_1 \xrightarrow{w_1} q_1$, $r_2 \xrightarrow{w_2} q_2$ |
| • $(q_1, q_2) \in P(\mathcal{A})$ iff $\forall k$ $(q_1, q_2) \in P_k(\mathcal{A})$ | • $(r_1, q_1, r_2, q_2) \in P(\mathcal{A})$ iff $\forall k$ $(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ |

# Levels of Precision, what do we mean by everything ?

| Final States | Intial and Final States | Monoids |
|---|---|---|
| • Automata $\mathcal{A}$. | • Automata $\mathcal{A}$. | • Morphism $\alpha : A^* \to M$. |
| • $(q_1, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$ and $q_i \xrightarrow{w_1} q_1$, $q_i \xrightarrow{w_2} q_2$ | • $(r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ iff $\exists w_1, w_2 \in A^*$ s.t. $w_1 \cong_k w_2$ and $r_1 \xrightarrow{w_1} q_1$, $r_2 \xrightarrow{w_2} q_2$ | • $(s_1, s_2) \in P_k(M)$ iff $\exists w_1, w_2$ s.t. $w_1 \cong_k w_2$ and $\alpha(w_1) = s_1$, $\alpha(w_2) = s_2$ |
| • $(q_1, q_2) \in P(\mathcal{A})$ iff $\forall k\ (q_1, q_2) \in P_k(\mathcal{A})$ | • $(r_1, q_1, r_2, q_2) \in P(\mathcal{A})$ iff $\forall k\ (r_1, q_1, r_2, q_2) \in P_k(\mathcal{A})$ | • $(s_1, s_2) \in P(M)$ iff $\forall k\ (s_1, s_2) \in P_k(M)$ |

# Levels of Precision, what do we mean by everything ?




Monoids

- Morphism $\alpha : A^* \to M$.
- $(s_1, s_2) \in P_k(M)$ 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
  $\forall k\ (s_1, s_2) \in P_k(M)$

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

Property of **FO**
$\forall k\ \forall w \in A^*\ \ w \cong_k w$

# A first (non complete) Algorithm

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

Property of **FO**

$\forall k\ \forall w_1, w_2, u_1, u_2 \in A^*\ w_1 \cong_k w_2$ and $u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$

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

# A first (non complete) Algorithm

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

Property of **FO**

$\forall k\ \forall w_1, w_2, u_1, u_2 \in A^*\ w_1 \cong_k w_2$ and $u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$

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

# A first (non complete) Algorithm

We have $\alpha : A^* \to M$ 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 $\alpha : A^* \to M$ 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 $(t_1, t_2) \in P(M) \Rightarrow (s_1 t_1, s_2 t_2) \in P(M)$
- 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 $\alpha : A^* \to M$ 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 $(t_1, t_2) \in P(M) \Rightarrow (s_1 t_1, s_2 t_2) \in P(M)$
- Operation: $(s_1, s_2) \in P(M) \Rightarrow ((s_1)^\omega, (s_2)^{\omega+1}) \in P(M)$ and $((s_1)^{\omega+1}, (s_2)^\omega) \in P(M)$

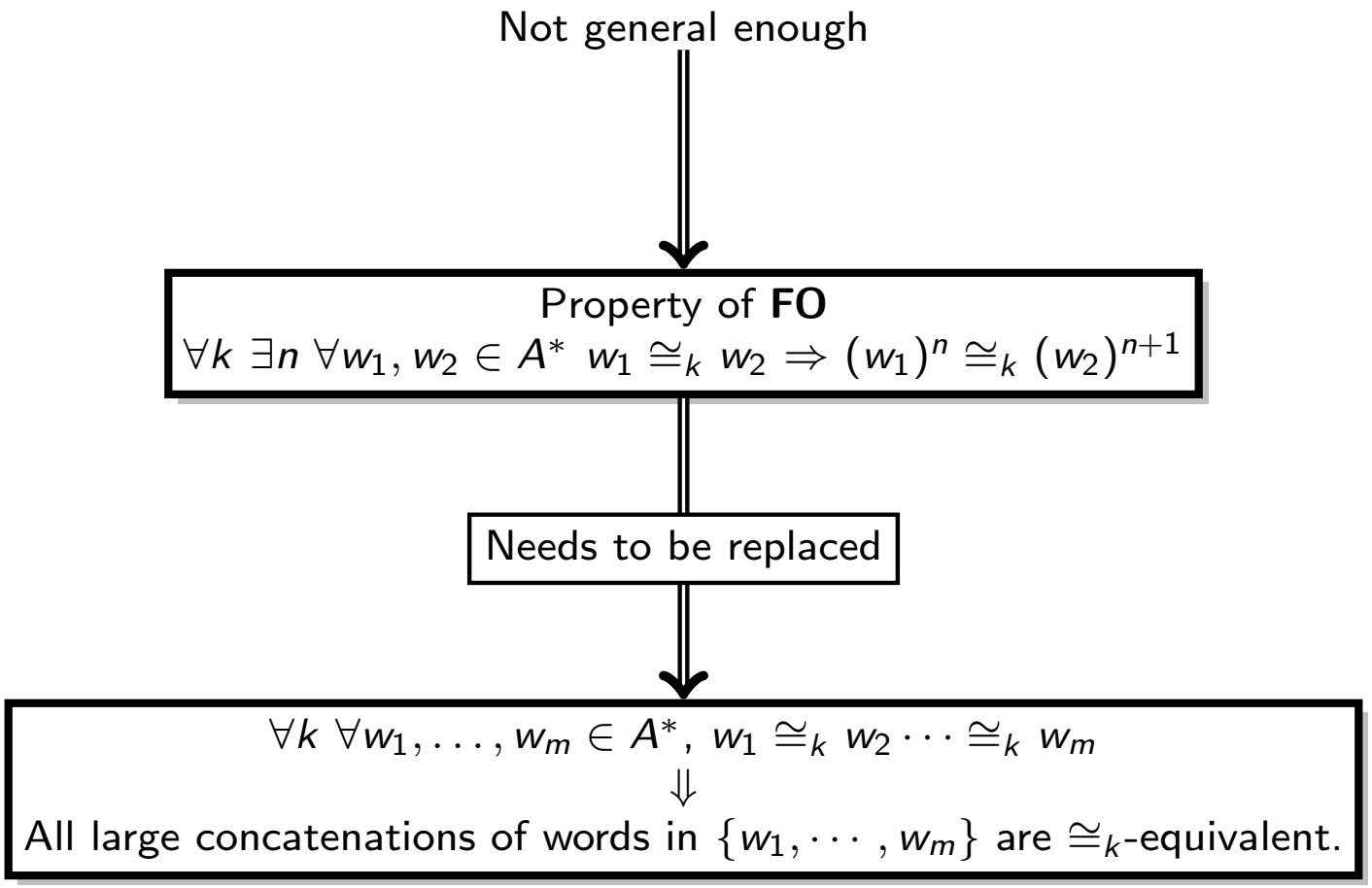
Correct by definition but not complete
We now explain why

