From star-free closure to mixed polynomial closure: the incredible story of operators.

Thomas Place

LaBRI, Bordeaux University

May 31, 2022

Classes of regular languages
and their *investigation*

General context

- Setting: **finite words** and **regular languages** (alphabet A).
- Goal: investigate **sub-classes** of the regular languages.

General context

- Setting: **finite words** and **regular languages** (alphabet A).
- Goal: investigate **sub-classes** of the regular languages.

Each sub-class is based on a **piece of syntax** defining its languages:

Two main kinds

General context

- Setting: **finite words** and **regular languages** (alphabet A).
- Goal: investigate **sub-classes** of the regular languages.

Each sub-class is based on a **piece of syntax** defining its languages:

Two main kinds

1. Classes based on **regular expressions**:
   A restriction of the regular expression yields a sub-class.

General context

- Setting: **finite words** and **regular languages** (alphabet A).
- Goal: investigate **sub-classes** of the regular languages.

Each sub-class is based on a **piece of syntax** defining its languages:

Two main kinds

1. Classes based on **regular expressions**:
   A restriction of the regular expression yields a sub-class.
2. Classes based on **logic**. By Büchi's theorem, MSO = REG:
   A restriction of MSO yields a sub-class.

The overused historical example: **first-order logic**

First-order logic over words (FO(<))

- Word: sequence of labeled positions that can be quantified:

$$a\ b\ b\ b\ c\ a\ a\ a \quad \in A^*$$
$$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

- Two kinds of predicates:
  1. for each letter $a \in A$, $a(x)$ selects positions x with label "a".
  2. single binary predicate for the (strict) order: $x < y$.
- A sentence defines a language:

$$\exists x \exists y\ a(x) \wedge b(y) \wedge x < y \wedge (\forall z\ x < z < y \Rightarrow c(z))$$
$$\text{defines} \quad A^* a c^* b A^*$$

The overused historical example: **first-order logic**

First-order logic over words (FO(<))

- Word: sequence of labeled positions that can be quantified:

$$a\ b\ b\ b\ c\ a\ a\ a \quad \in A^*$$
$$0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9$$

- Two kinds of predicates:
  1. for each letter $a \in A$, $a(x)$ selects positions x with label "a".
  2. single binary predicate for the (strict) order: $x < y$.
- A sentence defines a language:

$$\exists x \exists y\ a(x) \wedge b(y) \wedge x < y \wedge (\forall z\ x < z < y \Rightarrow c(z))$$
$$\text{defines} \quad A^*ac^*bA^*$$

Informal objective

"Understand" the **expressive power** of FO(<):

- What regular languages can we express?
- What are those that we cannot express ?

The overused historical example: **star-free languages**

The class of **star-free languages** (SF) is the least one such that:

The overused historical example: **star-free languages**

The class of **star-free languages** (SF) is the least one such that:

Example of star free language: $A^*ac^*bA^*$ $(A = \{a, b, c\})$.

$$A^*ac^*bA^* = A^* \; a \; \overline{(A^*aA^* \cup A^*bA^*)} \; b \; A^*$$

The overused historical example: **star-free languages**

The class of **star-free languages** (SF) is the least one such that:

- contains $\emptyset$ (empty language) and A^* (universal language).

Example of star free language: $A^*ac^*bA^*$ $(A = \{a, b, c\})$.

$$A^*ac^*bA^* = A^* \; a \; \overline{(A^*aA^* \cup A^*bA^*)} \; b \; A^*$$

Theorem of McNaughton-Papert'71: SF = FO(<)

Given a language L, the following are equivalent:

- L may be defined by a **first-order logic** sentence (FO(<)) .
- L is **star-free** (i.e. $L \in$ SF).

The overused historical example: **star-free languages**

The class of **star-free languages** (SF) is the least one such that:

- contains $\emptyset$ (empty language) and A^* (universal language).
- closed under **union** and **complement**.

$$K, L \mapsto K \cup L \qquad\qquad K \mapsto \overline{K}$$

Example of star free language: $A^*ac^*bA^*$ $(A = \{a, b, c\})$.

$$A^*ac^*bA^* = A^* \; a \; \overline{(A^*aA^* \cup A^*bA^*)} \; b \; A^*$$

Theorem of McNaughton-Papert'71: SF = FO(<)

Given a language L, the following are equivalent:

- L may be defined by a **first-order logic** sentence (FO(<)) .
- L is **star-free** (i.e. $L \in$ SF).

The overused historical example: star-free languages

The class of **star-free languages** (SF) is the least one such that:

- contains $\emptyset$ (empty language) and A^* (universal language).
- closed under **union** and **complement**.

$$K, L \mapsto K \cup L \qquad\qquad K \mapsto \overline{K}$$

- closed under **marked concatenation**:

$$\text{for a letter } a \in A \qquad K, L \mapsto KaL$$

Example of star free language: $A^*ac^*bA^*$ $(A = \{a, b, c\})$.

$$A^*ac^*bA^* = A^* \; a \; \overline{(A^*aA^* \cup A^*bA^*)} \; b \; A^*$$

Theorem of McNaughton-Papert'71: SF = FO(<)

Given a language L, the following are equivalent:

- L may be defined by a **first-order logic** sentence (FO(<)) .
- L is **star-free** (i.e. $L \in$ SF).

What does it mean to "investigate" FO(<)/SF

Informal objective

"Understand" star-free languages and **expressive power** of FO(<).

What does it mean to "investigate" FO(<)/SF

Informal objective

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

What does it mean to "investigate" FO(<)/SF

Informal objective

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** of L is **aperiodic**.

What does it mean to "investigate" FO(<)/SF

Informal objective

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** of L is **aperiodic**.

What does this mean ? Why does this give a membership algorithm ?

Monoids, regular languages: the (very) quick reminder

The basics

- A^* is a **monoid** (multiplication is word concatenation).
- We may look at **morphisms** $\alpha : A^* \to M$ into a monoid M.
- A language L is **recognized** by α if $L = \alpha^{-1}(F)$ for $F \subseteq M$.

Monoids, regular languages: the (very) quick reminder

The basics

- A^* is a **monoid** (multiplication is word concatenation).
- We may look at **morphisms** $\alpha : A^* \to M$ into a monoid M.
- A language L is **recognized** by α if $L = \alpha^{-1}(F)$ for $F \subseteq M$.

Monoid definition of the regular languages

L **regular** iff L recognized by a morphism into a **finite monoid**.

Monoids, regular languages: the (very) quick reminder

The basics

- A^* is a **monoid** (multiplication is word concatenation).
- We may look at **morphisms** $\alpha : A^* \to M$ into a monoid M.
- A language L is **recognized** by α if $L = \alpha^{-1}(F)$ for $F \subseteq M$.

Monoid definition of the regular languages

L **regular** iff L recognized by a morphism into a **finite monoid**.

Syntactic morphism

For each regular language L, there exists a **canonical morphism into a finite monoid** $\alpha_L : A^* \to M_L$ recognizing L:

- α_L is called the **syntactic morphism** of L
- M_L is called that **syntactic monoid** of L

Monoids, regular languages: the (very) quick reminder

The basics

- A^* is a **monoid** (multiplication is word concatenation).
- We may look at **morphisms** $\alpha : A^* \to M$ into a monoid M.
- A language L is **recognized** by α if $L = \alpha^{-1}(F)$ for $F \subseteq M$.

Monoid definition of the regular languages

L **regular** iff L recognized by a morphism into a **finite monoid**.

Syntactic morphism

For each regular language L, there exists a **canonical morphism into a finite monoid** $\alpha_L : A^* \to M_L$ recognizing L:

- α_L is called the **syntactic morphism** of L
- M_L is called that **syntactic monoid** of L

The syntactic morphism **can be computed**.
(monoid counterpart of the minimal automaton).

What does it mean to "investigate" FO(<)/SF

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** M_L is **aperiodic**.

What does it mean to "investigate" FO(<)/SF

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** M_L satisfies $s^\omega = s^{\omega+1}$ for all $s \in M_L$.

"ω" is a computable number associated to a **finite** monoid M:
for every $t \in M$, t^ω is **idempotent** ($t^\omega = t^\omega t^\omega$).

What does it mean to "investigate" FO(<)/SF

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** M_L satisfies $s^\omega = s^{\omega+1}$ for all $s \in M_L$.
   $\Rightarrow$ Decidable condition.

"ω" is a computable number associated to a **finite** monoid M:
for every $t \in M$, t^ω is **idempotent** ($t^\omega = t^\omega t^\omega$).

What does it mean to "investigate" FO(<)/SF

"Understand" star-free languages and **expressive power** of FO(<).

Standard approach: **membership algorithm** for SF = FO(<):

- **INPUT**: A regular language L.
- **OUTPUT**: Decide if L is star-free (*i.e.* $L \in \mathrm{SF}$).

Solution obtained from Schützenberger's theorem (1965)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **syntactic monoid** M_L satisfies $s^\omega = s^{\omega+1}$ for all $s \in M_L$.
   $\Rightarrow$ Decidable condition.

"ω" is a computable number associated to a **finite** monoid M:
for every $t \in M$, t^ω is **idempotent** ($t^\omega = t^\omega t^\omega$).

Important: constructive proof for 2) $\Rightarrow$ 1)

Normal forms for descriptions of star-free/first-order languages.

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.
- FO($<, MOD$): linear order, **modular predicates**.

for $d, m \in \mathbb{N}$, predicate $M_{d,m}(x)$ expressing,
"position x is congruent to d modulo m".

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.
- FO($<, MOD$): linear order, **modular predicates**.

for $d, m \in \mathbb{N}$, predicate $M_{d,m}(x)$ expressing,
"position x is congruent to d modulo m".

- FO($<, AMOD$): linear order, **alphabetic modular predicates**.

for $a \in A$ and $d, m \in \mathbb{N}$, predicate $M^a_{d,m}(x)$ expressing,
"there are d modulo m positions $y < x$ with label "a".

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.
- FO($<, MOD$): linear order, **modular predicates**.

for $d, m \in \mathbb{N}$, predicate $M_{d,m}(x)$ expressing,
"position x is congruent to d modulo m".

- FO($<, AMOD$): linear order, **alphabetic modular predicates**.

for $a \in A$ and $d, m \in \mathbb{N}$, predicate $M^a_{d,m}(x)$ expressing,
"there are d modulo m positions $y < x$ with label "a".

- FO($+1$): successor (binary predicate "$x + 1 = y$").

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.
- FO($<, MOD$): linear order, **modular predicates**.

for $d, m \in \mathbb{N}$, predicate $M_{d,m}(x)$ expressing,
"position x is congruent to d modulo m".

- FO($<, AMOD$): linear order, **alphabetic modular predicates**.

for $a \in A$ and $d, m \in \mathbb{N}$, predicate $M^a_{d,m}(x)$ expressing,
"there are d modulo m positions $y < x$ with label "a".

- FO($+1$): successor (binary predicate "$x + 1 = y$").

We want to **simultaneously investigate** all "natural" variants
$\Rightarrow$ more thorough understanding of FO.

Do we truly understand the "whole" first-order logic ?

Many "variants" of first-order logic: each associated to a **signature**.

- FO($<$): linear ordering.
- FO($<, MOD$): linear order, **modular predicates**.

for $d, m \in \mathbb{N}$, predicate $M_{d,m}(x)$ expressing,
"position x is congruent to d modulo m".

- FO($<, AMOD$): linear order, **alphabetic modular predicates**.

for $a \in A$ and $d, m \in \mathbb{N}$, predicate $M^a_{d,m}(x)$ expressing,
"there are d modulo m positions $y < x$ with label "a".

- FO($+1$): successor (binary predicate "$x + 1 = y$").

We want to **simultaneously investigate** all "natural" variants
$\Rightarrow$ more thorough understanding of FO.

In the talk, we look at variants allowing **at least the linear order**.

Defining families of logical classes:
Fragments of FO as ***operators***

Signature *associated to a class of languages* $\mathcal{C}$

We associate a **signature $\mathbb{S}_{\mathcal{C}}$ to every class $\mathcal{C}$**. It contains the labels and, for each $L \in \mathcal{C}$, a binary predicate $F_L(x, y)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models F_L(i, j) \iff i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L.$$

Signature *associated to a class of languages* $\mathcal{C}$

We associate a **signature $\mathbb{S}_{\mathcal{C}}$ to every class $\mathcal{C}$**. It contains the labels and, for each $L \in \mathcal{C}$, a binary predicate $F_L(x, y)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models F_L(i, j) \iff i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L.$$

- For each class $\mathcal{C}$: we consider the logical class $\mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.

