The separation problem in automata theory

Thomas Place
Joint work with Marc Zeitoun

LaBRI, Bordeaux University

April 22, 2024
CIBD Workshop

Classes of regular languages
and their *investigation*

Context: Classes of regular languages

- Setting: **finite words** and **regular languages** (alphabet A).
  Lots of **automata** in this talk !
- Goal: investigate **sub-classes** of the regular languages.

Context: Classes of regular languages

- Setting: **finite words** and **regular languages** (alphabet A). Lots of **automata** in this talk !
- Goal: investigate **sub-classes** of the regular languages.

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

Two main descriptive syntaxes for specifying regular languages

1. **Regular expressions** ($A^*aA^*bA^*$, $(ab)^*$, $(a(ab)^*b)^*$,...):
   Each restriction of the regular expressions yields a sub-class.

Context: Classes of regular languages

- Setting: **finite words** and **regular languages** (alphabet A). Lots of **automata** in this talk !
- Goal: investigate **sub-classes** of the regular languages.

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

Two main descriptive syntaxes for specifying regular languages

1. **Regular expressions** ($A^*aA^*bA^*$, $(ab)^*$, $(a(ab)^*b)^*$,...): Each restriction of the regular expressions yields a sub-class.
2. **Monadic second-order logic**. Büchi's theorem: MSO = REG: Each restriction of MSO yields sub-class.

Context: the historical example, **first-order logic**

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

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

$$\begin{array}{cccccccccc} a & b & b & b & c & a & a & a & \in A^* \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & \end{array}$$

- 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:

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

$$\text{defines} \quad A^* a c^* b A^*$$

Context: the historical example, **first-order logic**

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

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

$$\begin{array}{cccccccc} a & b & b & b & c & a & a & a \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \in A^*$$

- 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:

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

$$\text{defines} \quad A^* a c^* b A^*$$

Informal objective

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

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

Context: the historical example, **star-free languages**

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

Context: the 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).

Context: the 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**.

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

Context: the 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**:

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

Context: the 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**:

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

Examples of star free languages: $(A = \{a, b, c\})$.

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

Context: the 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**:

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

Examples of star free languages: $(A = \{a, b, c\})$.

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

$$\{\varepsilon\} \quad = \quad \overline{A^*aA^* \cup A^*bA^* \cup A^*cA^*}$$

Context: the 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**:

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

Examples of star free languages: $(A = \{a, b, c\})$.

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

$$\{\varepsilon\} \quad = \quad \overline{A^*aA^* \cup A^*bA^* \cup A^*cA^*}$$

$$(ab)^* \quad = \quad \overline{A^*cA^* \cup bA^* \cup A^*a\{\varepsilon\}aA^* \cup A^*b\{\varepsilon\}bA^* \cup A^*a}$$

Context: the 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**:

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

Theorem of McNaughton-Papert (1971): 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).

Context: what does it mean to "investigate" FO(<)/SF? (1)

Informal objective

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

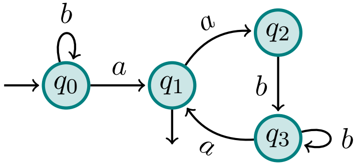
Context: what does it mean to "investigate" FO(<)/SF? (1)

Informal objective

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

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

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
($i.e.$ Does $L \in$ SF ?)

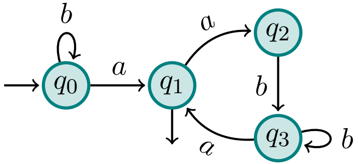
Context: what does it mean to "investigate" FO($<$)/SF? (1)

Informal objective

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

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

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
($i.e.$ Does $L \in$ SF ?)

Solution: Schützenberger (1965), McNaughton-Papert (1971)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.

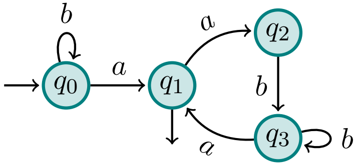
Context: what does it mean to "investigate" FO(<)/SF? (1)

Informal objective

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

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

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
($i.e.$ Does $L \in$ SF ?)

Solution: Schützenberger (1965), McNaughton-Papert (1971)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.

What are these things ? Why does this give a membership algorithm ?

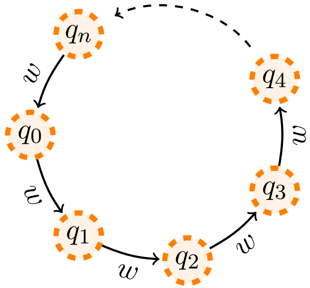
Counter inside an deterministic finite automaton

What is a counter inside an arbitrary automaton $\mathcal{A}$?

Sequence of states $q_0, \ldots, q_n$ such that,

- **Non-trivial** ($n \geq 1$).
- **Pairwise distinct** ($q_i \neq q_j$ for $i \neq j$).
- There exists a word w such that,

$$\boldsymbol{q_i \xrightarrow{w} q_{i+1} \quad \text{for } i < n \text{ and } \quad q_n \xrightarrow{w} q_0.}$$



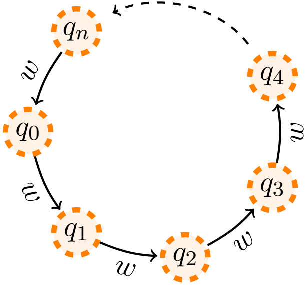
Counter inside an deterministic finite automaton

What is a counter inside an arbitrary automaton $\mathcal{A}$?