# Why it does not work

Property of **FO**

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

# Why it does not work



# From Pairs to Sets

For using this more general property, pairs are not general enough $\Rightarrow$ Use sets.

For all $k$, we set $\mathcal{S}_k(M) \subseteq 2^M$ the set s.t.:

$$\{s_1, s_2, s_3\} \in \mathcal{S}_k \text{ iff } \exists w_1, w_2, w_3 \in A^* \text{ s.t. } w_1 \cong_k w_2 \cong_k w_3 \text{ and } \begin{array}{l} \alpha(w_1) = s_1 \\ \alpha(w_2) = s_2 \\ \alpha(w_3) = s_3 \end{array}$$

## New Objective

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

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

# From Pairs to Sets

For using this more general property, pairs are not general enough $\Rightarrow$ Use sets.

For all $k$, we set $\mathcal{S}_k(M) \subseteq 2^M$ the set s.t.:

$$\{s_1, s_2, s_3\} \in \mathcal{S}_k \text{ iff } \exists w_1, w_2, w_3 \in A^* \text{ s.t. } w_1 \cong_k w_2 \cong_k w_3 \text{ and } \begin{array}{l} \alpha(w_1) = s_1 \\ \alpha(w_2) = s_2 \\ \alpha(w_3) = s_3 \end{array}$$

## New Objective

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

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

## Remark

$2^M$ is still 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

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

Property of **FO**
$\forall k \; \forall w \in A^* \;\; w \cong_k w$

# A new (working) Algorithm

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

Property of **FO**
$\forall k \; \forall w \in A^* \;\; w \cong_k w$

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

# A new (working) Algorithm

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

**Property of FO**

$\forall k\ \forall w_1, w_2, u_1, u_2 \in A^*\ w_1 \cong_k w_2$ and $u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$

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

# A new (working) Algorithm

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

**Property of FO**

$\forall k\ \forall w_1, w_2, u_1, u_2 \in A^*\ w_1 \cong_k w_2$ and $u_1 \cong_k u_2 \Rightarrow w_1 u_1 \cong_k w_2 u_2$

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

# A new (working) Algorithm

We have $\alpha : A^* \to M$ and want to compute $\mathcal{S}(M)$. 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 $w \in A^*$ $\{\alpha(w)\} \in \mathcal{S}(M)$
- Operation: $S_1 \in \mathcal{S}(M)$ and $S_2 \in \mathcal{S}(M) \Rightarrow S_1 S_2 \in \mathcal{S}(M)$

# A new (working) Algorithm

We have $\alpha : A^* \to M$ and want to compute $\mathcal{S}(M)$. 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 $w \in A^*$ $\{\alpha(w)\} \in \mathcal{S}(M)$
- Operation: $S_1 \in \mathcal{S}(M)$ and $S_2 \in \mathcal{S}(M) \Rightarrow S_1 S_2 \in \mathcal{S}(M)$
- Operation: $S \in \mathcal{S}(M) \Rightarrow (S^\omega \cup S^{\omega+1}) \in \mathcal{S}(M)$

# A new (working) Algorithm

We have $\alpha : A^* \to M$ and want to compute $\mathcal{S}(M)$. 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 $w \in A^*$ $\{\alpha(w)\} \in \mathcal{S}(M)$
- Operation: $S_1 \in \mathcal{S}(M)$ and $S_2 \in \mathcal{S}(M) \Rightarrow S_1 S_2 \in \mathcal{S}(M)$
- Operation: $S \in \mathcal{S}(M) \Rightarrow (S^\omega \cup S^{\omega+1}) \in \mathcal{S}(M)$

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

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

This also works

# Questions that remain

1. How do you prove completness ?
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 completness ?
   $\Rightarrow$ 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 completness ?
   $\Rightarrow$ By generalizing a well-known proof of the **FO** characterization by Wilke
2. Does this work for all logics ?
   $\Rightarrow$ 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 completness ?
   $\Rightarrow$ By generalizing a well-known proof of the **FO** characterization by Wilke
2. Does this work for all logics ?
   $\Rightarrow$ 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)?
   $\Rightarrow$ You can, but you will still have to be patient...