Signature *associated to a class of languages* $\mathcal{C}$

We associate a **signature $\mathbb{S}_{\mathcal{C}}$ to every class $\mathcal{C}$**. It contains the labels and, for each $L \in \mathcal{C}$, a binary predicate $F_L(x, y)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models F_L(i, j) \iff i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L.$$

- For each class $\mathcal{C}$: we consider the logical class $\mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
- More generally, we may look at **fragments of FO**:
  $\Rightarrow$ for each fragment $\mathcal{F}$, this define an operator $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

Signature *associated to a class of languages* $\mathcal{C}$

We associate a **signature $\mathbb{S}_{\mathcal{C}}$ to every class $\mathcal{C}$**. It contains the labels and, for each $L \in \mathcal{C}$, a binary predicate $F_L(x, y)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models F_L(i, j) \iff i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L.$$

- For each class $\mathcal{C}$: we consider the logical class $\mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
- More generally, we may look at **fragments of FO**:
  $\Rightarrow$ for each fragment $\mathcal{F}$, this define an operator $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

Example: the trivial class $\mathcal{C} = \{\emptyset, A^*\}$

- $F_{A^*}(x, y)$ is equivalent to $x < y$.
- $F_{\emptyset}(x, y)$ is equivalent to $\bot$ (false).
- $\Rightarrow$ we get $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}}) = \mathcal{F}(<)$.

Signature *associated to a class of languages* $\mathcal{C}$

We associate a **signature $\mathbb{S}_{\mathcal{C}}$ to every class $\mathcal{C}$**. It contains the labels and, for each $L \in \mathcal{C}$, a binary predicate $F_L(x, y)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models F_L(i, j) \iff i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L.$$

- For each class $\mathcal{C}$: we consider the logical class $\mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
- More generally, we may look at **fragments of FO**:
  $\Rightarrow$ for each fragment $\mathcal{F}$, this define an operator $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

Example: the trivial class $\mathcal{C} = \{\emptyset, A^*\}$

- $F_{A^*}(x, y)$ is equivalent to $x < y$.
- $F_{\emptyset}(x, y)$ is equivalent to $\bot$ (false).
- $\Rightarrow$ we get $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}}) = \mathcal{F}(<)$.

Definition robust when $\mathcal{C}$ is a **Boolean algebra** (union, intersection and complement) closed under **quotients**.

Key input classes: **Group languages** (1)

- Recognized by a **morphism into a finite group**.
- Equivalently recognized by a **permutation automaton**.

Permutation Automaton

Complete deterministic finite automaton (DFA) such that:

each letter induces a **permutation of the states**

Key input classes: **Group languages** (1)

- Recognized by a **morphism into a finite group**.
- Equivalently recognized by a **permutation automaton**.

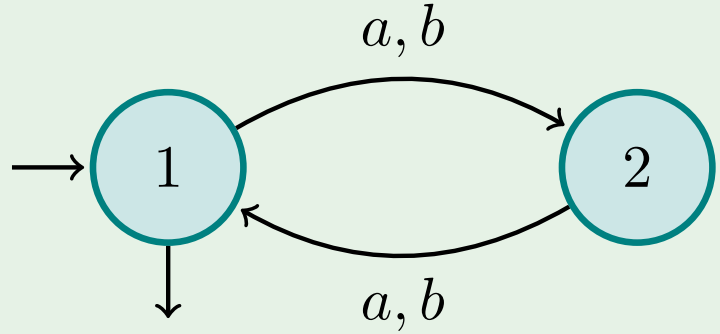
Permutation Automaton

Complete deterministic finite automaton (DFA) such that:

each letter induces a **permutation of the states**

Example 1: even length ($A = \{a, b\}$)

$$
\begin{aligned}
a &: 1 \mapsto 2 \\
&\quad 2 \mapsto 1 \\
b &: 1 \mapsto 2 \\
&\quad 2 \mapsto 1
\end{aligned}
$$



Represents the class MOD: **modulo languages**.

Key input classes: **Group languages** (1)

- Recognized by a **morphism into a finite group**.
- Equivalently recognized by a **permutation automaton**.

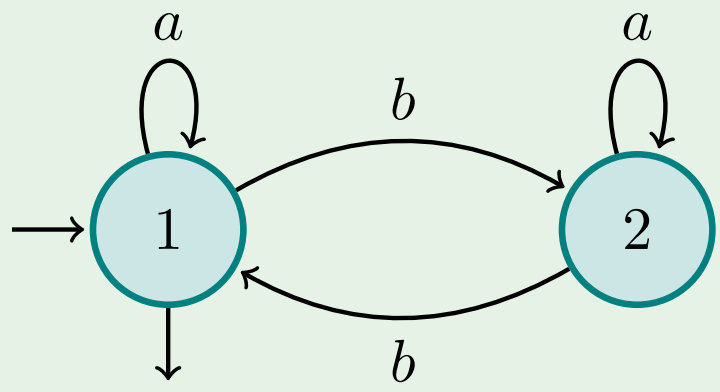
Permutation Automaton

Complete deterministic finite automaton (DFA) such that:

each letter induces a **permutation of the states**

Example 2: even number of "b", abelian language ($A = \{a, b\}$)

$$a : 1 \mapsto 1$$
$$\quad 2 \mapsto 2$$
$$b : 1 \mapsto 2$$
$$\quad 2 \mapsto 1$$



Represents the class AMT: **alphabet modulo testable languages**.

Key input classes: **Group languages** (1)

- Recognized by a **morphism into a finite group**.
- Equivalently recognized by a **permutation automaton**.

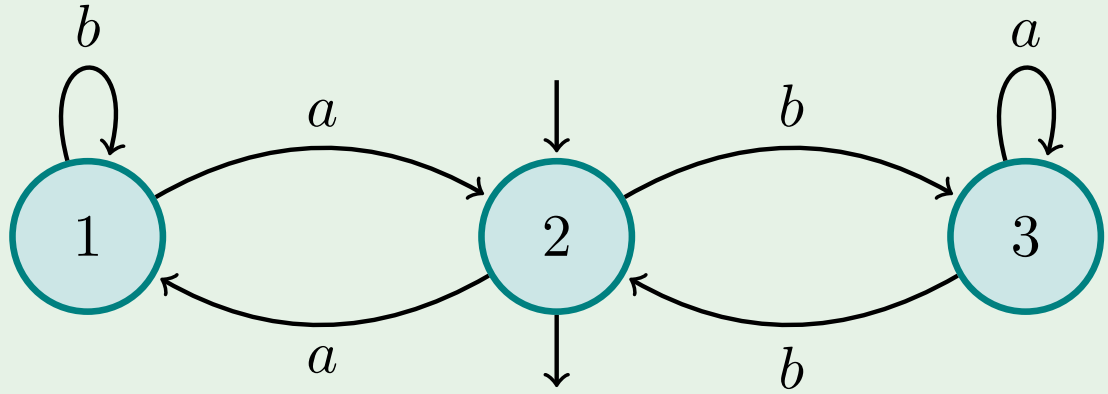
Permutation Automaton

Complete deterministic finite automaton (DFA) such that:

each letter induces a **permutation of the states**

Example 3: $(ab^*a + ba^*b)^*$ $(A = \{a, b\})$

$$
\begin{aligned}
a &: 1 \mapsto 2 \\
&\quad 2 \mapsto 1 \\
&\quad 3 \mapsto 3 \\
b &: 1 \mapsto 1 \\
&\quad 2 \mapsto 3 \\
&\quad 3 \mapsto 2
\end{aligned}
$$



Represents the class GR: **all group languages**.

Key input classes: **Group languages** (2)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.

Key input classes: **Group languages** (2)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.

We have $\mathcal{F}(\mathbb{S}_{\mathcal{G}}) = \mathcal{F}(<, \mathbb{P}_{\mathcal{G}})$.

For each $L \in \mathcal{G}$, $\mathbb{P}_{\mathcal{G}}$ contains unary predicate $P_L(x)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models P_L(i) \quad \iff \quad a_1 \cdots a_{i-1} \in L.$$

- Trivial class: $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}}) = \mathcal{F}(<)$ (linear order).

Key input classes: **Group languages** (2)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.

We have $\mathcal{F}(\mathbb{S}_{\mathcal{G}}) = \mathcal{F}(<, \mathbb{P}_{\mathcal{G}})$.

For each $L \in \mathcal{G}$, $\mathbb{P}_{\mathcal{G}}$ contains unary predicate $P_L(x)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models P_L(i) \quad \Longleftrightarrow \quad a_1 \cdots a_{i-1} \in L.$$

- Trivial class: $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}}) = \mathcal{F}(<)$ (linear order).
- Modulo languages: $\mathcal{F}(\mathbb{S}_{\mathrm{MD}}) = \mathcal{F}(<, MOD)$ (linear order + modular predicates).

Key input classes: **Group languages** (2)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.

We have $\mathcal{F}(\mathbb{S}_{\mathcal{G}}) = \mathcal{F}(<, \mathbb{P}_{\mathcal{G}})$.

For each $L \in \mathcal{G}$, $\mathbb{P}_{\mathcal{G}}$ contains unary predicate $P_L(x)$ such that,

$$a_1 a_2 a_3 \cdots a_n \models P_L(i) \quad \Longleftrightarrow \quad a_1 \cdots a_{i-1} \in L.$$

- Trivial class: $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}}) = \mathcal{F}(<)$ (linear order).
- Modulo languages: $\mathcal{F}(\mathbb{S}_{\mathrm{MD}}) = \mathcal{F}(<, MOD)$ (linear order + modular predicates).
- Alphabet modulo languages: $\mathcal{F}(\mathbb{S}_{\mathrm{AMT}}) = \mathcal{F}(<, AMOD)$ (linear order + alphabetic modular predicates).

Key input classes: **Group languages** (3)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.

Key input classes: **Group languages** (3)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.
- Let $\mathcal{G}^+$ be the **least Boolean algebra** containing $\mathcal{G}$ and $\{\varepsilon\}$.

Key input classes: **Group languages** (3)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.
- Let $\mathcal{G}^+$ be the **least Boolean algebra** containing $\mathcal{G}$ and $\{\varepsilon\}$.

We have $\mathcal{F}(\mathbb{S}_{\mathcal{G}^+}) = \mathcal{F}(<, +1, \mathbb{P}_{\mathcal{G}})$.

Adds the **successor predicate** "$+1$":
corresponds to the binary factor predicate $F_{\{\varepsilon\}}(x, y)$.

Key input classes: Group languages (3)

- Pick a **class of group languages** $\mathcal{G}$.
- Pick a **fragment of first-order logic** $\mathcal{F}$.
- Let $\mathcal{G}^+$ be the **least Boolean algebra** containing $\mathcal{G}$ and $\{\varepsilon\}$.

We have $\mathcal{F}(\mathbb{S}_{\mathcal{G}^+}) = \mathcal{F}(<, +1, \mathbb{P}_{\mathcal{G}})$.

Adds the **successor predicate "+1"**:
corresponds to the binary factor predicate $F_{\{\varepsilon\}}(x, y)$.

- Trivial class: $\mathcal{F}(\mathbb{S}_{\{\emptyset, A^*\}^+}) = \mathcal{F}(<, +1)$ (linear order + successor).
- Modulo languages: $\mathcal{F}(\mathbb{S}_{\mathrm{MD}^+}) = \mathcal{F}(<, +1, MOD)$
(linear order + successor + modular predicates).
- Alphabet modulo languages: $\mathcal{F}(\mathbb{S}_{\mathrm{AMT}^+}) = \mathcal{F}(<, +1, AMOD)$
(linear order + successor + alphabetic modular predicates).

New objective

For a fragment of first-order logic $\mathcal{F}$ (e.g. FO itself):

- **Objective**: understand the **operator** $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

New objective

For a fragment of first-order logic $\mathcal{F}$ (e.g. FO itself):

- **Objective**: understand the **operator** $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

For each fragment $\mathcal{F}$, we look at **two main questions**:

1. Reformulating $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$ using "regular expressions".
2. Connecting **membership** for $\mathcal{F}(\mathbb{S}_{\mathcal{C}})$ to properties of $\mathcal{C}$.

New objective

For a fragment of first-order logic $\mathcal{F}$ (e.g. FO itself):

- **Objective**: understand the **operator** $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$.

For each fragment $\mathcal{F}$, we look at **two main questions**:

1. Reformulating $\mathcal{C} \mapsto \mathcal{F}(\mathbb{S}_{\mathcal{C}})$ using "regular expressions".
2. Connecting **membership** for $\mathcal{F}(\mathbb{S}_{\mathcal{C}})$ to properties of $\mathcal{C}$.

In the talk, we look at several prominent fragments.

Capturing first-order logic:
Star-free closure

Star-free closure

Idea: replace **basic languages** $(\emptyset, A^*)$ with a **larger input class** $\mathcal{C}$.

- yields closure operator $\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$.

Star-free closure

Idea: replace **basic languages** $(\emptyset, A^*)$ with a **larger input class** $\mathcal{C}$.

- yields closure operator $\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$.

For an alphabet A, $\mathrm{SF}(\mathcal{C})$ is the least class of languages such that:

- contains $\mathcal{C}$.
- closed under **union** and **complement**.

$$K, L \mapsto K \cup L \qquad\qquad K \mapsto A^* \setminus K$$

- closed under **marked concatenation**:

$$\text{for a letter } a \in A \qquad K, L \mapsto KaL$$

Star-free closure

Idea: replace **basic languages** $(\emptyset, A^*)$ with a **larger input class** $\mathcal{C}$.