Sequence of states $q_0, \ldots, q_n$ such that,

- **Non-trivial** ($n \geq 1$).
- **Pairwise distinct** ($q_i \neq q_j$ for $i \neq j$).
- There exists a word w such that,

  $$q_i \xrightarrow{w} q_{i+1} \quad \text{for } i < n \text{ and } \quad q_n \xrightarrow{w} q_0.$$

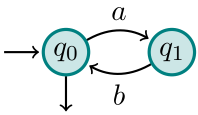


Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.
   ($i.e.$, It does **not** contain a counter)

Counter inside an deterministic finite automaton - **Examples**

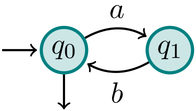
$(ab)^*$



No counter:
Star-free

Counter inside an deterministic finite automaton - **Examples**

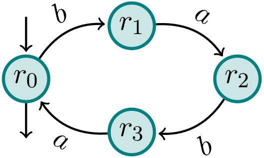
$(ab)^*$



No counter:

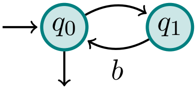
Star-free

$(baba)^*$



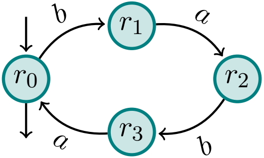
Counter inside an deterministic finite automaton - **Examples**

$(ab)^*$

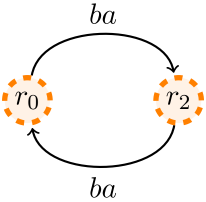


No counter:
Star-free

$(baba)^*$



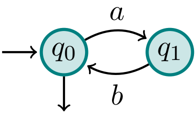
There is a counter:



NOT Star-free

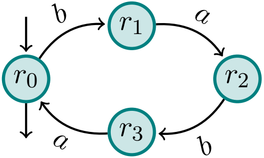
Counter inside an deterministic finite automaton - **Examples**

$\boxed{(ab)^*}$

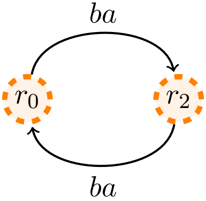


No counter:
Star-free

$\boxed{(baba)^*}$

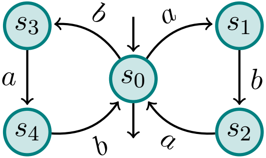


There is a counter:



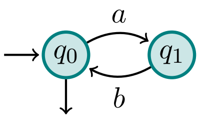
NOT Star-free

$\boxed{(aba + bab)^*}$



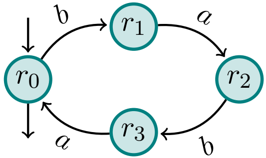
Counter inside an deterministic finite automaton - **Examples**

$(ab)^*$

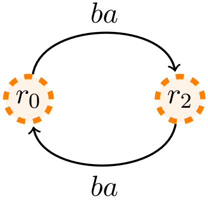


No counter:
Star-free

$(baba)^*$

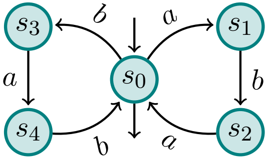


There is a counter:

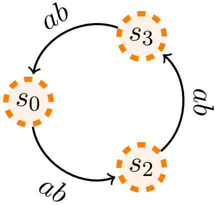


NOT Star-free

$(aba + bab)^*$



There is a counter:

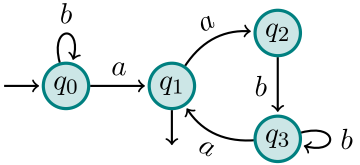


NOT Star-free

Context: The membership problem

Membership algorithm for SF = FO(<):

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
(*i.e.* Does $L \in$ SF ?)

Solution: Schützenberger (1965), McNaughton-Papert (1971)

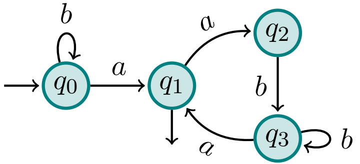
Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.

Context: The membership problem

Membership algorithm for SF = FO($<$):

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
($i.e.$ Does $L \in$ SF ?)

Solution: Schützenberger (1965), McNaughton-Papert (1971)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.

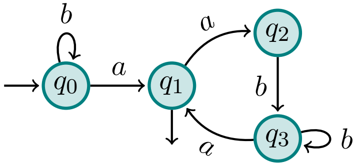
Key point: Most of the insight on SF comes **the proof** of $2 \Rightarrow 1$.

- Hypothesis: **Abstract** on a recognizer of L.

Context: The membership problem

Membership algorithm for SF $=$ FO($<$):

INPUT: A regular language L.



QUESTION:

Decide if L is **star-free**.
($i.e.$ Does $L \in$ SF ?)

Solution: Schützenberger (1965), McNaughton-Papert (1971)

Given a regular language L, the following are equivalent:

1. L is star-free.
2. The **minimal automaton** of L is **counter-free**.

Key point: Most of the insight on SF comes **the proof** of $2 \Rightarrow 1$.

- Hypothesis: **Abstract** on a recognizer of L.
- Objective: **Build a SF expression or FO sentence** for L.
- Byproduct: **Normal forms** for expressions and sentences.

What now ?

1. Look at **other significant classes** (lots of historical examples):

- Piecewise testable languages/Existential first-order logic ($\mathcal{B}\Sigma_1(<)$). (Simon'75).
- Unambiguous languages/Two-variable first-order logic ($\mathrm{FO}^2(<)$). (Schützenberger'76, Thérien-Wilke'98).
- ...

What now ?

1. Look at **other significant classes** (lots of historical examples):

- Piecewise testable languages/Existential first-order logic ($\mathcal{B}\Sigma_1(<)$). (Simon'75).
- Unambiguous languages/Two-variable first-order logic ($\mathrm{FO}^2(<)$). (Schützenberger'76, Thérien-Wilke'98).
- ...

2. Look at objects generalizing classes: **operators**.

What now ?

1. Look at **other significant classes** (lots of historical examples):
   - Piecewise testable languages/Existential first-order logic ($\mathcal{B}\Sigma_1(<)$). (Simon 1975).
   - Unambiguous languages/Two-variable first-order logic ($FO^2(<)$). (Schützenberger 1976, Thérien-Wilke 1998).
   - $\dots$
2. Look at objects generalizing classes: **operators**.

Defining families of classes:

operators

An **operator** ? What's that ?

- **Operator**: correspondence $\mathcal{C} \mapsto Op(\mathcal{C})$.
  It builds a new class $Op(\mathcal{C})$ from every input class $\mathcal{C}$.
- A single operator specifies a **family of closely related classes**.

New objective: understand **operators** rather than single classes.

An **operator** ? What's that ?