# Completeness

# Completness: What we need to prove

## Reminder

$\mathcal{S}(M) = \bigcap_{k \in \mathbb{N}} \mathcal{S}_k(M)$. In particular, for all $k$, $\mathcal{S}(M) \subseteq \mathcal{S}_k(M)$.

# Completness: What we need to prove

## Reminder

$\mathcal{S}(M) = \bigcap_{k \in \mathbb{N}} \mathcal{S}_k(M)$. In particular, for all $k$, $\mathcal{S}(M) \subseteq \mathcal{S}_k(M)$.

## What we prove

For $k = |M|(2^{|M|})!$, the algorithm computes $\mathcal{S}_k(M)$. $\Rightarrow$ 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 $Gen_k(w) \in \mathcal{S}_k$:

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

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.

# 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 $\alpha(w)$ ?

---

$w = \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$

# 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 $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

$w =$ .....

# 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 $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

$w =$ .....

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

# 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 $\alpha(a)M \subsetneq M$ or $M\alpha(a) \subsetneq M$

$w = w_0\ aw_1\, aw_2\, aw_3\, aw_4 \cdots\cdots\cdots\cdots aw_m$

# 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 $\alpha(w)$ ?

---

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

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

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 $\alpha(w)$ ?

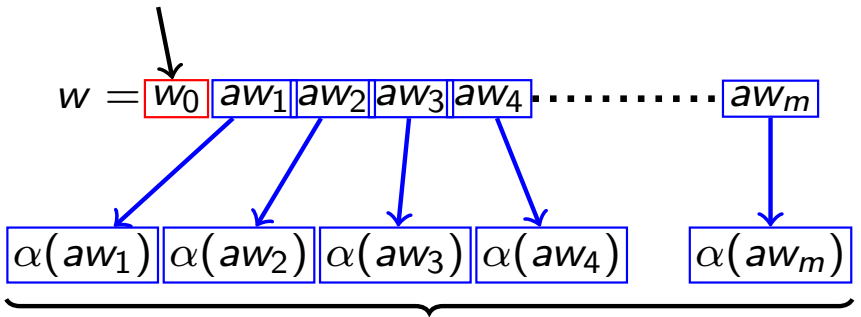
---

Two Cases:

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

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

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




New meta-word on alphabet $\alpha(a)M$
Use a morphism $\beta : (\alpha(a)M)^* \to \alpha(a)M \subsetneq M$
Detectable by induction on $|M|$

# Wilke Proof of the **FO** characterization

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

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

Disadvantage:

Advantage:

- For al
and $M\alpha$

- There
$\alpha(a)M$

$aw_m$

$(aw_m)$

$M$

$M \subsetneq M$

# Wilke Proof of the **FO** characterization

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

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

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

Advantage:

- For al
and $M\alpha$

- There
$\alpha(a)M$

$aw_m$

$(aw_m)$

$M$

$M \subsetneq M$

# Wilke Proof of the **FO** characterization

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

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

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

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

• For al
and $M\alpha$

• There
$\alpha(a)M$

$aw_m$

$(aw_m)$

$M$

$M \subsetneq M$

# Let's summarize what we have

1. An algorithm for computing $\mathcal{S}(M)$. Therefore, we can answer *yes* or *n* 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 $\mathcal{S}(M) = \{S_1, \ldots, S_n\}$. By reversing the completness 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 convering: $\{w \mid w \models \varphi_1 \vee \cdots \vee \models \varphi_n\} = A^*$.
- 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 $\mathcal{S}(M)$ is a union of these formulas.

# Conclusion of Part I

We have the following results:

- Separation by **FO** is decidable (in $\mathrm{EXPTIME}$).
- 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

# Separation on Words: State of the Art

Most well-known classes are solved:

- Piecewise Testable Languages ($\mathcal{B}\Sigma_1(<)$). [Czerwinski,Martens, and Masopust] and independently [P.,Rooijen and Zeitoun] (2013).
- Two-variables First-Order Logic ($\mathrm{FO}^2(<)$). [P.,Rooijen and Zeitoun] (2013).
- Locally Testable languages (**LT**). [P.,Rooijen and Zeitoun] (2013).
- Locally Threshold Testable languages (**LTT**). [P.,Rooijen and Zeitoun] (2013).
- First-Order Logic ($\mathrm{FO}(<)$). [P. and Zeitoun] (2013).

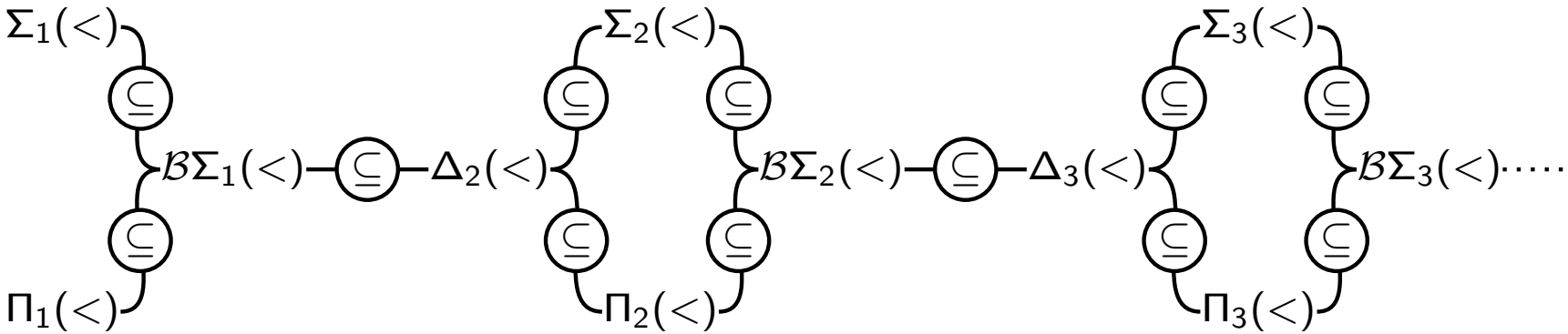
# Separation on Words: State of the Art

Most well-known classes are solved:

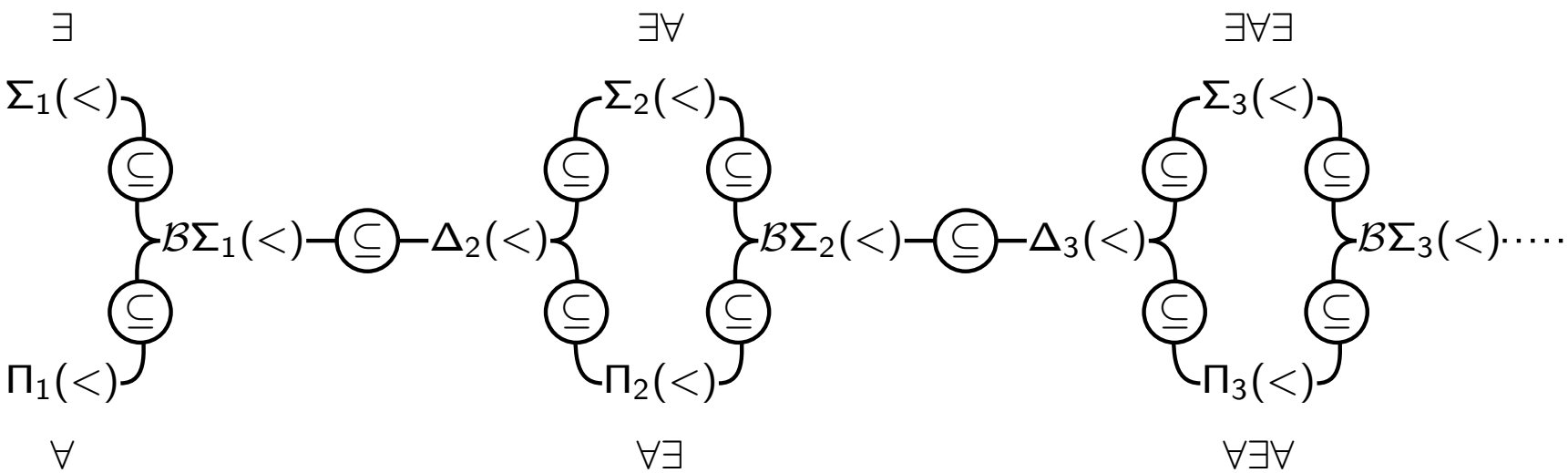
- Piecewise Testable Languages ($\mathcal{B}\Sigma_1(<)$). [Czerwinski,Martens, and Masopust] and independently [P.,Rooijen and Zeitoun] (2013).
- Two-variables First-Order Logic ($\mathrm{FO}^2(<)$). [P.,Rooijen and Zeitoun] (2013).
- Locally Testable languages (**LT**). [P.,Rooijen and Zeitoun] (2013).
- Locally Threshold Testable languages (**LTT**). [P.,Rooijen and Zeitoun] (2013).
- First-Order Logic ($\mathrm{FO}(<)$). [P. and Zeitoun] (2013).

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

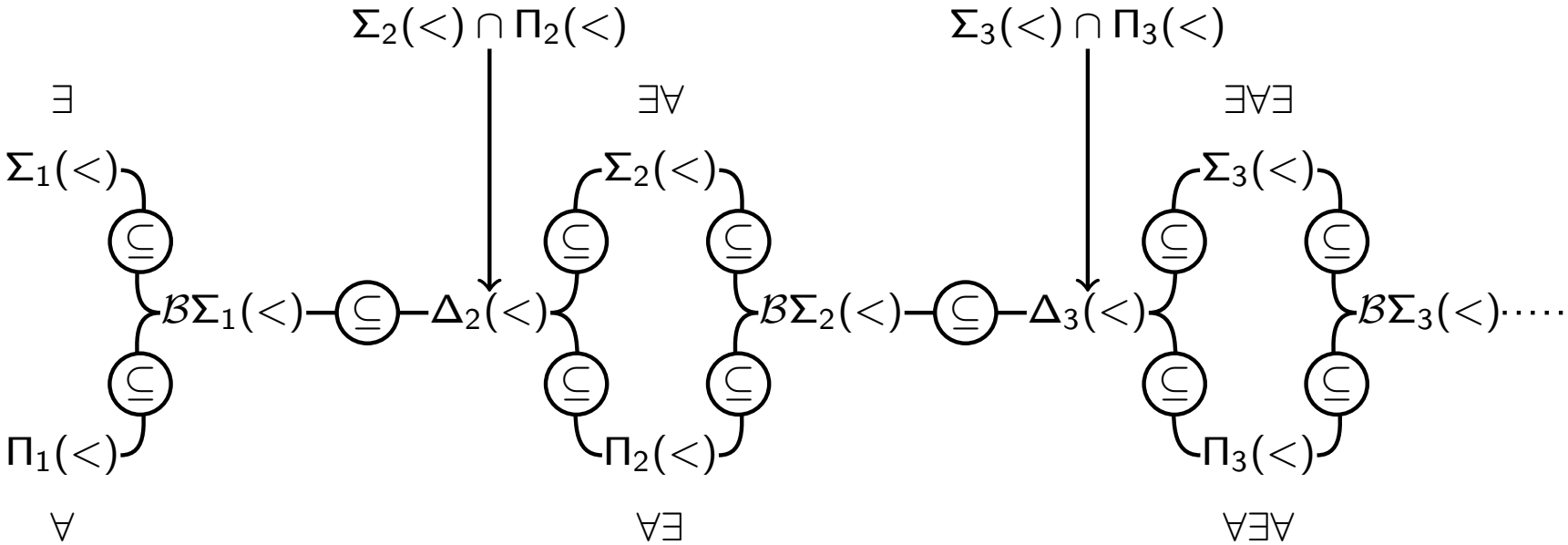
# **FO** Quantifier Alternation Hierarchy