- yields closure operator $\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$.

For an alphabet A, $\mathrm{SF}(\mathcal{C})$ is the least class of languages such that:

- contains $\mathcal{C}$.
- closed under **union** and **complement**.

$$K, L \mapsto K \cup L \qquad\qquad K \mapsto A^* \setminus K$$

- closed under **marked concatenation**:

$$\text{for a letter } a \in A \qquad K, L \mapsto KaL$$

For each class $\mathcal{C}$, we have the correspondence $\mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.

Membership: the **general case** (P., Zeitoun'19)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
2. For all $s \in M$, if s is a **$\mathcal{C}$-stutter**, then $s^{\omega+1} = s^{\omega}$.

Membership: the **general case** (P., Zeitoun'19)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
2. For all $s \in M$, if s is a **$\mathcal{C}$-stutter**, then $s^{\omega+1} = s^{\omega}$.

$s \in M$ is a $\mathcal{C}$-stutter if and only if for every **finite** set $\mathbf{K} \subseteq \mathcal{C}$ such that $\alpha^{-1}(s) \subseteq \bigcup_{K \in \mathbf{K}}$, there exists $K \in \mathbf{K}$ such that $K \cap KK \neq \emptyset$.

Membership: the **general case** (P., Zeitoun'19)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
2. For all $s \in M$, if s is a **$\mathcal{C}$-stutter**, then $s^{\omega+1} = s^{\omega}$.

$s \in M$ is a $\mathcal{C}$-stutter if and only if for every **finite** set $\mathbf{K} \subseteq \mathcal{C}$ such that $\alpha^{-1}(s) \subseteq \bigcup_{K \in \mathbf{K}}$, there exists $K \in \mathbf{K}$ such that $K \cap KK \neq \emptyset$.

$\Rightarrow$ SF($\mathcal{C}$)-membership decidable provided that $\mathcal{C}$-stutters are computable.

Membership: the **general case** (P., Zeitoun'19)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C}) = \mathrm{FO}(\mathbb{S}_{\mathcal{C}})$.
2. For all $s \in M$, if s is a **$\mathcal{C}$-stutter**, then $s^{\omega+1} = s^{\omega}$.

$s \in M$ is a $\mathcal{C}$-stutter if and only if for every **finite** set $\mathbf{K} \subseteq \mathcal{C}$ such that $\alpha^{-1}(s) \subseteq \bigcup_{K \in \mathbf{K}}$, there exists $K \in \mathbf{K}$ such that $K \cap KK \neq \emptyset$.

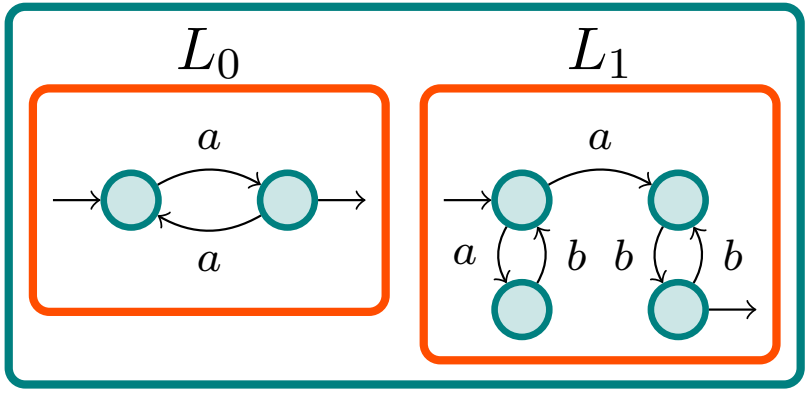
$\Rightarrow$ $\mathrm{SF}(\mathcal{C})$-membership decidable provided that $\mathcal{C}$-stutters are computable.

What is this weird condition ?!
Fortunately, the special case of **group** classes is simpler.

Towards the group case: **separation** and **pairs**

Separation problem for a class $\mathcal{C}$

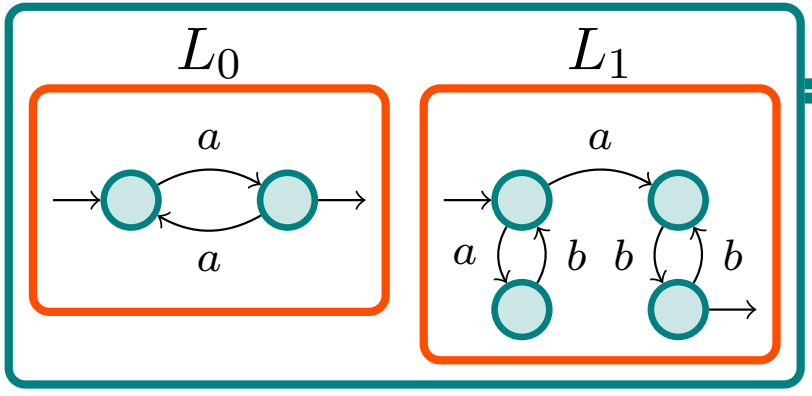
INPUT: **two** regular languages



Towards the group case: **separation** and **pairs**

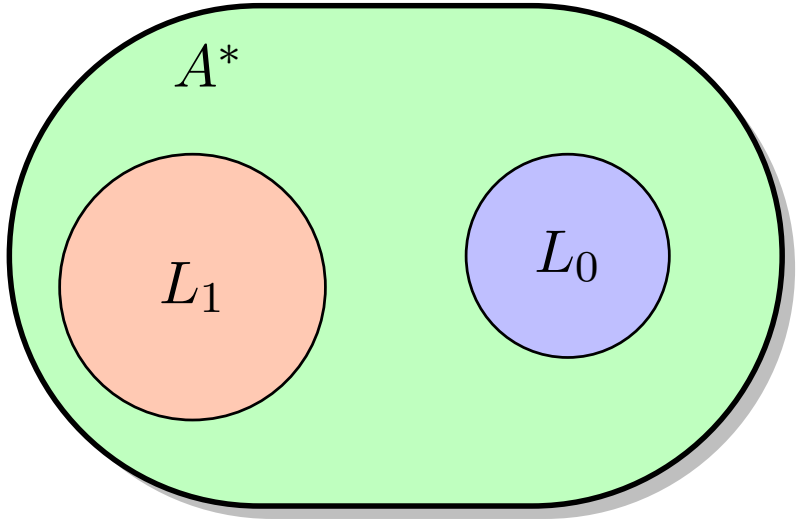
Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



QUESTION

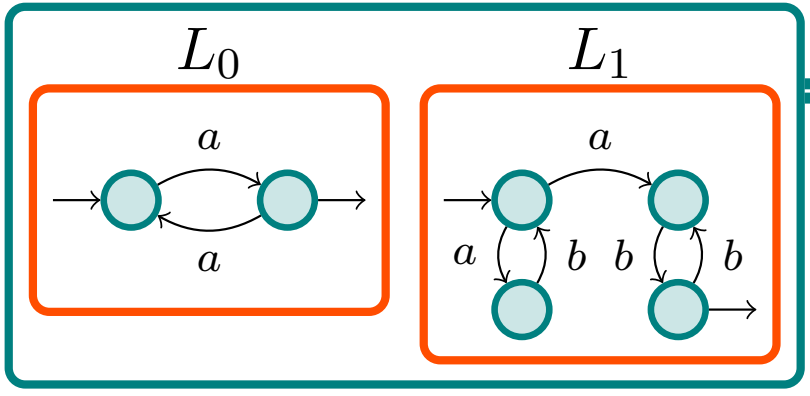
Can L_0 be separated from L_1 with a language of $\mathcal{C}$?



Towards the group case: **separation** and **pairs**

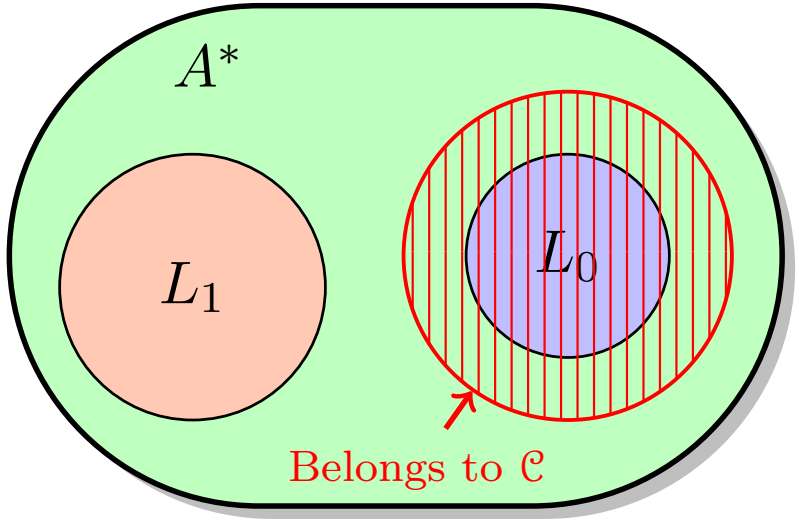
Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



QUESTION

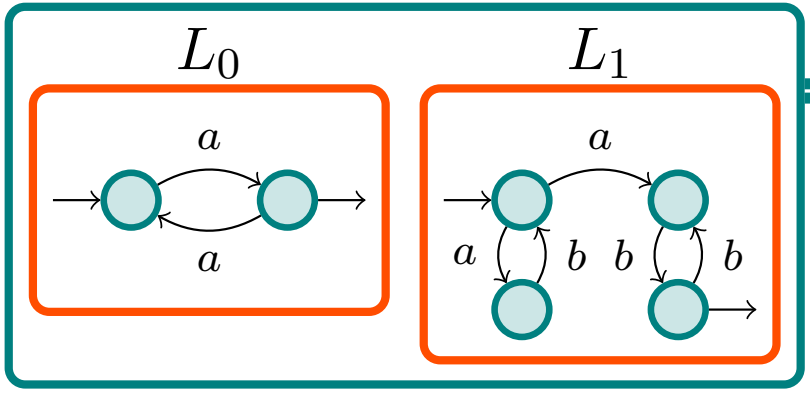
Can L_0 be separated from L_1 with a language of $\mathcal{C}$?



Towards the group case: **separation** and **pairs**

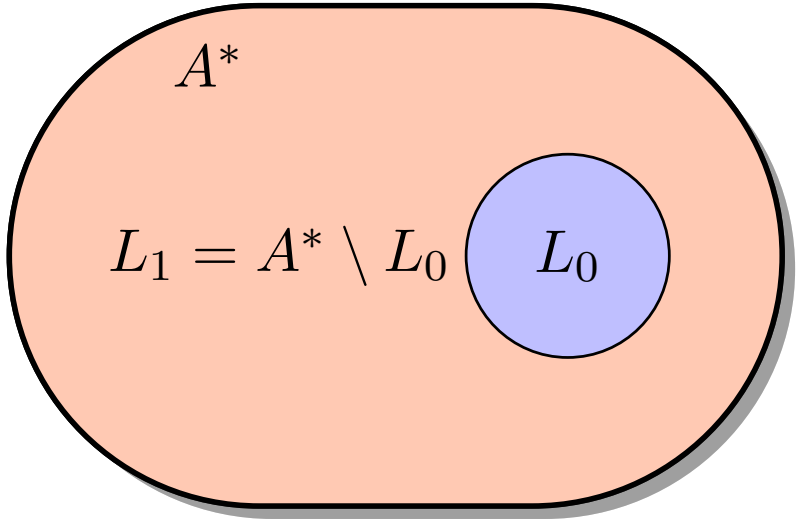
Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



QUESTION

Can L_0 be separated from L_1 with a language of $\mathcal{C}$?