- **Operator**: correspondence $\mathcal{C} \mapsto Op(\mathcal{C})$.
  It builds a new class $Op(\mathcal{C})$ from every input class $\mathcal{C}$.
- A single operator specifies a **family of closely related classes**.

New objective: understand **operators** rather than single classes.

Why ? What are the "concrete" operators ?
Where are they coming from ?

Operators - a first motivation:
quantifier-alternation hierarchies of FO

A natural follow-up question: **quantifier alternation**

Intuition:

- **High quantifier alternation**: hard to understand.

$$\exists u \exists v\ \forall x \forall y\ \exists z \left(\begin{array}{l} \quad a(u) \wedge a(v) \wedge u < v \\ \wedge\ (u < x < z < y < v) \Rightarrow (\neg b(x) \vee \neg b(y) \vee c(z)) \end{array} \right)$$

$$\text{Defines:} \quad A^*a\overline{\left(A^*b\overline{(A^*cA^*)}bA^*\right)}aA^*.$$

Validated by theory:

- **Satisfiability** is **non-elementary** hard for FO($<$).
- Directly **tied to quantifier alternation**.

Natural idea:

- Look at membership for levels in quantifier alternation hierarchy.

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\mathbf{\Sigma_1}(<)$

$\underbrace{\exists^*}$

$\boxed{\begin{array}{c} \exists \boldsymbol{x} \exists \boldsymbol{y}\ x < y \wedge a(x) \wedge b(y) \\ A^*aA^*bA^* \end{array}}$

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\mathbf{\Sigma_1(<)} - \underbrace{\mathcal{B}\mathbf{\Sigma_1(<)}}$

$\exists^*$

$$(\exists x \exists y\ a(x) \wedge b(y)) \wedge \neg (\exists u \exists v\ u < v \wedge b(u) \wedge a(v))$$
$$aa^*bb^*$$

Boolean combinations of $\Sigma_1(<)$ sentences

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<)}$

$\exists^*$ $\qquad$ $\underbrace{\exists^*\forall^*}$

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

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\boldsymbol{\Sigma_1(<)} - \boldsymbol{\mathcal{B}\Sigma_1(<)} - \boldsymbol{\Sigma_2(<)} - \boldsymbol{\mathcal{B}\Sigma_1(<)}$

$\exists^*$ $\underbrace{\exists^*\forall^*}$

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

$A^* a c^* b A^*$

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$$\boldsymbol{\Sigma_1(<)} - \boldsymbol{\mathcal{B}\Sigma_1(<)} - \boldsymbol{\Sigma_2(<)} - \boldsymbol{\mathcal{B}\Sigma_1(<)} - \boldsymbol{\Sigma_3(<)}$$

$\exists^*$ $\qquad$ $\underbrace{\exists^*\forall^*}$ $\qquad$ $\exists^*\forall^*\exists^*$

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

$$A^*ac^*bA^*$$

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \underbrace{\Sigma_2(<)}_{} - \mathcal{B}\Sigma_1(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<)}$$

$\exists^*$ $\quad$ $\exists^*\forall^*$ $\quad$ $\exists^*\forall^*\exists^*$

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

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\boldsymbol{\Sigma_1(<)} - \boldsymbol{\mathcal{B}\Sigma_1(<)} - \underbrace{\boldsymbol{\Sigma_2(<)}}_{\exists^*\forall^*} - \boldsymbol{\mathcal{B}\Sigma_1(<)} - \boldsymbol{\Sigma_3(<)} - \boldsymbol{\mathcal{B}\Sigma_3(<)} - \boldsymbol{\Sigma_4(<)} \cdots\cdots\cdots$

$\exists^*$ $\qquad$ $\exists^*\forall^*$ $\qquad$ $\exists^*\forall^*\exists^*$ $\qquad$ $\exists^*\forall^*\exists^*\forall^*$

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

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$$\begin{array}{ccccccccccccc}
\boldsymbol{\Sigma_1(<)} & - & \boldsymbol{\mathcal{B}\Sigma_1(<)} & - & \boldsymbol{\Sigma_2(<)} & - & \boldsymbol{\mathcal{B}\Sigma_1(<)} & - & \boldsymbol{\Sigma_3(<)} & - & \boldsymbol{\mathcal{B}\Sigma_3(<)} & - & \boldsymbol{\Sigma_4(<)} \cdots\cdots\cdots \\
\exists^* & & \uparrow & & \exists^*\forall^* & & & & \exists^*\forall^*\exists^* & & & & \exists^*\forall^*\exists^*\forall^* \\
 & & \text{Simon (1975)} & & & & & & & & & &
\end{array}$$

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$$\boldsymbol{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_1(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<)} \cdots\cdots\cdots\cdots$$

$\exists^*$ $\exists^*\forall^*$ $\exists^*\forall^*\exists^*$ $\exists^*\forall^*\exists^*\forall^*$

Simon (1975)

Arfi (1987)
Pin, Weil (1995)

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_1(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<)} \cdots\cdots$$

$\exists^*$ $\quad$ $\exists^*\forall^*$ $\quad$ $\exists^*\forall^*\exists^*$ $\quad$ $\exists^*\forall^*\exists^*\forall^*$