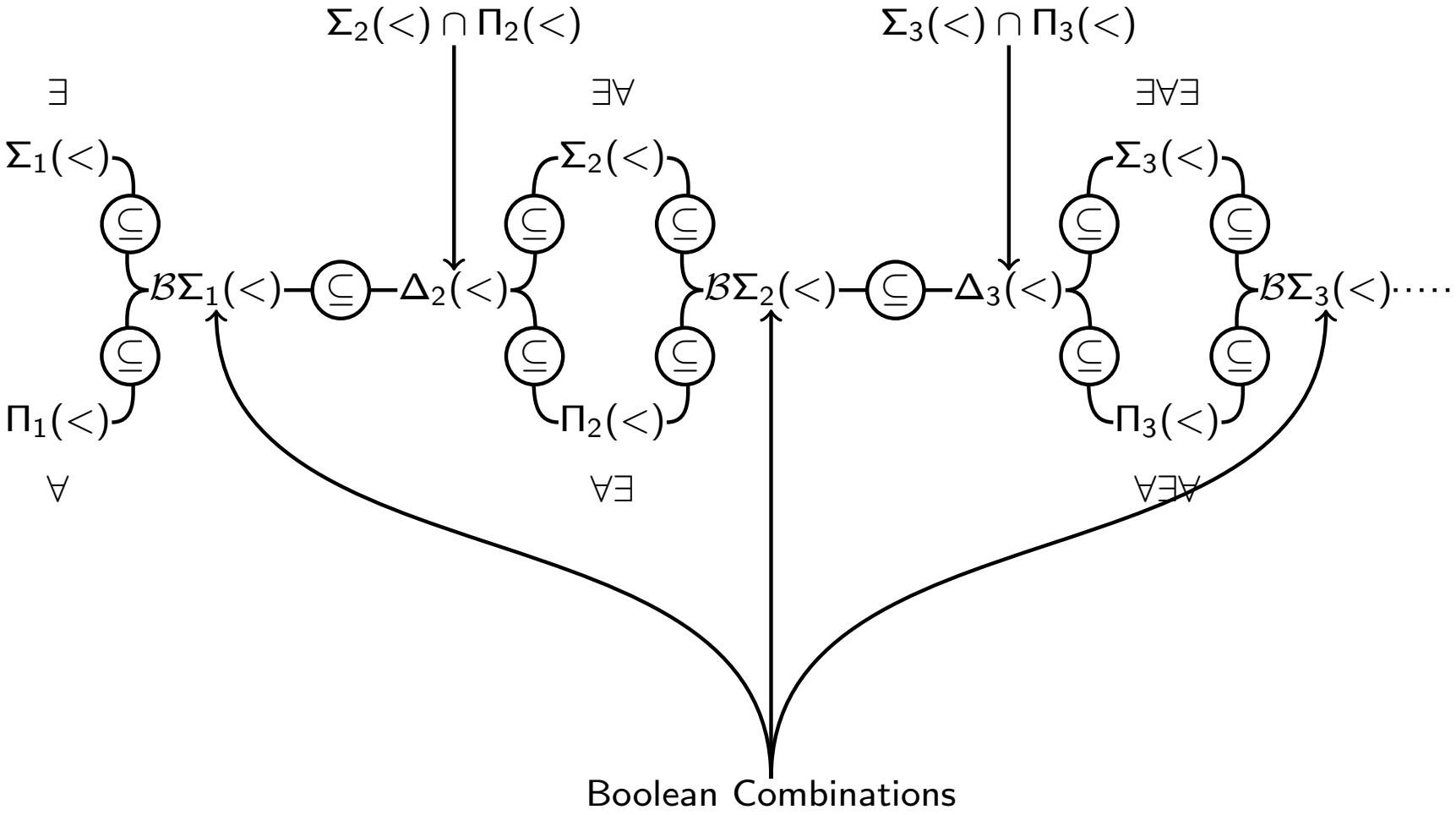
# **FO** Quantifier Alternation Hierarchy