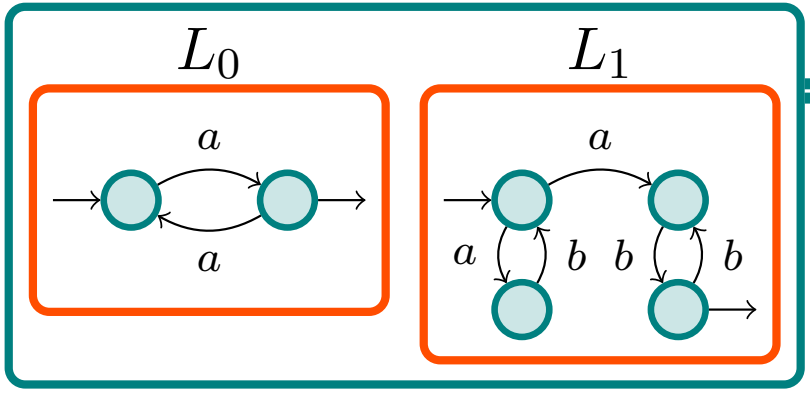


Effective **reduction** from membership to separation

Towards the group case: **separation** and **pairs**

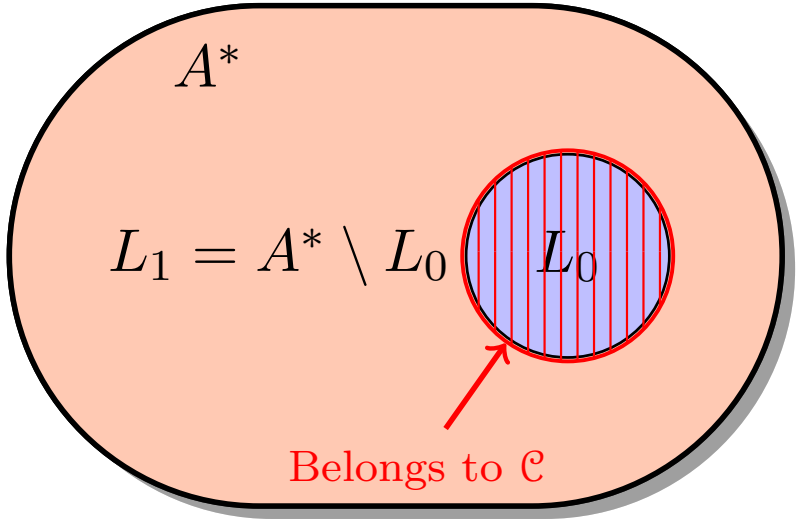
Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



QUESTION

Can L_0 be separated from L_1 with a language of $\mathcal{C}$?

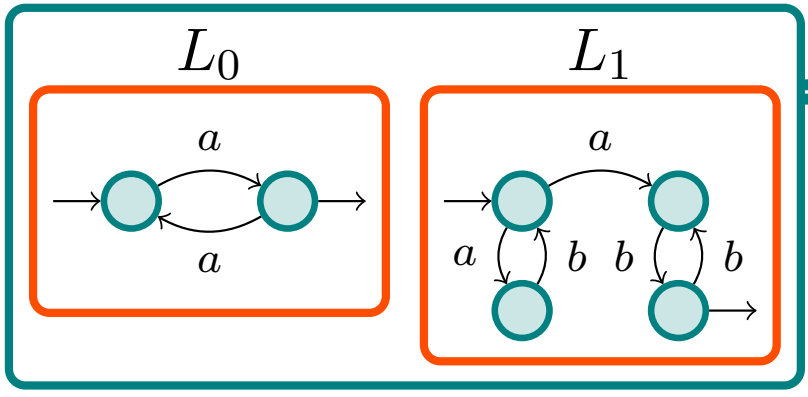


Effective **reduction** from membership to separation

Towards the group case: **separation** and **pairs**

Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



OUTPUT

Can L_0 be separated from L_1 with a language of $\mathcal{C}$?

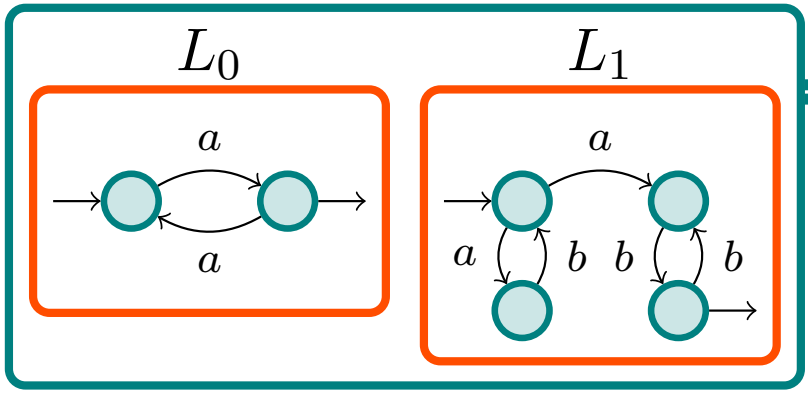
The $\mathcal{C}$-pairs associated to a morphism

Given a morphism $\alpha : A^* \to M$ a **$\mathcal{C}$-pair for α** is a pair $(s, t) \in M^2$ such that $\alpha^{-1}(s)$ is **not** $\mathcal{C}$-separable from $\alpha^{-1}(t)$.

Towards the group case: **separation** and **pairs**

Separation problem for a class $\mathcal{C}$

INPUT: **two** regular languages



OUTPUT

Can L_0 be separated from L_1 with a language of $\mathcal{C}$?

The $\mathcal{C}$-pairs associated to a morphism

Given a morphism $\alpha : A^* \to M$ a **$\mathcal{C}$-pair for α** is a pair $(s, t) \in M^2$ such that $\alpha^{-1}(s)$ is **not** $\mathcal{C}$-separable from $\alpha^{-1}(t)$.

Key point: $\mathcal{C}$-pairs can be computed when $\mathcal{C}$-separation is decidable.

Membership: the **group case** ($\mathcal{G}$ is a group class)

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{G}) = \mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$.
2. For every $s \in M$, if $(1_M, s)$ **is a $\mathcal{G}$-pair**, then $s^{\omega+1} = s^{\omega}$.

Membership: the **group case** ($\mathcal{G}$ is a group class)

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{G}) = \mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$.
2. For every $s \in M$, if $(1_M, s)$ **is a $\mathcal{G}$-pair**, then $s^{\omega+1} = s^{\omega}$.

$\Rightarrow$ $\mathrm{SF}(\mathcal{G})$-membership is decidable when $\mathcal{G}$-separation is decidable.

Membership: the **group case** ($\mathcal{G}$ is a group class)

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{G}) = \mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$.
2. For every $s \in M$, if $(1_M, s)$ **is a $\mathcal{G}$-pair**, then $s^{\omega+1} = s^{\omega}$.

$\Rightarrow$ $\mathrm{SF}(\mathcal{G})$-membership is decidable when $\mathcal{G}$-separation is decidable.

Remark 1

We do not consider $\mathcal{G}^+$ here since $\mathrm{SF}(\mathcal{G}) = \mathrm{SF}(\mathcal{G}^+)$.

Membership: the **group case** ($\mathcal{G}$ is a group class)

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{G}) = \mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$.
2. For every $s \in M$, if $(1_M, s)$ **is a $\mathcal{G}$-pair**, then $s^{\omega+1} = s^{\omega}$.

$\Rightarrow$ $\mathrm{SF}(\mathcal{G})$-membership is decidable when $\mathcal{G}$-separation is decidable.

Remark 1

We do not consider $\mathcal{G}^+$ here since $\mathrm{SF}(\mathcal{G}) = \mathrm{SF}(\mathcal{G}^+)$.

Remark 2: actually, we can do **much better**

$\mathrm{SF}(\mathcal{G})$-**separation** is decidable when $\mathcal{G}$-separation is decidable.
(but this would be a story for another talk).

The story so far for a **group class** $\mathcal{G}$

Logic	Operator
first-order logic: $\mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$	star-free closure: $\mathrm{SF}(\mathcal{G})$

Membership decidable if **$\mathcal{G}$-separation decidable**.

Let's move to other operators:

Concatenation hierarchies

Concatenation hierarchies (Brzozowski,Cohen'71)

What is a "simple" star-free language?

- $A^*aA^*bcbA^*aA^*$.
- $(ab)^* = \overline{aA^* \cup A^*aaA^* \cup A^*bbA^* \cup A^*a}$.

What is a "complicated" star-free language

- $A^*a\left(\overline{A^*ab\left(\overline{A^*a\left(\overline{A^*bbA^*}\right)acA^*}\right)cc\left(\overline{A^*bcaA^*}\right)baA^*}\right)cbA^*$.

Concatenation hierarchies (Brzozowski,Cohen'71)

What is a "simple" star-free language?

- $A^*aA^*bcbA^*aA^*$.
- $(ab)^* = \overline{aA^* \cup A^*aaA^* \cup A^*bbA^* \cup A^*a}$.

What is a "complicated" star-free language

- $A^*a\left(\overline{A^*ab\left(\overline{A^*a\left(\overline{A^*bbA^*}\right)acA^*}\right)cc\left(\overline{A^*bcaA^*}\right)baA^*}\right)cbA^*$.

Complicated = Alternation between concatenation and complement.

Intuition is mathematically validated:

- Deciding emptiness of a star-free language is **non-elementary**. (Meyer,Stockmeyer 73).
- Lower bound tied to alternations concatenation/complement.

Concatenation hierarchies (Brzozowski,Cohen'71)

For an input class $\mathcal{C}$ (**basis**), classify $\mathrm{SF}(\mathcal{C})$ into **half** and **full levels**.

Complexity measure: **alternation complement/concatenation**.

$$0 \longrightarrow \tfrac{1}{2} \longrightarrow 1 \longrightarrow \tfrac{3}{2} \longrightarrow 2 \longrightarrow \tfrac{5}{2} \longrightarrow 3 \cdots\cdots \mathrm{SF}(\mathcal{C})$$

Concatenation hierarchies (Brzozowski,Cohen'71)

For an input class $\mathcal{C}$ (**basis**), classify $\mathrm{SF}(\mathcal{C})$ into **half** and **full levels**.

Complexity measure: **alternation complement/concatenation**.

$$0 \longrightarrow \tfrac{1}{2} \longrightarrow 1 \longrightarrow \tfrac{3}{2} \longrightarrow 2 \longrightarrow \tfrac{5}{2} \longrightarrow 3 \cdots\cdots \mathrm{SF}(\mathcal{C})$$

$\mathcal{C}$

Concatenation hierarchies (Brzozowski,Cohen'71)

For an input class $\mathcal{C}$ (**basis**), classify $\mathrm{SF}(\mathcal{C})$ into **half** and **full levels**.

Complexity measure: **alternation complement/concatenation**.

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \longrightarrow 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \longrightarrow 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \longrightarrow 3 \cdots\cdots \mathrm{SF}(\mathcal{C})$$

$\mathcal{C}$

Polynomial closure

$Pol(\mathcal{D})$ finite unions of **marked products** $L_0 a_1 L_1 \cdots a_n L_n$ where $a_1, \ldots, a_n \in A$ and $L_0, \ldots, L_n \in \mathcal{D}$.

Concatenation hierarchies (Brzozowski,Cohen'71)

For an input class $\mathcal{C}$ (**basis**), classify $\mathrm{SF}(\mathcal{C})$ into **half** and **full levels**.

Complexity measure: **alternation complement/concatenation**.

$$0 \xrightarrow[\mathcal{C}]{\mathrm{Pol}} \tfrac{1}{2} \xrightarrow[\mathrm{Bool}]{} 1 \xrightarrow[\mathcal{C}]{\mathrm{Pol}} \tfrac{3}{2} \xrightarrow[\mathrm{Bool}]{} 2 \xrightarrow[\mathcal{C}]{\mathrm{Pol}} \tfrac{5}{2} \xrightarrow[\mathrm{Bool}]{} 3 \cdots\cdots \mathrm{SF}(\mathcal{C})$$

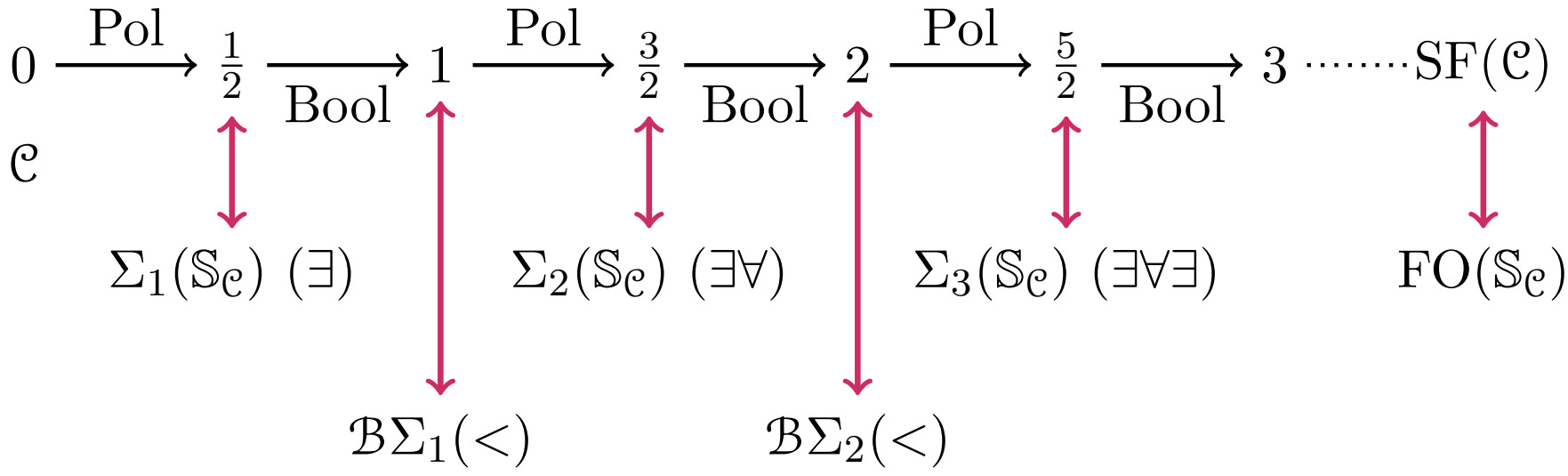
Boolean closure

$\mathit{Bool}(\mathcal{D})$: least **Boolean algebra** containing $\mathcal{D}$ (closure under union and complement).