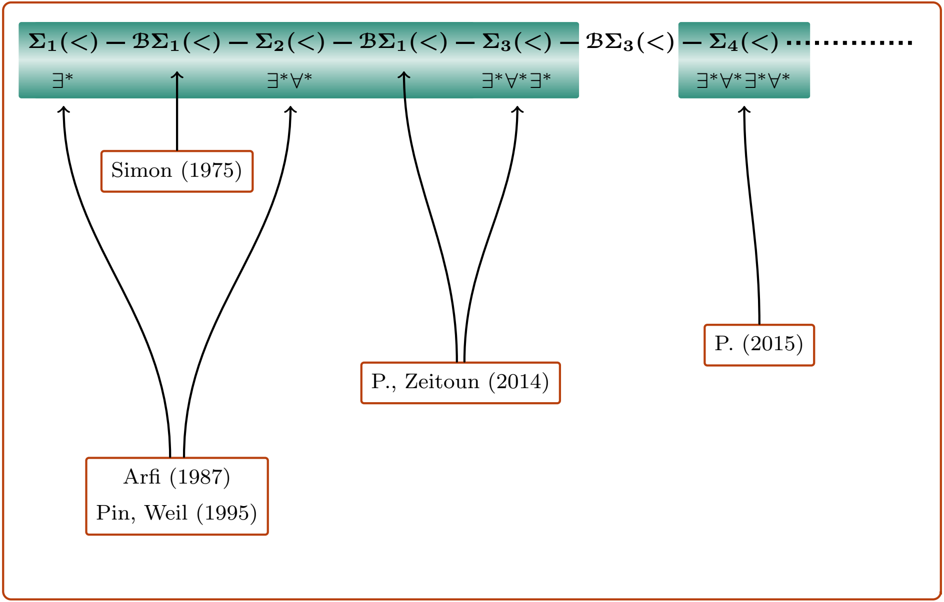
Simon (1975)

P., Zeitoun (2014)

Arfi (1987)
Pin, Weil (1995)

The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation



The quantifier alternation hierarchy of FO(<)

Idea: Classify the sentences according to quantifier alternation

$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_1(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<)} \cdots\cdots\cdots$

$\exists^*$ $\exists^*\forall^*$ $\exists^*\forall^*\exists^*$ $\exists^*\forall^*\exists^*\forall^*$

Simon (1975)

P., Zeitoun (2024)

P. (2015)

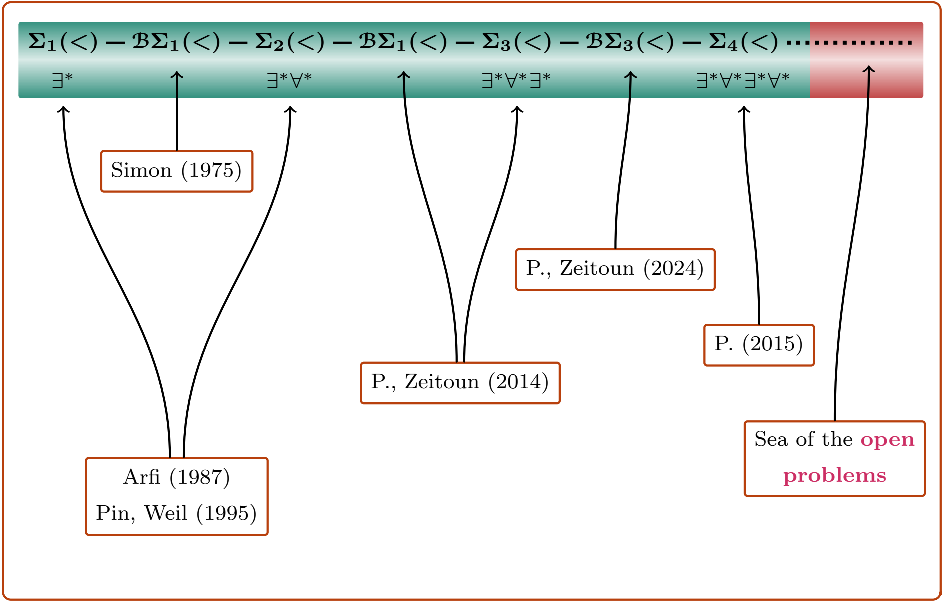
P., Zeitoun (2014)

Arfi (1987)

Pin, Weil (1995)

The **quantifier alternation hierarchy of FO(<)**

Idea: Classify the sentences according to quantifier alternation



Characterization by operators: **concatenation hierarchies**

$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<) \cdots\cdots\cdots}$

Construction process characterized by **two operators** (Thomas'82)

Characterization by operators: **concatenation hierarchies**

$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<)} \cdots\cdots\cdots$

Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

Pol($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:

  $$K, L \mapsto K \cup L.$$

- Marked concatenation:

  $$K, L, a \mapsto KaL.$$

Characterization by operators: **concatenation hierarchies**

$\mathbf{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<)} \cdots\cdots$

Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

$\text{Pol}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $$K, L \mapsto K \cup L.$$
- Marked concatenation:
  $$K, L, a \mapsto KaL.$$

Boolean closure of a class $\mathcal{C}$

$\text{Bool}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $$K, L \mapsto K \cup L.$$
- Complement:
  $$K \mapsto A^* \setminus K.$$

Characterization by operators: **concatenation hierarchies**

$\mathbf{\Sigma_1(<)} - \mathbf{\mathcal{B}\Sigma_1(<)} - \mathbf{\Sigma_2(<)} - \mathbf{\mathcal{B}\Sigma_2(<)} - \mathbf{\Sigma_3(<)} - \mathbf{\mathcal{B}\Sigma_3(<)} - \mathbf{\Sigma_4(<)} \cdots\cdots$

$\{\emptyset, A^*\}$

Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

Pol($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

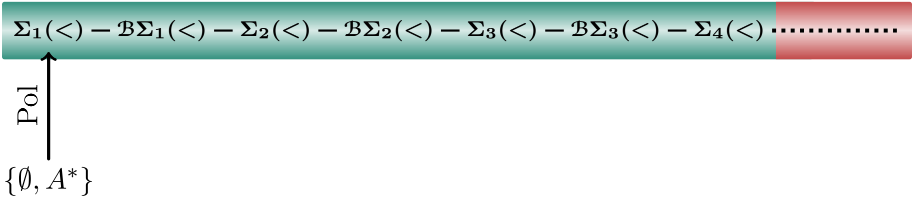
- Union:
  $K, L \mapsto K \cup L.$
- Marked concatenation:
  $K, L, a \mapsto KaL.$

Boolean closure of a class $\mathcal{C}$

Bool($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Complement:
  $K \mapsto A^* \setminus K.$

Characterization by operators: **concatenation hierarchies**



Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

Pol($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Marked concatenation:
  $K, L, a \mapsto KaL.$

Boolean closure of a class $\mathcal{C}$

Bool($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Complement:
  $K \mapsto A^* \setminus K.$