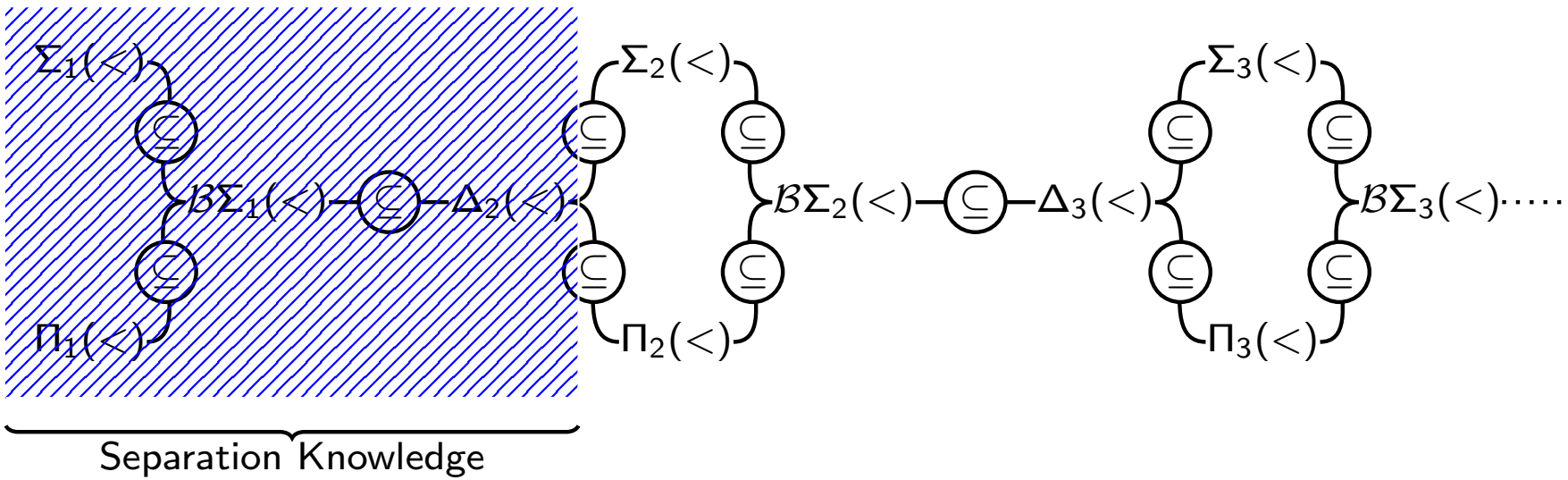
# **FO** Quantifier Alternation Hierarchy



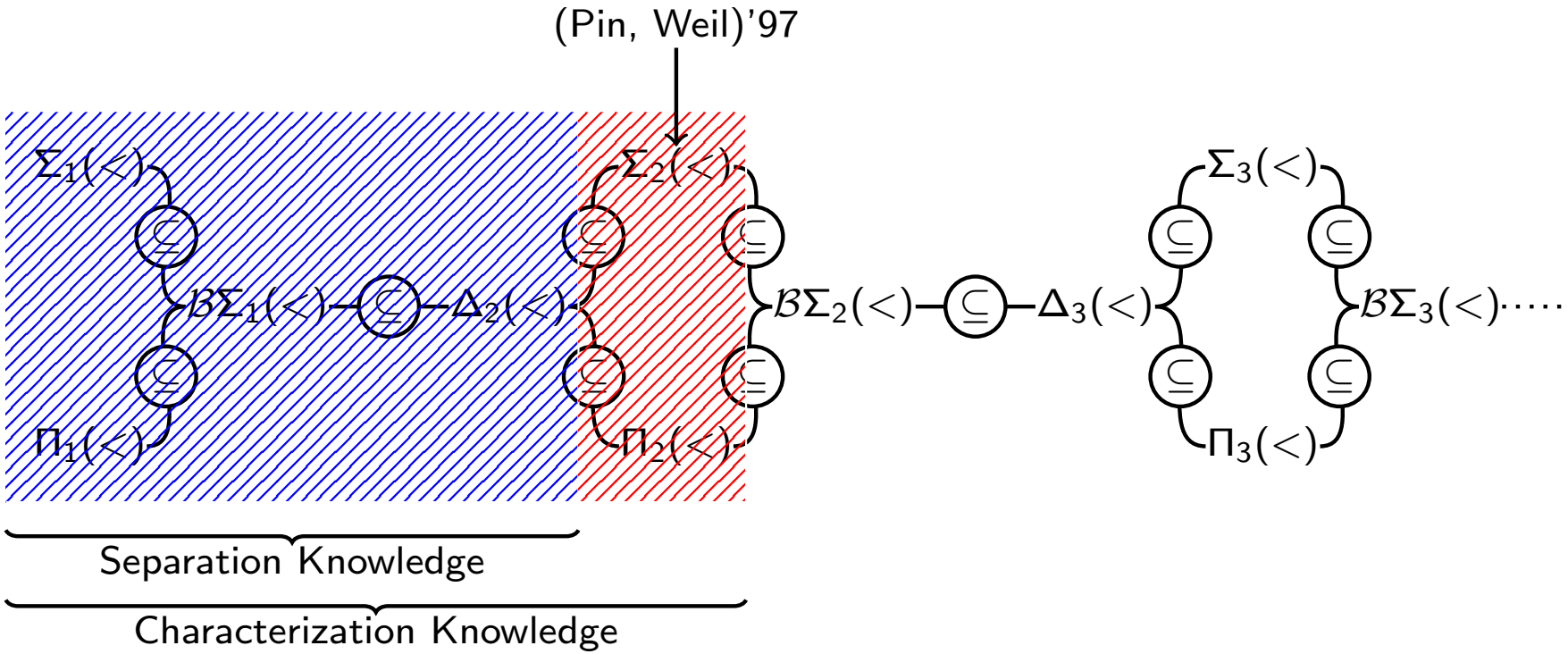
# **FO** Quantifier Alternation Hierarchy