Concatenation hierarchies (Brzozowski,Cohen'71)

For an input class $\mathcal{C}$ (**basis**), classify $\mathrm{SF}(\mathcal{C})$ into **half** and **full levels**.

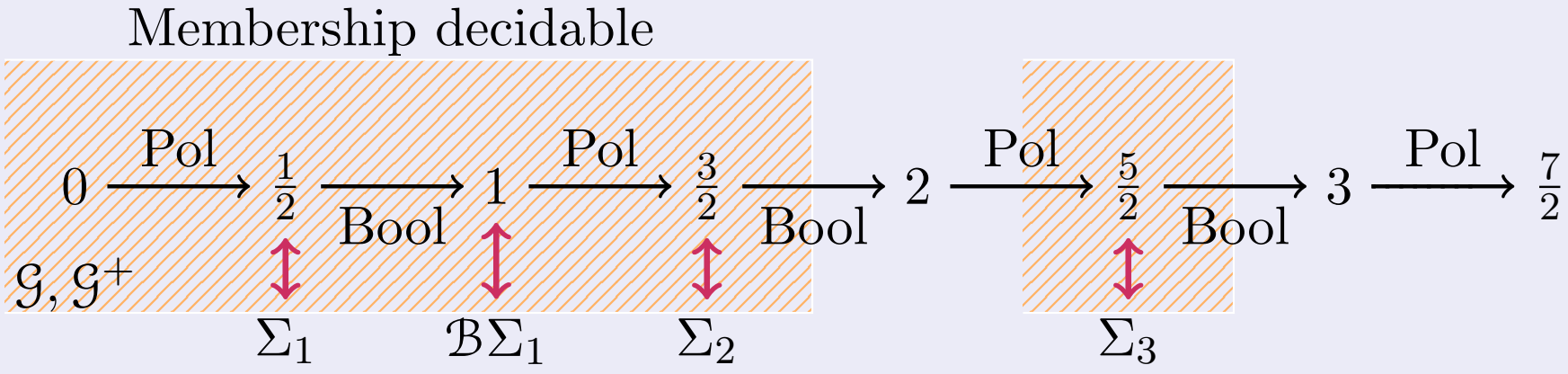
Complexity measure: **alternation complement/concatenation**.



Logical point of view: **quantifier alternation** (Thomas 1982).

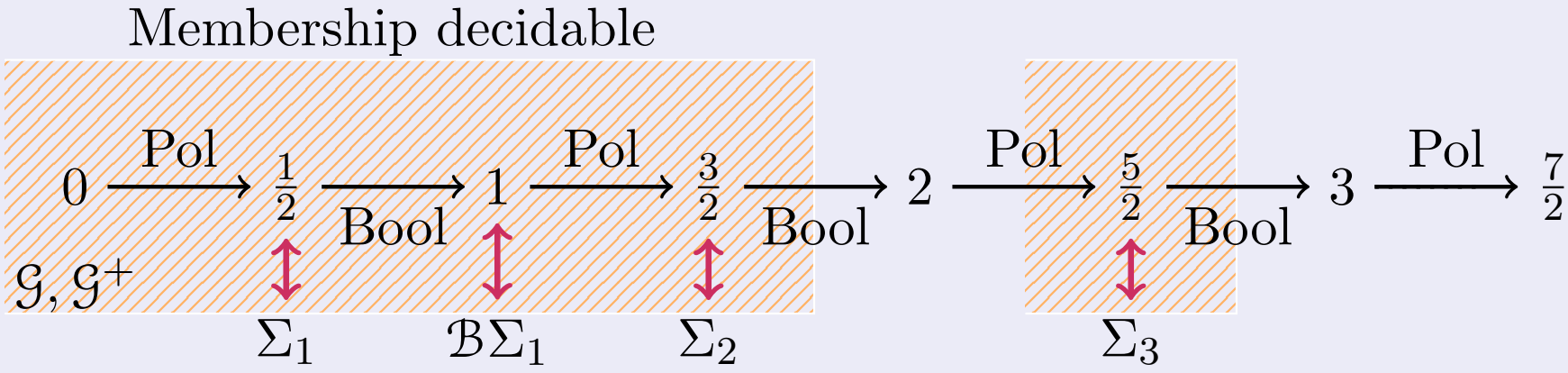
Overview of the membership results for group classes

$\mathcal{G}$ a group class **with decidable separation**. The bases $\mathcal{G}$ and $\mathcal{G}^+$:

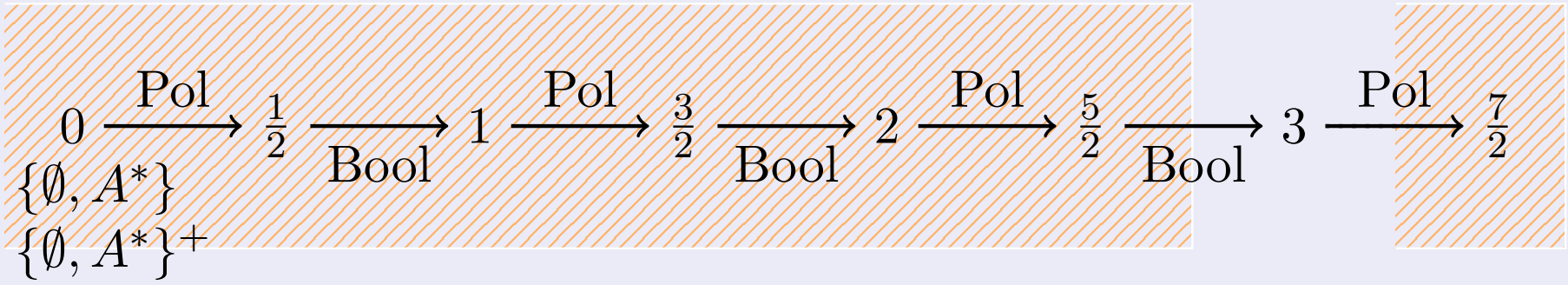


Overview of the membership results for group classes

$\mathcal{G}$ a group class **with decidable separation**. The bases $\mathcal{G}$ and $\mathcal{G}^+$:



Specialized techniques for the bases $\{\emptyset, A^*\}$ and $\{\emptyset, A^*\}^+$:



In other words, we may also handle the classes:

- $\mathcal{B}\Sigma_2(<)$, $\Sigma_4(<)$, $\mathcal{B}\Sigma_2(<, +1)$ and $\Sigma_4(<, +1)$.

Membership for the **level one** ($\mathcal{G}$ is a group class)

Let us illustrate with the class $Bool(Pol(\mathcal{G})) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
We write it $BPol(\mathcal{G})$ for short.

Membership for the **level one** ($\mathcal{G}$ is a group class)

Let us illustrate with the class $Bool(Pol(\mathcal{G})) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
We write it $BPol(\mathcal{G})$ for short.

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent:

1. $L \in BPol(\mathcal{G}) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
2. For $q, r, s, t \in M$, if (q, s) is a $\mathcal{G}$-pair, $(qr)^{\omega+1}(st)^\omega = (qr)^\omega qt(st)^\omega$.

Membership for the **level one** ($\mathcal{G}$ is a group class)

Let us illustrate with the class $Bool(Pol(\mathcal{G})) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
We write it $BPol(\mathcal{G})$ for short.

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent:

1. $L \in BPol(\mathcal{G}) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
2. For $q, r, s, t \in M$, if (q, s) is a $\mathcal{G}$-pair, $(qr)^{\omega+1}(st)^\omega = (qr)^\omega qt(st)^\omega$.

$\Rightarrow$ $BPol(\mathcal{G})$-membership is decidable when $\mathcal{G}$-separation is decidable.

Membership for the **level one** ($\mathcal{G}$ is a group class)

Let us illustrate with the class $Bool(Pol(\mathcal{G})) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
We write it $BPol(\mathcal{G})$ for short.

L a regular language and $\alpha : A^* \to M$ its syntactic morphism.
The following properties are equivalent:

1. $L \in BPol(\mathcal{G}) = \mathcal{B}\Sigma_1(<, \mathbb{P}_\mathcal{G})$.
2. For $q, r, s, t \in M$, if (q, s) is a $\mathcal{G}$-pair, $(qr)^{\omega+1}(st)^\omega = (qr)^\omega qt(st)^\omega$.

$\Rightarrow$ $BPol(\mathcal{G})$-membership is decidable when $\mathcal{G}$-separation is decidable.

Remark 1

A similar characterization holds for $BPol(\mathcal{G}^+) = \mathcal{B}\Sigma_1(<, +1, \mathbb{P}_\mathcal{G})$.

Remark 2: actually, we can do **much better**

$BPol(\mathcal{G})$-**separation** is decidable when $\mathcal{G}$-separation is decidable.
(But this would be a story for another talk).

The story so far for a **group class** $\mathcal{G}$

Logic	Operator
first-order logic: $\mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$	star-free closure: $\mathrm{SF}(\mathcal{G})$
$\mathcal{B}\Sigma_1(<, \mathbb{P}_{\mathcal{G}})$ and $\mathcal{B}\Sigma_1(<, +1, \mathbb{P}_{\mathcal{G}})$	$BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$

In **all cases**, membership decidable if **$\mathcal{G}$-separation decidable**.

A weaker operator:
Unambiguous polynomial closure

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^* \; a \; A^*$ is unambiguous.

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^*\ a\ A^*$ is unambiguous.
- If $A^*\ a\ \{\varepsilon\}\ b\ \{\varepsilon\}$ is unambiguous (it defines A^*ab).

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^*\ a\ A^*$ is unambiguous.
- If $A^*\ a\ \{\varepsilon\}\ b\ \{\varepsilon\}$ is unambiguous (it defines A^*ab).
- If $(ab)^*\ a\ (ca)^*$ is unambiguous.

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^*\ a\ A^*$ is unambiguous.
- If $A^*\ a\ \{\varepsilon\}\ b\ \{\varepsilon\}$ is unambiguous (it defines A^*ab).
- If $(ab)^*\ a\ (ca)^*$ is unambiguous.
- If $(a+c)^*\ b\ (a+b)^*\ c\ A^*\ a\ (b+c)^*$ is unambiguous.

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^*\ a\ A^*$ is unambiguous.
- If $A^*\ a\ \{\varepsilon\}\ b\ \{\varepsilon\}$ is unambiguous (it defines A^*ab).
- If $(ab)^*\ a\ (ca)^*$ is unambiguous.
- If $(a+c)^*\ b\ (a+b)^*\ c\ A^*\ a\ (b+c)^*$ is unambiguous.

Tied to the **product itself**, not only on the resulting language

- $b^*a(a+b)^*$ is unambiguous.
- $(a+b)^*a(a+b)^*$ is **not** unambiguous.

These two products evaluate to the same language.

Unambiguous polynomial closure: definition

$L_0 a_1 L_1 \cdots a_n L_n$ is **unambiguous** if every word $w \in L_0 a_1 L_1 \cdots a_n L_n$ admits a **unique** decomposition witnessing this membership:

$$\text{if } u_i, v_i \in L_i \text{ for all } i \leq n \text{ and } u_0 a_1 u_1 \cdots a_n u_n = v_0 a_1 v_1 \cdots a_n v_n,$$
$$\text{then } u_i = v_i \text{ for every } i \leq n.$$

- $b^*\ a\ A^*$ is unambiguous.
- If $A^*\ a\ \{\varepsilon\}\ b\ \{\varepsilon\}$ is unambiguous (it defines A^*ab).
- If $(ab)^*\ a\ (ca)^*$ is unambiguous.
- If $(a+c)^*\ b\ (a+b)^*\ c\ A^*\ a\ (b+c)^*$ is unambiguous.

Unambiguous polynomial closure

$UPol(\mathcal{D})$: finite **disjoint** unions of **unambiguous** marked products $L_0 a_1 L_1 \cdots a_n L_n$ where $a_1, \ldots, a_n \in A$ and $L_0, \ldots, L_n \in \mathcal{D}$.

$UPol$ defined since the 70s/80s (Schützenberger'76, Pin'81).

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.

$\Rightarrow$ $UPol(\mathcal{C})$-membership is decidable if $\mathcal{C}$-separation is decidable.

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.

$\Rightarrow$ $UPol(\mathcal{C})$-membership is decidable if $\mathcal{C}$-separation is decidable.

We can do **better**.

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.

$\Rightarrow$ $UPol(\mathcal{C})$-membership is decidable if $\mathcal{C}$-separation is decidable.

For every morphism $\alpha : A^* \to M$, we define an equivalence $\sim_{\mathcal{C}}$ on M:

$s \sim_{\mathcal{C}} t$ if and only if $s \in F \Leftrightarrow t \in F$ for all $F \subseteq M$ s.t. $\alpha^{-1}(F) \in \mathcal{C}$.