Characterization by operators: **concatenation hierarchies**

$$\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<) \cdots\cdots$$

$$\{\emptyset, A^*\} \xrightarrow{\text{Pol}} \Sigma_1(<) \xrightarrow{\text{Bool}} \mathcal{B}\Sigma_1(<)$$

Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

$\text{Pol}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

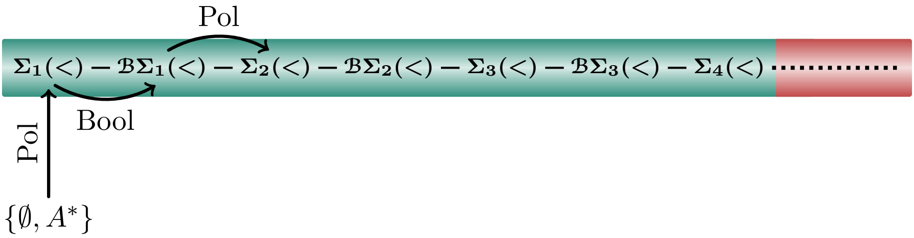
- Union:
  $$K, L \mapsto K \cup L.$$
- Marked concatenation:
  $$K, L, a \mapsto KaL.$$

Boolean closure of a class $\mathcal{C}$

$\text{Bool}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $$K, L \mapsto K \cup L.$$
- Complement:
  $$K \mapsto A^* \setminus K.$$

Characterization by operators: concatenation hierarchies



Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

$\text{Pol}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:

  $K, L \mapsto K \cup L.$

- Marked concatenation:

  $K, L, a \mapsto KaL.$

Boolean closure of a class $\mathcal{C}$

$\text{Bool}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

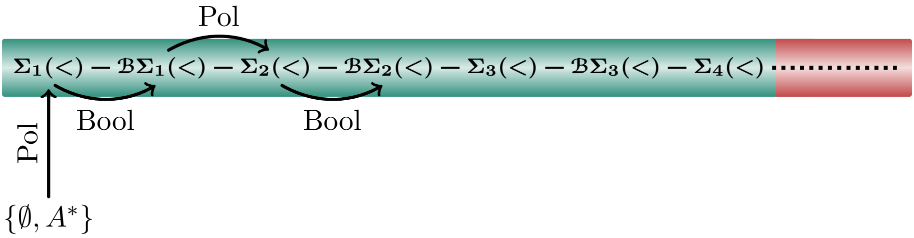
- Union:

  $K, L \mapsto K \cup L.$

- Complement:

  $K \mapsto A^* \setminus K.$

Characterization by operators: concatenation hierarchies



Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

$\text{Pol}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

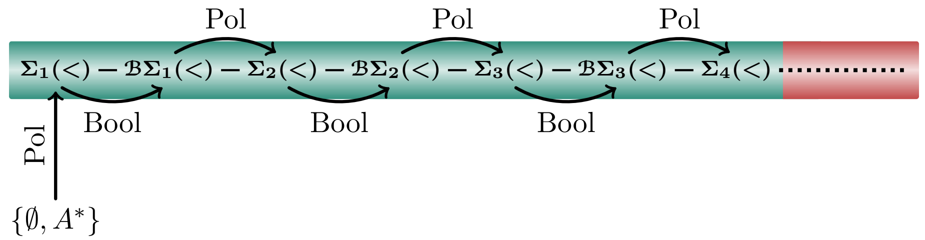
- Union:
  $$K, L \mapsto K \cup L.$$
- Marked concatenation:
  $$K, L, a \mapsto KaL.$$

Boolean closure of a class $\mathcal{C}$

$\text{Bool}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $$K, L \mapsto K \cup L.$$
- Complement:
  $$K \mapsto A^* \setminus K.$$

Characterization by operators: **concatenation hierarchies**



Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

Pol($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Marked concatenation:
  $K, L, a \mapsto KaL.$

Boolean closure of a class $\mathcal{C}$

Bool($\mathcal{C}$) is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Complement:
  $K \mapsto A^* \setminus K.$

Characterization by operators: concatenation hierarchies

$$\begin{tikzcd}
\boldsymbol{\Sigma_1(<)} \arrow[r, dash] \arrow[r, bend right, "\mathrm{Bool}"'] & \boldsymbol{\mathcal{B}\Sigma_1(<)} \arrow[r, dash] \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_2(<)} \arrow[r, dash] \arrow[r, bend right, "\mathrm{Bool}"'] & \boldsymbol{\mathcal{B}\Sigma_2(<)} \arrow[r, dash] \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_3(<)} \arrow[r, dash] \arrow[r, bend right, "\mathrm{Bool}"'] & \boldsymbol{\mathcal{B}\Sigma_3(<)} \arrow[r, dash] \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_4(<)} \cdots\cdots \\
\{\emptyset, A^*\} \arrow[u, "\mathrm{Pol}"]
\end{tikzcd}$$

Exact correspondence with
the **concatenation hierarchy of basis $\{\emptyset, A^*\}$**
(called Straubing-Thérien hierarchy (1981))

Construction process characterized by **two operators** (Thomas'82)

Polynomial closure of a class $\mathcal{C}$

$\mathrm{Pol}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Marked concatenation:
  $K, L, a \mapsto KaL.$

Boolean closure of a class $\mathcal{C}$

$\mathrm{Bool}(\mathcal{C})$ is the closure of $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L.$
- Complement:
  $K \mapsto A^* \setminus K.$

Characterization by operators: **concatenation hierarchies**

$$\{\emptyset, A^*\} \xrightarrow{\text{Pol}} \Sigma_1(<) \xrightarrow{\text{Bool}} \mathcal{B}\Sigma_1(<) \xrightarrow{\text{Pol}} \Sigma_2(<) \xrightarrow{\text{Bool}} \mathcal{B}\Sigma_2(<) \xrightarrow{\text{Pol}} \Sigma_3(<) \xrightarrow{\text{Bool}} \mathcal{B}\Sigma_3(<) \xrightarrow{\text{Pol}} \Sigma_4(<) \cdots$$

Natural next objective:

Generic analysis of the operators Pol and Bool.
(and their composition BPol)

Construc ... s'82)

Polynon ... ss $\mathcal{C}$

Pol($\mathcal{C}$) is ...