# **FO** Quantifier Alternation Hierarchy

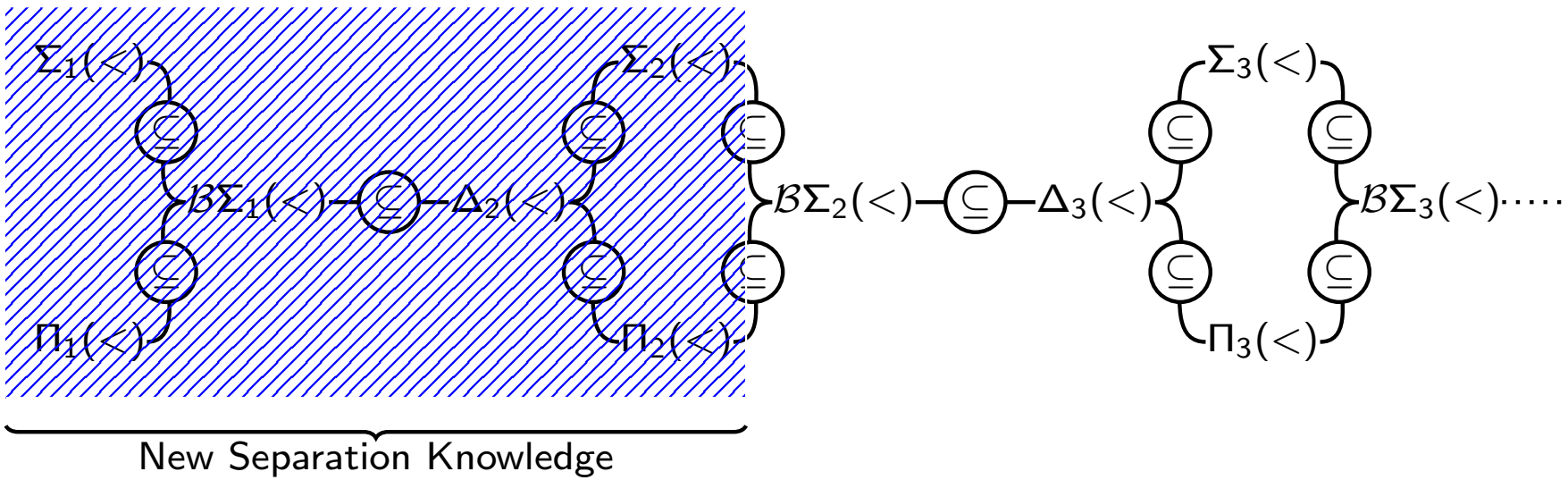


# **FO** Quantifier Alternation Hierarchy

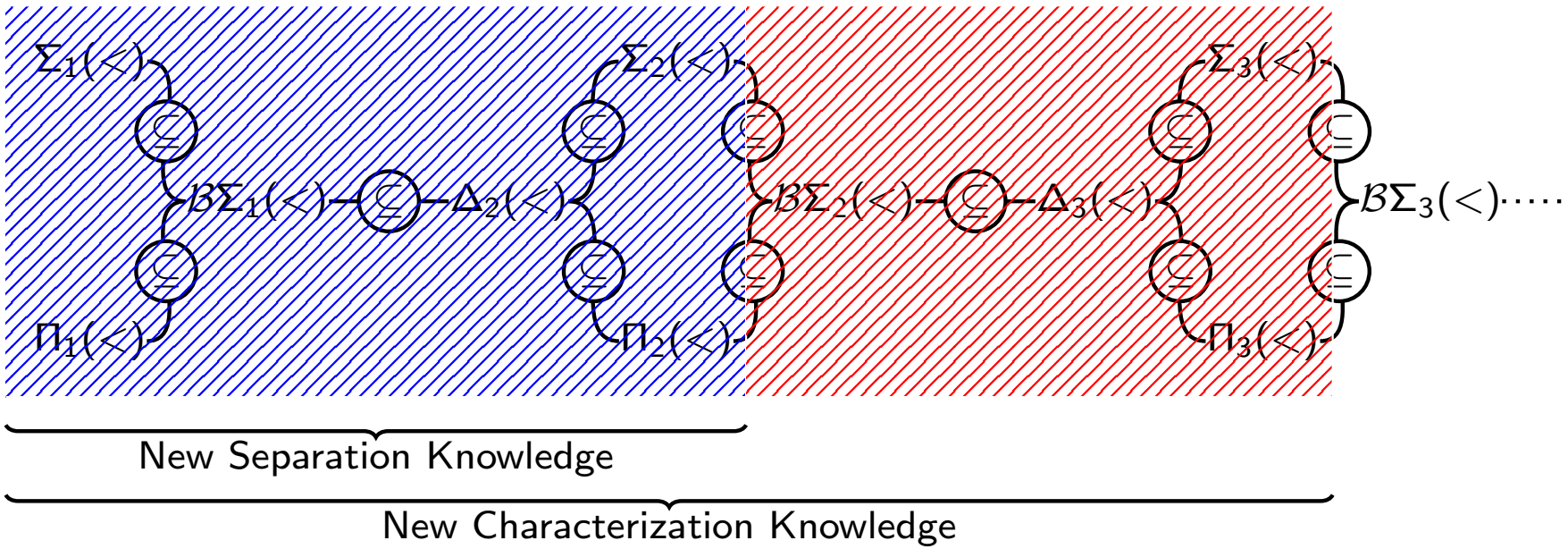



$\Sigma_2(<)$ and $\Pi_2(<)$ were open for separation

# **FO** Quantifier Alternation Hierarchy



# **FO** Quantifier Alternation Hierarchy




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

# Three Main Results

1. Separation by $\Sigma_2(<)$.
2. Characterization for $\Sigma_3(<)$.
3. Characterization for $\mathcal{B}\Sigma_2(<)$.

# Three Main Results

1. Separation by $\Sigma_2(<)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(<)$.
3. Characterization for $\mathcal{B}\Sigma_2(<)$.

# Three Main Results

1. Separation by $\Sigma_2(<)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(<)$. $\Rightarrow$ Not that Difficult.
3. Characterization for $\mathcal{B}\Sigma_2(<)$.