Key point: $\sim_{\mathcal{C}}$ can be computed if $\mathcal{C}$-**membership** is decidable.

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.
3. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^\omega t s^\omega$.

$\Rightarrow$ $UPol(\mathcal{C})$-membership is decidable if $\mathcal{C}$-separation is decidable.

For every morphism $\alpha : A^* \to M$, we define an equivalence $\sim_{\mathcal{C}}$ on M:

$s \sim_{\mathcal{C}} t$ if and only if $s \in F \Leftrightarrow t \in F$ for all $F \subseteq M$ s.t. $\alpha^{-1}(F) \in \mathcal{C}$.

Key point: $\sim_{\mathcal{C}}$ can be computed if $\mathcal{C}$-**membership** is decidable.

Membership for UPol (P., Zeitoun'18)

$\mathcal{C}$ a class, L a regular language, $\alpha : A^* \to M$ its syntactic morphism. The following properties are equivalent:

1. $L \in UPol(\mathcal{C})$.
2. for all $s, t \in M$, if (s, t) is a $\mathcal{C}$-pair, then $s^{\omega+1} = s^\omega t s^\omega$.
3. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^\omega t s^\omega$.

$\Rightarrow$ $UPol(\mathcal{C})$-membership is decidable if $\mathcal{C}$-**membership** is decidable.
i.e., $UPol$ **preserves the decidability of membership**.

For every morphism $\alpha : A^* \to M$, we define an equivalence $\sim_{\mathcal{C}}$ on M:

$s \sim_{\mathcal{C}} t$ if and only if $s \in F \Leftrightarrow t \in F$ for all $F \subseteq M$ s.t. $\alpha^{-1}(F) \in \mathcal{C}$.

Key point: $\sim_{\mathcal{C}}$ can be computed if $\mathcal{C}$-**membership** is decidable.

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

If K, L are group languages and $a \in A$, KaL is **not** unambiguous:

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

If K, L are group languages and $a \in A$, KaL is **not** unambiguous:

- K, L are recognized by a morphism $\alpha : A^* \to G$ into a group.
- Let $u \in K$, $v \in L$ and $x \in A^*$ such that $\alpha(x) = (\alpha(av))^{-1}$.

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

If K, L are group languages and $a \in A$, KaL is **not** unambiguous:

- K, L are recognized by a morphism $\alpha : A^* \to G$ into a group.
- Let $u \in K$, $v \in L$ and $x \in A^*$ such that $\alpha(x) = (\alpha(av))^{-1}$.
- $\alpha(uavx) = \alpha(u) \Rightarrow uavx \in K$.
- $\alpha(vxav) = \alpha(v) \Rightarrow vxav \in L$.

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

If K, L are group languages and $a \in A$, KaL is **not** unambiguous:

- K, L are recognized by a morphism $\alpha : A^* \to G$ into a group.
- Let $u \in K$, $v \in L$ and $x \in A^*$ such that $\alpha(x) = (\alpha(av))^{-1}$.
- $\alpha(uavx) = \alpha(u) \Rightarrow uavx \in K$.
- $\alpha(vxav) = \alpha(v) \Rightarrow vxav \in L$.
- Altogether, we have $uavxav \in KaL$ and **two decompositions** witness this membership.

Back to logic: what are the good input classes ?

Using **group classes** directly is **pointless**:

For every group class $\mathcal{G}$, we have $UPol(\mathcal{G}) = \mathcal{G}$.

If K, L are group languages and $a \in A$, KaL is **not** unambiguous:

- K, L are recognized by a morphism $\alpha : A^* \to G$ into a group.
- Let $u \in K$, $v \in L$ and $x \in A^*$ such that $\alpha(x) = (\alpha(av))^{-1}$.
- $\alpha(uavx) = \alpha(u) \Rightarrow uavx \in K$.
- $\alpha(vxav) = \alpha(v) \Rightarrow vxav \in L$.
- Altogether, we have $uavxav \in KaL$ and **two decompositions** witness this membership.

The **important input classes** are $BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$.

Logical characterization of UPol: **two-variable FO**

FO^2: the **two-variable** restriction of FO

At most two (reusable) variables are allowed in an FO^2 sentence:

$$\exists x \exists y \left(a(x) \wedge b(y) \wedge x < y \wedge (\exists x \left(c(x) \wedge y < x\right))\right).$$

This FO^2 sentence defines $A^* a A^* b A^* c A^*$.

Logical characterization of UPol: **two-variable FO**

FO^2: the **two-variable** restriction of FO

At most two (reusable) variables are allowed in an FO^2 sentence:

$$\exists x \exists y \left(a(x) \wedge b(y) \wedge x < y \wedge \left(\exists x \left(c(x) \wedge y < x\right)\right)\right).$$

This FO^2 sentence defines $A^* a A^* b A^* c A^*$.

For every group class $\mathcal{G}$, the following correspondences hold:

- $\mathrm{FO}^2(<, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}))$.
- $\mathrm{FO}^2(<, +1, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}^+))$.

(This was first proved by Thérien and Wilke'98 for $\mathcal{G} = \{\emptyset, A^*\}$).

Logical characterization of UPol: **two-variable FO**

FO^2: the **two-variable** restriction of FO

At most two (reusable) variables are allowed in an FO^2 sentence:

$$\exists x \exists y \left(a(x) \wedge b(y) \wedge x < y \wedge (\exists x \left(c(x) \wedge y < x\right))\right).$$

This FO^2 sentence defines $A^*aA^*bA^*cA^*$.

For every group class $\mathcal{G}$, the following correspondences hold:

- $\mathrm{FO}^2(<, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}))$.
- $\mathrm{FO}^2(<, +1, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}^+))$.

(This was first proved by Thérien and Wilke'98 for $\mathcal{G} = \{\emptyset, A^*\}$).

Warning: **specific to group classes**.
$\mathit{UPol}(\mathit{BPol}(\mathcal{C}))$ **strictly included** in $\mathrm{FO}^2(\mathbb{S}_\mathcal{C})$ in general.

Logical characterization of UPol: **two-variable FO**

FO^2: the **two-variable** restriction of FO

At most two (reusable) variables are allowed in an FO^2 sentence:

$$\exists x \exists y \left(a(x) \wedge b(y) \wedge x < y \wedge (\exists x \left(c(x) \wedge y < x\right))\right).$$

This FO^2 sentence defines $A^*aA^*bA^*cA^*$.

For every group class $\mathcal{G}$, the following correspondences hold:

- $\mathrm{FO}^2(<, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}))$.
- $\mathrm{FO}^2(<, +1, \mathbb{P}_\mathcal{G}) = \mathit{UPol}(\mathit{BPol}(\mathcal{G}^+))$.

(This was first proved by Thérien and Wilke'98 for $\mathcal{G} = \{\emptyset, A^*\}$).

Warning: **specific to group classes**.
$\mathit{UPol}(\mathit{BPol}(\mathcal{C}))$ **strictly included** in $\mathrm{FO}^2(\mathbb{S}_\mathcal{C})$ in general.

Consequence: if $\mathcal{G}$-**separation** is decidable, membership is decidable for $\mathrm{FO}^2(<, \mathbb{P}_\mathcal{G})$ and $\mathrm{FO}^2(<, +1, \mathbb{P}_\mathcal{G})$.

The story so far for a **group class** $\mathcal{G}$

Logic	Operator
first-order logic: $\mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$	star-free closure: $\mathrm{SF}(\mathcal{G})$
$\mathcal{B}\Sigma_1(<, \mathbb{P}_{\mathcal{G}})$ and $\mathcal{B}\Sigma_1(<, +1, \mathbb{P}_{\mathcal{G}})$	$BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$
two-variable first-order logic: $\mathrm{FO}^2(<, \mathbb{P}_{\mathcal{G}})$ and $\mathrm{FO}^2(<, +1, \mathbb{P}_{\mathcal{G}})$	$UPol(BPol(\mathcal{G}))$ and $UPol(BPol(\mathcal{G}^+))$

In **all cases**, membership decidable if **$\mathcal{G}$-separation decidable**.

Let's look at even weaker operators:

Deterministic polynomial closures

Deterministic marked polynomial closures

Consider a marked product $L_0 a_1 L_1 \cdots a_n L_n$. For $1 \leq i \leq n$, we write $L'_i = L_0 a_1 L_1 \cdots a_{i-1} L_{i-1}$ and $L''_i = L_i a_{i+1} \cdots L_{n-1} a_n L_n$.

$L_0 a_1 L_1 \cdots a_n L_n$ is **left deterministic** if $L'_i \cap L'_i a_i A^* = \emptyset$ for all i.

- $b^*\ a\ A^*$ is left deterministic.
- $(ab)^*\ b\ (ba)^*\ a\ A^*$ is left deterministic.

Deterministic marked polynomial closures

Consider a marked product $L_0 a_1 L_1 \cdots a_n L_n$. For $1 \leq i \leq n$, we write $L'_i = L_0 a_1 L_1 \cdots a_{i-1} L_{i-1}$ and $L''_i = L_i a_{i+1} \cdots L_{n-1} a_n L_n$.

$L_0 a_1 L_1 \cdots a_n L_n$ is **left deterministic** if $L'_i \cap L'_i a_i A^* = \emptyset$ for all i.

- $b^*\ a\ A^*$ is left deterministic.
- $(ab)^*\ b\ (ba)^*\ a\ A^*$ is left deterministic.

$L_0 a_1 L_1 \cdots a_n L_n$ is **right deterministic** if $L''_i \cap A^* a_i L''_i = \emptyset$ for all i.

- $A^*\ a\ b^*$ is right deterministic.
- $A^*\ a\ (ab)^*\ a\ (ba)^*$ is right deterministic.

Deterministic marked polynomial closures

Consider a marked product $L_0 a_1 L_1 \cdots a_n L_n$. For $1 \leq i \leq n$, we write $L'_i = L_0 a_1 L_1 \cdots a_{i-1} L_{i-1}$ and $L''_i = L_i a_{i+1} \cdots L_{n-1} a_n L_n$.

$L_0 a_1 L_1 \cdots a_n L_n$ is **left deterministic** if $L'_i \cap L'_i a_i A^* = \emptyset$ for all i.

- $b^*\ a\ A^*$ is left deterministic.
- $(ab)^*\ b\ (ba)^*\ a\ A^*$ is left deterministic.

$L_0 a_1 L_1 \cdots a_n L_n$ is **right deterministic** if $L''_i \cap A^* a_i L''_i = \emptyset$ for all i.

- $A^*\ a\ b^*$ is right deterministic.
- $A^*\ a\ (ab)^*\ a\ (ba)^*$ is right deterministic.

$L_0 a_1 L_1 \cdots a_n L_n$ is **mixed deterministic** if either $L'_i \cap L'_i a_i A^* = \emptyset$ or $L''_i \cap A^* a_i L''_i = \emptyset$ for all i.

- $a^*\ b\ A^*\ a\ b^*$ is mixed deterministic (but neither left nor right).
- $(ac)^*\ a\ (ba)^*$ is **not** mixed deterministic (but it is unambiguous).

Deterministic marked polynomial closures

Left, right and mixed polynomial closures

Three new operators $LPol(\mathcal{D})$, $RPol(\mathcal{D})$ and $MPol(\mathcal{D})$.

$MPol(\mathcal{D})$: finite **disjoint** unions of **mixed deterministic** marked products $L_0 a_1 L_1 \cdots a_n L_n$ where $L_0, \ldots, L_n \in \mathcal{D}$.

$LPol/RPol$ defined alongside $UPol$ (Schützenberger'76, Pin'81).
On the other hand, $MPol$ **is new** (P.'22).

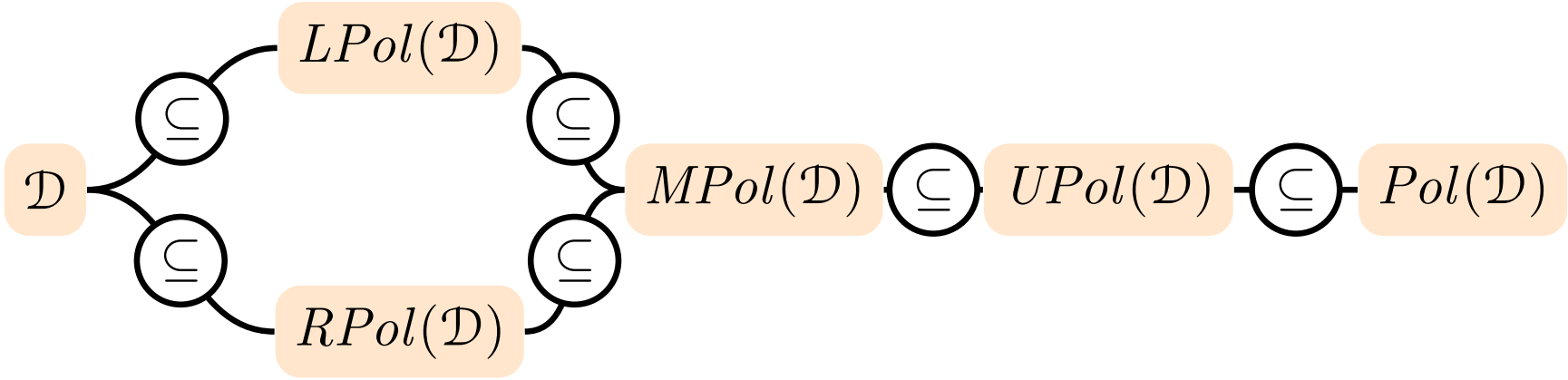
Deterministic marked polynomial closures

Left, right and mixed polynomial closures

Three new operators $LPol(\mathcal{D})$, $RPol(\mathcal{D})$ and $MPol(\mathcal{D})$.