- Union:
  $K, L \mapsto K \cup L$.
- Marked concatenation:
  $K, L, a \mapsto KaL$.

... $\mathcal{C}$ under,

- Union:
  $K, L \mapsto K \cup L$.
- Complement:
  $K \mapsto A^* \setminus K$.

Operators - a second motivation:
natural variants of FO

Choosing a signature for first-order logic

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

- FO($<$): linear ordering.

Choosing a signature for first-order logic

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

- FO($<$): linear ordering.
- FO($<, \textsc{mod}$): linear order, **modular predicates**.

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

Choosing a signature for first-order logic

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

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

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

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

for $a \in A$ and $d, m \in \mathbb{N}$, unary predicate $M^a_{d,m}(x)$ expressing,
“number of a’s preceding x is congruent to d modulo m”.

Choosing a signature for first-order logic

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

- FO($<$): linear ordering.
- FO($<, \textsc{mod}$): linear order, **modular predicates**.

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

- FO($<, \textsc{amod}$): linear order, **alphabetic modular predicates**.

for $a \in A$ and $d, m \in \mathbb{N}$, unary predicate $M^a_{d,m}(x)$ expressing,
"number of a's preceding x is congruent to d modulo m".

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

- Successor pointless for FO as FO($<$) = FO($<, +1$).
- **Important for alternation hierarchies**: $\Sigma_n(<) \neq \Sigma_n(<,+1)$.

Another alternation hierarchy (successor)

$\mathbf{\Sigma_1}(<,+1)$ - $\mathcal{B}\mathbf{\Sigma_1}(<,+1)$ - $\mathbf{\Sigma_2}(<,+1)$ - $\mathcal{B}\mathbf{\Sigma_2}(<,+1)$ - $\mathbf{\Sigma_3}(<,+1)$ - $\mathcal{B}\mathbf{\Sigma_3}(<,+1)$ - $\mathbf{\Sigma_4}(<,+1)$ ·······

Another alternation hierarchy (successor)

Exact correspondence with the
concatenation hierarchy of basis $\{\emptyset, \{\varepsilon\}, A^+ A^*\}$
(the dot-depth hierarchy - Brzozowski, Cohen (1971))

$$
\begin{tikzcd}
\{\emptyset, \{\varepsilon\}, A^+, A^*\} \arrow[d, "\mathrm{Pol}"] \\
\boldsymbol{\Sigma_1}(<,+1) \arrow[r, bend right, "\mathrm{Bool}"'] & \mathcal{B}\boldsymbol{\Sigma_1}(<,+1) \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_2}(<,+1) \arrow[r, bend right, "\mathrm{Bool}"'] & \mathcal{B}\boldsymbol{\Sigma_2}(<,+1) \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_3}(<,+1) \arrow[r, bend right, "\mathrm{Bool}"'] & \mathcal{B}\boldsymbol{\Sigma_3}(<,+1) \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_4}(<,+1) \arrow[r, dash] & \cdots\cdots
\end{tikzcd}
$$

Another alternation hierarchy (successor)

$\{\emptyset, \{\varepsilon\}, A^+, A^*\}$

Pol

Exact correspondence with the
concatenation hierarchy of basis $\{\emptyset, \{\varepsilon\}, A^+A^*\}$
(the dot-depth hierarchy - Brzozowski, Cohen (1971))

Pol Pol Pol

$$\mathbf{\Sigma_1}(<,+1) \text{ - } \mathcal{B}\mathbf{\Sigma_1}(<,+1) \text{ - } \mathbf{\Sigma_2}(<,+1) \text{ - } \mathcal{B}\mathbf{\Sigma_2}(<,+1) \text{ - } \mathbf{\Sigma_3}(<,+1) \text{ - } \mathcal{B}\mathbf{\Sigma_3}(<,+1) \text{ - } \mathbf{\Sigma_4}(<,+1) \cdots\cdots$$

Bool Bool Bool

Knast (1983)

Glaßer, Schmitz (2000)

Pin, Weil (1995)

Also transfer results from
the "<" alternation hierarchy

$\mathcal{B}\Sigma$ levels: Straubing (1985)
Σ levels: Pin Weil (2002)

Yet another alternation hierarchy (MOD)

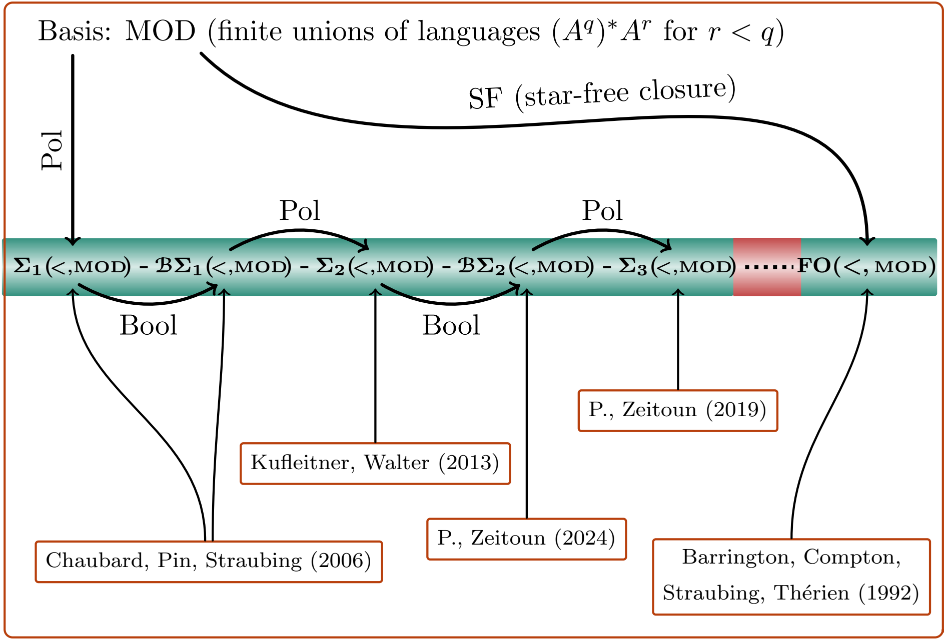
$\Sigma_1(<,\textsc{mod})$ - $\mathcal{B}\Sigma_1(<,\textsc{mod})$ - $\Sigma_2(<,\textsc{mod})$ - $\mathcal{B}\Sigma_2(<,\textsc{mod})$ - $\Sigma_3(<,\textsc{mod})$ ······ $\mathbf{FO}(<,\textsc{mod})$