# Three Main Results

1. Separation by $\Sigma_2(<)$. $\Rightarrow$ Difficult.
2. Characterization for $\Sigma_3(<)$. $\Rightarrow$ Not that Difficult.
3. Characterization for $\mathcal{B}\Sigma_2(<)$. $\Rightarrow$ 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:

$$w_1 \lesssim_k^2 w_2 \text{ iff for any } \Sigma_2(<) \text{ formula } \Psi \text{ rank } k, (w_1 \models \Psi \Rightarrow w_2 \models \Psi)$$

*$L$ is $\Sigma_2(<)$ definable with rank $k$ iff* $L = \{w \mid \exists w' \in L \text{ s.t. } w' \lesssim_k^2 w\}$

# Separation and $\Sigma_2(<)$, what we compute (2)

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

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

What we want to compute (and can compute) is

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

This solves separation:

## Separation Solution

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

# Separation and $\Sigma_2(<)$, what we compute (2)

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

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

What we want to compute (and can compute) is

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

This solves separation:

## Separation Solution

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

However, computing $\mathcal{C}^2(M)$ gives much more.

# A Decidable Characterization for $\Sigma_3(<)$.

# Deriving a characterization for $\Sigma_3(<)$.

**Theorem**

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

$$s^\omega \leqslant s^\omega t s^\omega \quad \text{for } (t, s) \in \mathcal{C}^2$$

# Deriving a characterization for $\Sigma_3(<)$.

**Theorem**

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

$$s^\omega \leqslant s^\omega t s^\omega \quad \text{for } (t, s) \in \mathcal{C}^2$$

What does $\leqslant$ means ? The syntactic monoid comes fromes Myhill-Nerode Equivalence:

$$w \equiv w' \text{ iff } \forall u, v \; uwv \in L \Leftrightarrow uw'v \in L$$

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

$$w \leqslant w' \text{ iff } \forall u, v \; uwv \in L \Rightarrow uw'v \in L$$

# What of $\mathcal{B}\Sigma_2(<)$?

# 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 $\mathcal{C}^2_k(M) \subseteq M^*$ such that:

$$(s_1, \ldots, s_n) \in \mathcal{C}^2_k(M) \text{ iff } \exists w_1, \ldots, w_n \in A^* \text{ s.t. } w_1 \lesssim^2_k \cdots \lesssim^2_k w_n \text{ and } \begin{array}{r} \alpha(w_1) = s_1 \\ \vdots \\ \alpha(w_n) = s_n \end{array}$$

We set

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

## Fun Fact

$\mathcal{C}^2(M)$ 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 $\mathcal{C}_k^2(M) \subseteq M^*$ such that:

$$(s_1, \ldots, s_n) \in \mathcal{C}_k^2(M) \text{ iff } \exists w_1, \ldots, w_n \in A^* \text{ s.t. } w_1 \lesssim_k^2 \cdots \lesssim_k^2 w_n \text{ and } \begin{array}{r} \alpha(w_1) = s_1 \\ \vdots \\ \alpha(w_n) = s_n \end{array}$$

We set

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

## Fun Fact

$\mathcal{C}^2(M)$ 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 $(s_1 s_2)^* \subseteq \mathcal{C}^2(M)$.

# A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(<)$

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

- We do not know how to do it.

# A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(<)$

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

# A (non-effective) Separation Criterion for $\mathcal{B}\Sigma_2(<)$

Deciding separation for $\mathcal{B}\Sigma_2(<)$ 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 $\mathcal{C}^2(M)$.
- What if we only want a decidable characterization ?

# Deriving a characterization for $\mathcal{B}\Sigma_2(<)$. (half of it)

## Theorem

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

$$\begin{array}{lcll} s_1^\omega s_3^\omega & = & s_1^\omega s_2 s_3^\omega & \\ s_3^\omega s_1^\omega & = & s_3^\omega s_2 s_1^\omega & \text{for } (s_1, s_2, s_3) \in C^2(M) \end{array}$$

# Deriving a characterization for $\mathcal{B}\Sigma_2(<)$. (half of it)

## Theorem

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

$$\begin{array}{rcll} s_1^\omega s_3^\omega & = & s_1^\omega s_2 s_3^\omega & \text{for } (s_1, s_2, s_3) \in \mathcal{C}^2(M) \\ s_3^\omega s_1^\omega & = & s_3^\omega s_2 s_1^\omega & \end{array}$$

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

# Deriving a characterization for $\mathcal{B}\Sigma_2(<)$. (half of it)

## Theorem

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

$$\begin{array}{rcll} s_1^\omega s_3^\omega & = & s_1^\omega s_2 s_3^\omega & \text{for } (s_1, s_2, s_3) \in \mathcal{C}^2(M) \\ s_3^\omega s_1^\omega & = & s_3^\omega s_2 s_1^\omega & \end{array}$$

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

## Decidability

It suffices to compute words of length 2 and 3 in $\mathcal{C}^2(M)$ 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 $\alpha : A^* \to M$ be its syntactic morphism. $L$ is definable in $\mathcal{B}\Sigma_2(<)$ iff $\alpha$ satisfies the two following equations:*

$$\begin{array}{rcll} s_1^\omega s_3^\omega & = & s_1^\omega s_2 s_3^\omega & \text{for } (s_1, s_2, s_3) \in \mathcal{C}^2(M) \\ s_3^\omega s_1^\omega & = & s_3^\omega s_2 s_1^\omega & \end{array}$$

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

## Decidability

It suffices to compute words of length 2 and 3 in $\mathcal{C}^2(M)$ 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 $\mathrm{FO}(<)$.
- Separation for $\Sigma_2(<)$.
- Decidable Characterization for $\mathcal{B}\Sigma_2(<)$.
- Decidbale Characterizations for $\Sigma_3(<)$, $\Pi_3(<)$ and $\Delta_3(<)$.

# Conclusion

We have the following results:

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

What is left:

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

# Thank You