$MPol(\mathcal{D})$: finite **disjoint** unions of **mixed deterministic** marked products $L_0 a_1 L_1 \cdots a_n L_n$ where $L_0, \ldots, L_n \in \mathcal{D}$.

$LPol/RPol$ defined alongside $UPol$ (Schützenberger'76, Pin'81).
On the other hand, $MPol$ **is new** (P.'22).



I will now convince you that $MPol$ is the **best operator ever**.

LPol, RPol, MPol: algebraic characterizations (P.'22)

$\mathcal{C}$ a class, L a language, $\alpha : A^* \to M$ its syntactic morphism.

LPol, RPol, MPol: algebraic characterizations (P.'22)

$\mathcal{C}$ a class, L a language, $\alpha : A^* \to M$ its syntactic morphism.

For $LPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in LPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^{\omega}t$.

LPol, RPol, MPol: algebraic characterizations (P.'22)

$\mathcal{C}$ a class, L a language, $\alpha : A^* \to M$ its syntactic morphism.

For $LPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in LPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^{\omega}t$.

For $RPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in RPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = ts^{\omega}$.

LPol, RPol, MPol: algebraic characterizations (P.'22)

$\mathcal{C}$ a class, L a language, $\alpha : A^* \to M$ its syntactic morphism.

For $LPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in LPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^\omega t$.

For $RPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in RPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = ts^\omega$.

For $MPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in MPol(\mathcal{C})$.
2. for all $q, r, s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $(sq)^\omega s(rs)^\omega = (sq)^\omega t(rs)^\omega$.

LPol, RPol, MPol: algebraic characterizations (P.'22)

$\mathcal{C}$ a class, L a language, $\alpha : A^* \to M$ its syntactic morphism.

For $LPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in LPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = s^\omega t$.

For $RPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in RPol(\mathcal{C})$.
2. for all $s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $s^{\omega+1} = ts^\omega$.

For $MPol(\mathcal{C})$, the following properties are equivalent:

1. $L \in MPol(\mathcal{C})$.
2. for all $q, r, s, t \in M$, if $s \sim_{\mathcal{C}} t$, then $(sq)^\omega s(rs)^\omega = (sq)^\omega t(rs)^\omega$.

$\Rightarrow LPol, RPol, MPol$ **preserve the decidability of membership**.

Logical application
of **mixed polynomial closure.**

The quantifier alternation hierarchy of FO^2

For $n \in \mathbb{N}$, $\mathcal{B}\Sigma_n^2$ denotes the **formulas** which belong to both:

- FO^2 (two-variable first-order logic).
- $\mathcal{B}\Sigma_n$ (in the quantifier hierarchy of full first-order logic).

The quantifier alternation hierarchy of FO^2

For $n \in \mathbb{N}$, $\mathcal{B}\Sigma_n^2$ denotes the **formulas** which belong to both:

- FO^2 (two-variable first-order logic).
- $\mathcal{B}\Sigma_n$ (in the quantifier hierarchy of full first-order logic).

All levels $\mathcal{B}\Sigma_n^2(<)$ (the "linear order variant"): decidable membership.
Independent proofs by Kufleitner-Weil'12 and Krebs-Straubing'12.

The quantifier alternation hierarchy of FO^2

For $n \in \mathbb{N}$, $\mathcal{B}\Sigma_n^2$ denotes the **formulas** which belong to both:

- FO^2 (two-variable first-order logic).
- $\mathcal{B}\Sigma_n$ (in the quantifier hierarchy of full first-order logic).

All levels $\mathcal{B}\Sigma_n^2(<)$ (the "linear order variant"): decidable membership.
Independent proofs by Kufleitner-Weil'12 and Krebs-Straubing'12.

I will use my own approach based on **mixed polynomial closure**.
Yet, there is a connection to the Kufleitner-Weil approach.

Deterministic hierarchies (1)

Design principle: in general $LPol(\mathcal{C})$ and $RPol(\mathcal{C})$ are **incomparable**.

Deterministic hierarchies (1)

Design principle: in general $LPol(\mathcal{C})$ and $RPol(\mathcal{C})$ are **incomparable**.

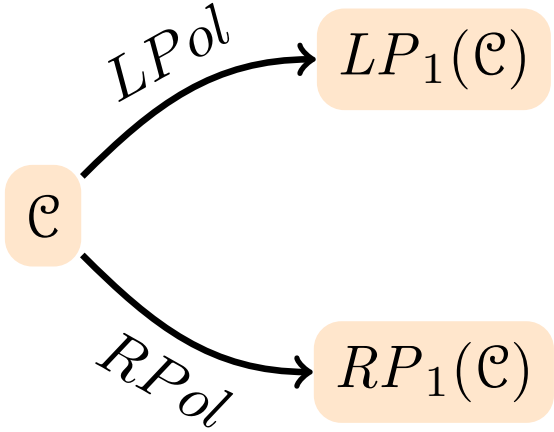
$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathcal{C}$ (the **deterministic hierarchy of basis $\mathcal{C}$**).

$\mathcal{C}$

Deterministic hierarchies (1)

Design principle: in general $LPol(\mathcal{C})$ and $RPol(\mathcal{C})$ are **incomparable**.

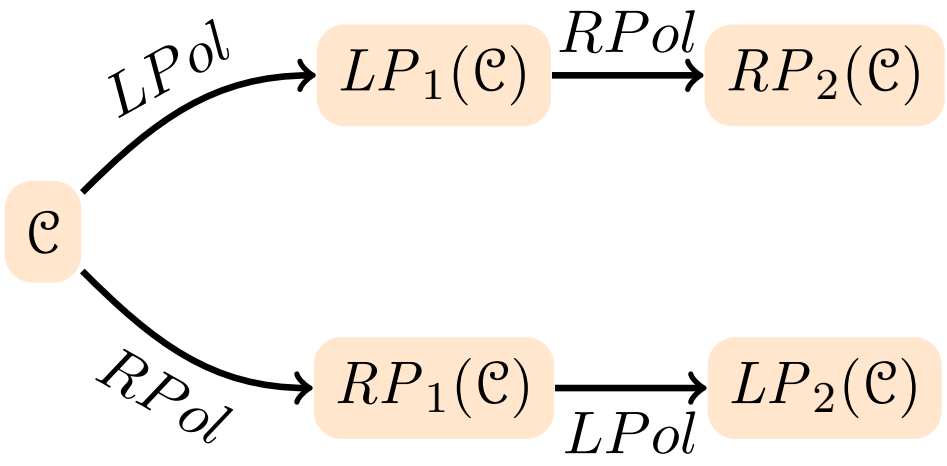
$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathcal{C}$ (the **deterministic hierarchy of basis $\mathcal{C}$**).



Deterministic hierarchies (1)

Design principle: in general $LPol(\mathfrak{C})$ and $RPol(\mathfrak{C})$ are **incomparable**.

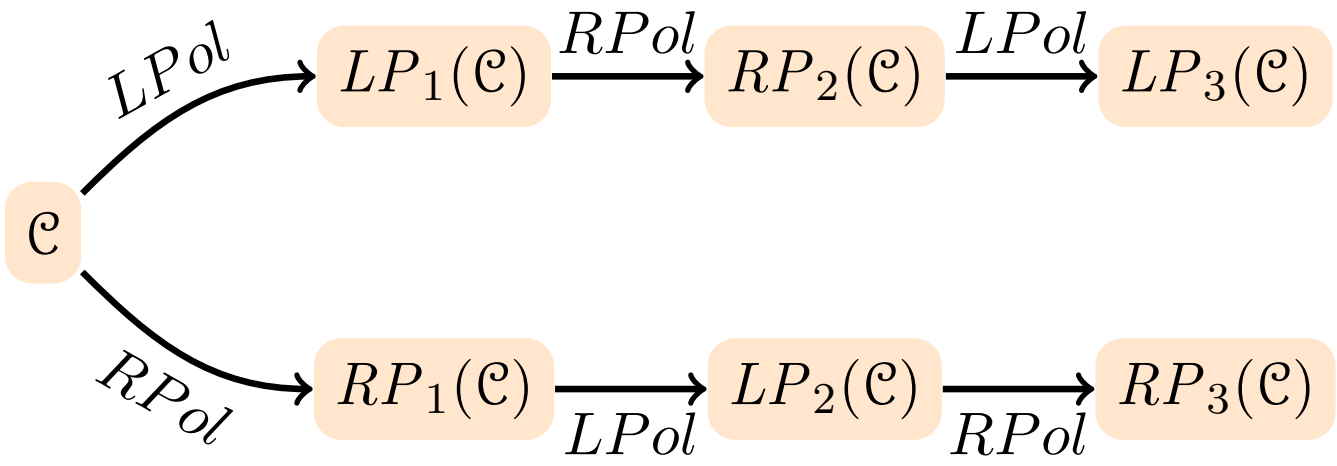
$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathfrak{C}$ (the **deterministic hierarchy of basis $\mathfrak{C}$**).



Deterministic hierarchies (1)

Design principle: in general $LPol(\mathfrak{C})$ and $RPol(\mathfrak{C})$ are **incomparable**.

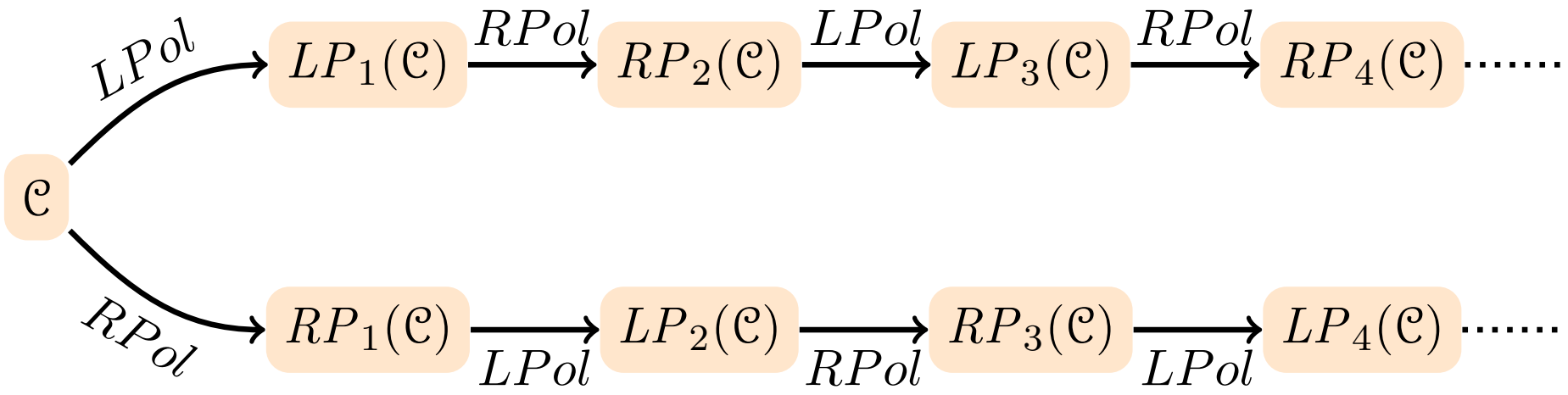
$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathfrak{C}$ (the **deterministic hierarchy of basis $\mathfrak{C}$**).



Deterministic hierarchies (1)

Design principle: in general $LPol(\mathfrak{C})$ and $RPol(\mathfrak{C})$ are **incomparable**.