Yet another alternation hierarchy (MOD)

Basis: MOD (finite unions of languages $(A^q)^* A^r$ for $r < q$)

$$\begin{tikzcd}
\mathrm{MOD} \arrow[d, "\mathrm{Pol}"'] \arrow[drrrrrr, bend left, "\text{SF (star-free closure)}"] & & & & & & \\
\boldsymbol{\Sigma_1}(<,\textsc{mod}) \arrow[r, dash] \arrow[r, bend right, "\mathrm{Bool}"'] & \mathcal{B}\boldsymbol{\Sigma_1}(<,\textsc{mod}) \arrow[r, dash] \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_2}(<,\textsc{mod}) \arrow[r, dash] \arrow[r, bend right, "\mathrm{Bool}"'] & \mathcal{B}\boldsymbol{\Sigma_2}(<,\textsc{mod}) \arrow[r, dash] \arrow[r, bend left, "\mathrm{Pol}"] & \boldsymbol{\Sigma_3}(<,\textsc{mod}) \arrow[rr, dotted, dash] & & \mathbf{FO}(<, \textsc{mod})
\end{tikzcd}$$

Yet another alternation hierarchy (MOD)



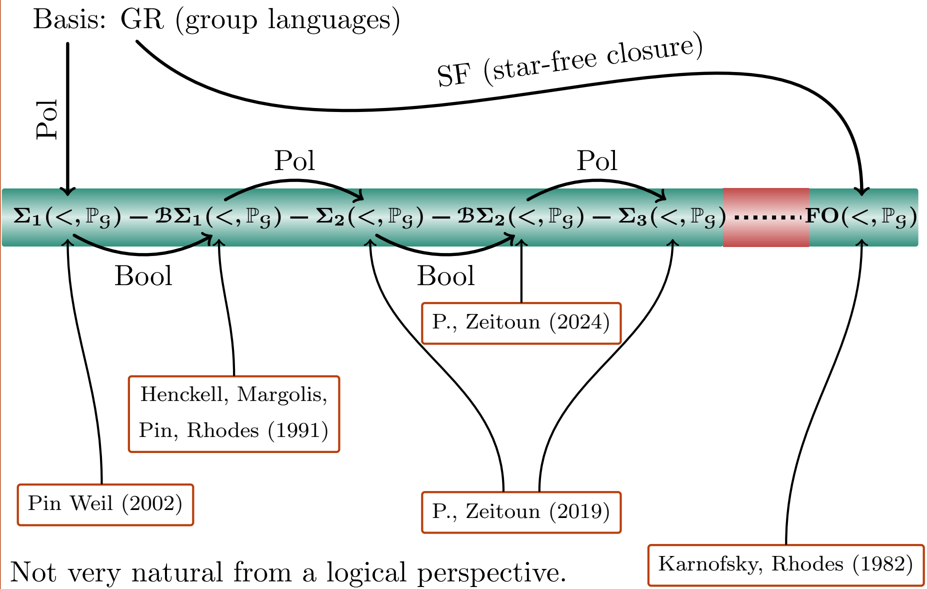
Yet again another alternation hierarchy (Groups)

$$
\begin{tikzcd}[column sep=small]
\text{Basis: GR (group languages)} \arrow[d, "\text{Pol}"'] \arrow[rrrrrrd, bend left, "\text{SF (star-free closure)}"] & & & & & & \\
\boldsymbol{\Sigma_1}(<,\mathbb{P}_{\mathcal{G}}) \arrow[r, dash] \arrow[r, bend right, "\text{Bool}"'] & \boldsymbol{\mathcal{B}\Sigma_1}(<,\mathbb{P}_{\mathcal{G}}) \arrow[r, dash] \arrow[r, bend left, "\text{Pol}"] & \boldsymbol{\Sigma_2}(<,\mathbb{P}_{\mathcal{G}}) \arrow[r, dash] \arrow[r, bend right, "\text{Bool}"'] & \boldsymbol{\mathcal{B}\Sigma_2}(<,\mathbb{P}_{\mathcal{G}}) \arrow[r, dash] \arrow[r, bend left, "\text{Pol}"] & \boldsymbol{\Sigma_3}(<,\mathbb{P}_{\mathcal{G}}) \arrow[r, dotted, dash] & \cdots \arrow[r, dotted, dash] & \mathbf{FO}(<,\mathbb{P}_{\mathcal{G}})
\end{tikzcd}
$$

Not very natural from a logical perspective.

Prominent from the concatenation hierarchy point of view.

Yet again another alternation hierarchy (Groups)



Not very natural from a logical perspective.
Prominent from the concatenation hierarchy point of view.

Yet again another alternation hierarchy (Groups)

Basis: GR (group languages)

SF (star-free closure)

Pol

$\Sigma_1(<, \mathbb{P}_\mathcal{G})$

$\mathrm{FO}(<, \mathbb{P}_\mathcal{G})$

Pin Weil

…des (1982)

Natural variants captured by operators

Generic analysis is desirable for,

- Polynomial closure: $\mathcal{C} \mapsto \mathrm{Pol}(\mathcal{C})$.
  Union: $K, L \mapsto K \cup L$
  Marked concatenation: $K, L, a \mapsto KaL$

- Boolean polynomial closure: $\mathcal{C} \mapsto \mathrm{BPol}(\mathcal{C})$.
  $\mathrm{BPol}(\mathcal{C}) = \mathrm{Bool}(\mathrm{Pol}(\mathcal{C}))$

- Star-free closure: $\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$.
  Union: $K, L \mapsto K \cup L$
  Complement: $K \mapsto A^* \setminus K$
  Marked concatenation: $K, L, a \mapsto KaL$

Not very …

Prominent from the concatenation hierarchy point of view.