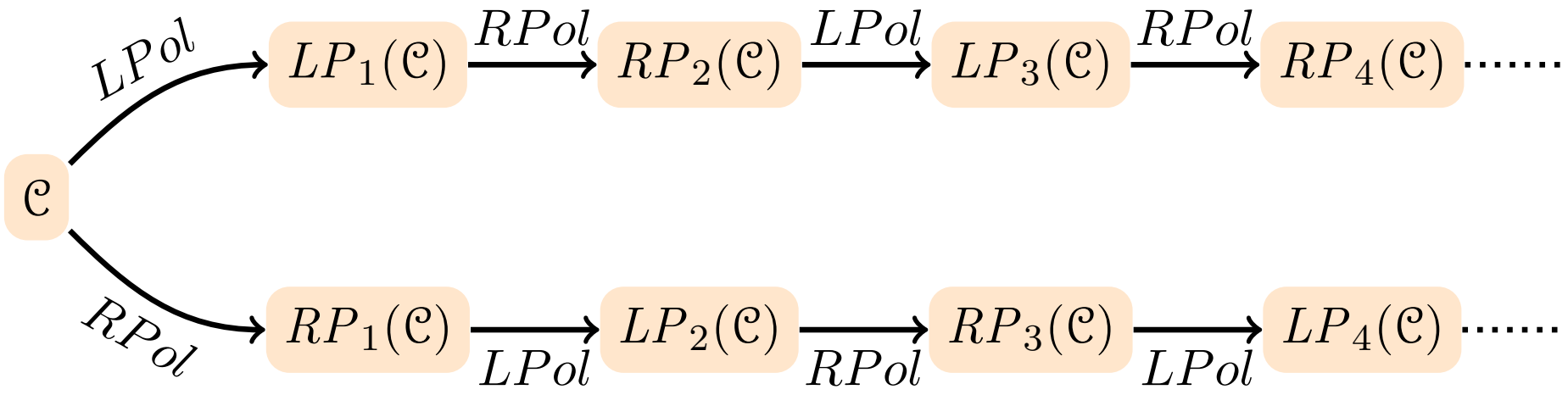
$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathfrak{C}$ (the **deterministic hierarchy of basis $\mathfrak{C}$**).



Deterministic hierarchies (1)

Design principle: in general $LPol(\mathcal{C})$ and $RPol(\mathcal{C})$ are **incomparable**.

$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathcal{C}$ (the **deterministic hierarchy of basis $\mathcal{C}$**).

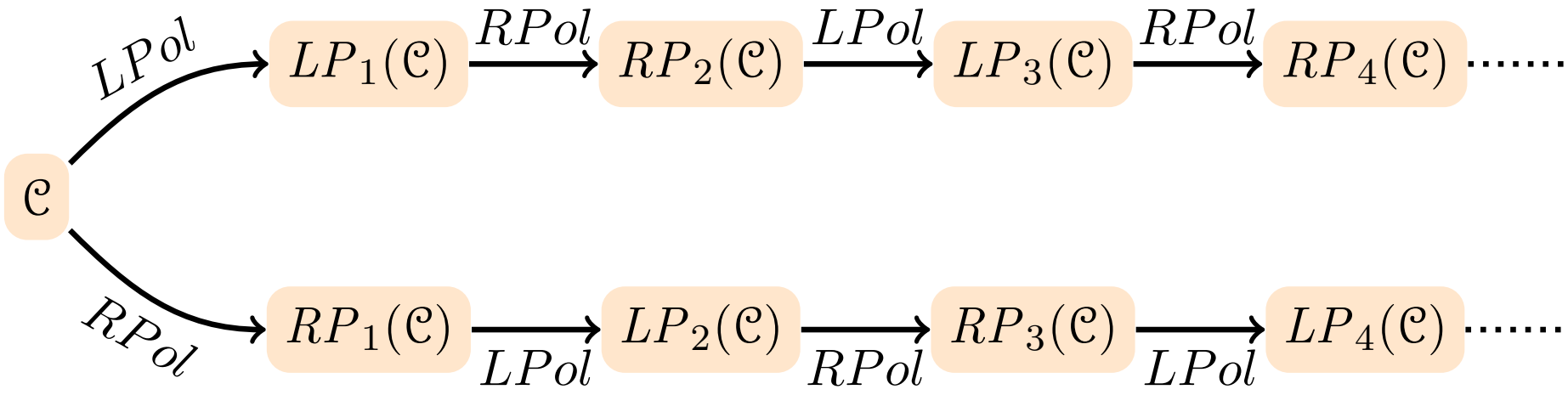


The **union of all levels** is $UPol(\mathcal{C})$ (P., Zeitoun'18).

Deterministic hierarchies (1)

Design principle: in general $LPol(\mathcal{C})$ and $RPol(\mathcal{C})$ are **incomparable**.

$\Rightarrow$ Idea: apply $LPol$ and $RPol$ alternately to build a hierarchy from $\mathcal{C}$ (the **deterministic hierarchy of basis $\mathcal{C}$**).



The **union of all levels** is $UPol(\mathcal{C})$ (P., Zeitoun'18).

If $\mathcal{C}$-membership is decidable, then the problem is also decidable for **all** levels in this hierarchy.

Deterministic hierarchies (2)

Consider the class $BPol(\{\emptyset, A^*\})$.

Deterministic hierarchies (2)

Consider the class $BPol(\{\emptyset, A^*\})$.

Theorem (Kufleitner-Weil'12)

For every level $n \geq 1$:

$$\mathcal{B}\Sigma_n^2(<) = LP_n(BPol(\{\emptyset, A^*\})) \cap RP_n(BPol(\{\emptyset, A^*\})).$$

Deterministic hierarchies (2)

Consider the class $BPol(\{\emptyset, A^*\})$.

Theorem (Kufleitner-Weil'12)

For every level $n \geq 1$:

$$\mathcal{B}\Sigma_n^2(<) = LP_n(BPol(\{\emptyset, A^*\})) \cap RP_n(BPol(\{\emptyset, A^*\})).$$

This fails in general for the logics $\mathcal{B}\Sigma_1^2(<, \mathbb{P}_\mathcal{G})$ and $\mathcal{B}\Sigma_1^2(<, +1, \mathbb{P}_\mathcal{G})$.

Let us explain why.

The approach based on **mixed polynomial closure**

Dealing with the first level (applies to **all input classes**)

$\mathcal{C}$ a class. Then $\mathcal{B}\Sigma_1(\mathbb{S}_\mathcal{C}) = \mathcal{B}\Sigma_1^2(\mathbb{S}_\mathcal{C}) = BPol(\mathcal{C})$.

Hence, if $\mathcal{G}$ group class, then,

- $\mathcal{B}\Sigma_1^2(<, \mathbb{P}_\mathcal{G}) = BPol(\mathcal{G})$.
- $\mathcal{B}\Sigma_1^2(<, +1, \mathbb{P}_\mathcal{G}) = BPol(\mathcal{G}^+)$.

The approach based on **mixed polynomial closure**

Dealing with the first level (applies to **all input classes**)

$\mathcal{C}$ a class. Then $\mathcal{B}\Sigma_1(\mathbb{S}_\mathcal{C}) = \mathcal{B}\Sigma_1^2(\mathbb{S}_\mathcal{C}) = BPol(\mathcal{C})$.

Hence, if $\mathcal{G}$ group class, then,

- $\mathcal{B}\Sigma_1^2(<, \mathbb{P}_\mathcal{G}) = BPol(\mathcal{G})$.
- $\mathcal{B}\Sigma_1^2(<, +1, \mathbb{P}_\mathcal{G}) = BPol(\mathcal{G}^+)$.

Climbing with $MPol$ (specific to the **group input classes**)

$\mathcal{G}$ a group class. Then, for every $n \geq 1$,

- $\mathcal{B}\Sigma_{n+1}^2(<, \mathbb{P}_\mathcal{G}) = MPol(\mathcal{B}\Sigma_n^2(<, \mathbb{P}_\mathcal{G}))$.
- $\mathcal{B}\Sigma_{n+1}^2(<, +1, \mathbb{P}_\mathcal{G}) = MPol(\mathcal{B}\Sigma_n^2(<, +1, \mathbb{P}_\mathcal{G}))$.

$\Rightarrow$ If $\mathcal{G}$ has decidable separation, then membership is decidable for **all** levels $\mathcal{B}\Sigma_n^2(<, \mathbb{P}_\mathcal{G})$ and $\mathcal{B}\Sigma_n^2(<, +1, \mathbb{P}_\mathcal{G})$.

Connection with the Kufleitner-Weil characterization

First result: intersection levels can be climbed with $MPol$ (P.'22)

For every class $\mathcal{C}$ and $n \geq 1$,

$$LP_{n+1}(\mathcal{C}) \cap RP_{n+1}(\mathcal{C}) = MPol(LP_n(\mathcal{C}) \cap RP_n(\mathcal{C})).$$

Connection with the Kufleitner-Weil characterization

First result: intersection levels can be climbed with $MPol$ (P.'22)

For every class $\mathcal{C}$ and $n \geq 1$,

$$LP_{n+1}(\mathcal{C}) \cap RP_{n+1}(\mathcal{C}) = MPol(LP_n(\mathcal{C}) \cap RP_n(\mathcal{C})).$$

Second result: specific to $BPol(\{\emptyset, A^*\})$ (standard)

$$LP_1(BPol(\{\emptyset, A^*\})) \cap RP_1(BPol(\{\emptyset, A^*\})) = BPol(\{\emptyset, A^*\}).$$

(fails in general for the classes $BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$).

Connection with the Kufleitner-Weil characterization

First result: intersection levels can be climbed with $MPol$ (P.'22)

For every class $\mathcal{C}$ and $n \geq 1$,

$$LP_{n+1}(\mathcal{C}) \cap RP_{n+1}(\mathcal{C}) = MPol(LP_n(\mathcal{C}) \cap RP_n(\mathcal{C})).$$

Second result: specific to $BPol(\{\emptyset, A^*\})$ (standard)

$$LP_1(BPol(\{\emptyset, A^*\})) \cap RP_1(BPol(\{\emptyset, A^*\})) = BPol(\{\emptyset, A^*\}).$$

(fails in general for the classes $BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$).

Combined with the "$MPol$ characterization" of the levels $\mathcal{B}\Sigma_n^2(<)$, this yields the Kufleitner-Weil characterization:

$$\mathcal{B}\Sigma_n^2(<) = LP_n(BPol(\{\emptyset, A^*\})) \cap RP_n(BPol(\{\emptyset, A^*\})) \quad \text{for all } n \geq 1.$$

To conclude : another fun result about $MPol$

- Reminder: $LP_n(\mathcal{C})$ and $RP_n(\mathcal{C})$ are incomparable in general.
- $\Rightarrow$ let's look at the least Boolean algebra containing both classes:

$$\text{we write it} \quad LP_n(\mathcal{C}) \vee RP_n(\mathcal{C}).$$

To conclude : another fun result about $MPol$

- Reminder: $LP_n(\mathcal{C})$ and $RP_n(\mathcal{C})$ are incomparable in general.
- $\Rightarrow$ let's look at the least Boolean algebra containing both classes:

$$\text{we write it} \quad LP_n(\mathcal{C}) \vee RP_n(\mathcal{C}).$$

Theorem: joined levels can be climbed with $MPol$ (P.'22)

For every class $\mathcal{C}$ and $n \geq 1$,

$$LP_{n+1}(\mathcal{C}) \vee RP_{n+1}(\mathcal{C}) = MPol(LP_n(\mathcal{C}) \vee RP_n(\mathcal{C})).$$

To conclude : another fun result about $MPol$

- Reminder: $LP_n(\mathcal{C})$ and $RP_n(\mathcal{C})$ are incomparable in general.
- $\Rightarrow$ let's look at the least Boolean algebra containing both classes:

$$\text{we write it} \quad LP_n(\mathcal{C}) \vee RP_n(\mathcal{C}).$$

Theorem: joined levels can be climbed with $MPol$ (P.'22)

For every class $\mathcal{C}$ and $n \geq 1$,

$$LP_{n+1}(\mathcal{C}) \vee RP_{n+1}(\mathcal{C}) = MPol(LP_n(\mathcal{C}) \vee RP_n(\mathcal{C})).$$

$\Rightarrow$ Membership is decidable for **all** levels $LP_n(\mathcal{C}) \vee RP_n(\mathcal{C})$ as soon as this is the case for the first one (*i.e.*, $LP_1(\mathcal{C}) \vee RP_1(\mathcal{C})$).

Conclusion

The story ends here today (for a **group class** $\mathcal{G}$)

Logic	Operator
first-order logic: $\mathrm{FO}(<, \mathbb{P}_{\mathcal{G}})$	star-free closure: $\mathrm{SF}(\mathcal{G})$
$\mathcal{B}\Sigma_1(<, \mathbb{P}_{\mathcal{G}})$ and $\mathcal{B}\Sigma_1(<, +1, \mathbb{P}_{\mathcal{G}})$	$BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$
two-variable first-order logic: $\mathrm{FO}^2(<, \mathbb{P}_{\mathcal{G}})$ and $\mathrm{FO}^2(<, +1, \mathbb{P}_{\mathcal{G}})$	$UPol(BPol(\mathcal{G}))$ and $UPol(BPol(\mathcal{G}^+))$
FO^2 hierarchy: all levels $\mathcal{B}\Sigma_n^2(<, \mathbb{P}_{\mathcal{G}})$ and $\mathcal{B}\Sigma_n^2(<, +1, \mathbb{P}_{\mathcal{G}})$	start from $BPol(\mathcal{G})$ and $BPol(\mathcal{G}^+)$, climb with $MPol$

In **all cases**, membership decidable if **$\mathcal{G}$-separation decidable**.

Thank you !