Operators: what now ?

- **Operator**: correspondence $\mathcal{C} \mapsto Op(\mathcal{C})$.
  It builds a new class $Op(\mathcal{C})$ from every input class $\mathcal{C}$.
- A single operator specifies a **family of closely related classes**.

New aim: understand **operators** rather than single classes.

Operators: what now ?

- **Operator**: correspondence $\mathcal{C} \mapsto Op(\mathcal{C})$.
  It builds a new class $Op(\mathcal{C})$ from every input class $\mathcal{C}$.
- A single operator specifies a **family of closely related classes**.

New aim: understand **operators** rather than single classes.

What does this mean in the context of membership?

Previous question: tied to a single fixed class $\mathcal{D}$

Does $\mathcal{D}$ have decidable membership?

Operators: what now ?

- **Operator**: correspondence $\mathcal{C} \mapsto Op(\mathcal{C})$.
  It builds a new class $Op(\mathcal{C})$ from every input class $\mathcal{C}$.
- A single operator specifies a **family of closely related classes**.

New aim: understand **operators** rather than single classes.

What does this mean in the context of membership?

Previous question: tied to a single fixed class $\mathcal{D}$

Does $\mathcal{D}$ have decidable membership?

New question: for an operator $\mathcal{C} \mapsto Op(\mathcal{C})$ (family of classes)

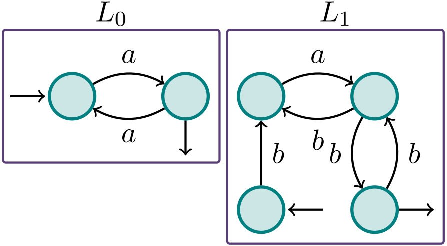
Find hypotheses on $\mathcal{C}$ ensuring that $Op(\mathcal{C})$ has decidable membership.

In many (but not all) cases, the answer is separation.

Enter the **separation problem**.

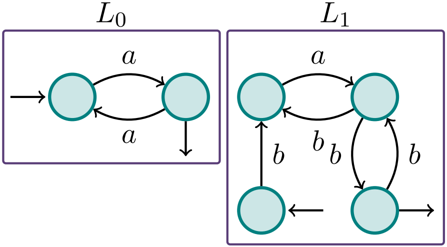
The **separation problem** for a class of languages $\mathcal{C}$

INPUT: **two** regular languages.



The **separation problem** for a class of languages $\mathcal{C}$

INPUT: **two** regular languages.



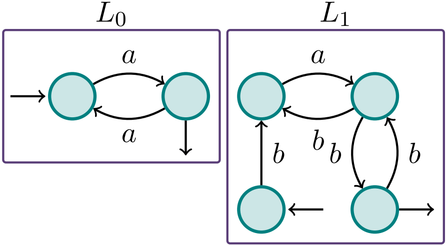
QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

We want $K \in \mathcal{C}$ such that,

- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.

The **separation problem** for a class of languages $\mathcal{C}$

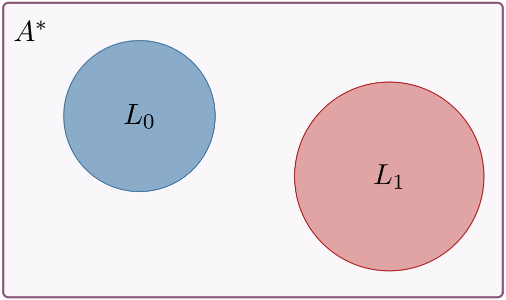
INPUT: **two** regular languages.



QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

We want $K \in \mathcal{C}$ such that,

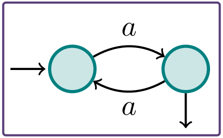
- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.



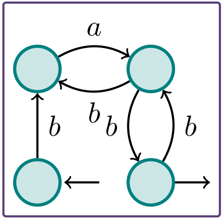
The **separation problem** for a class of languages $\mathcal{C}$

INPUT: **two** regular languages.

L_0



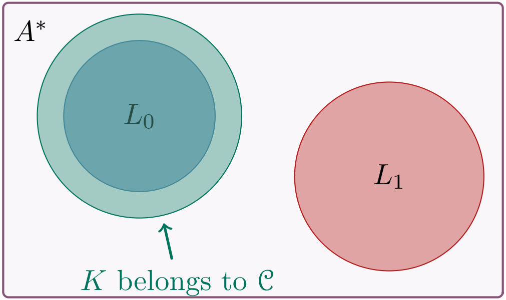
L_1



QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

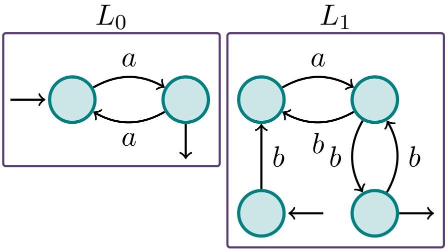
We want $K \in \mathcal{C}$ such that,

- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.



The **separation problem** for a class of languages $\mathcal{C}$

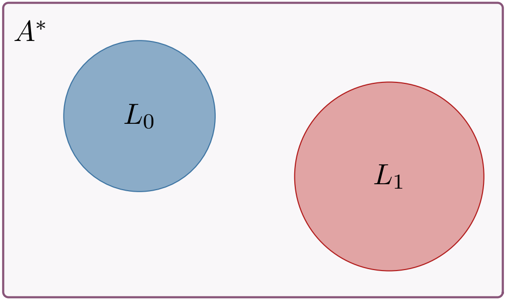
INPUT: **two** regular languages.



QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

We want $K \in \mathcal{C}$ such that,

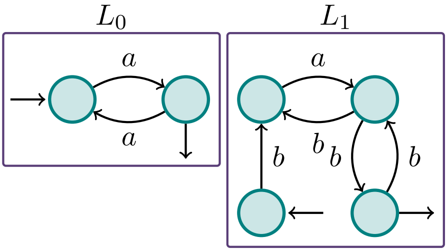
- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.



- **Reduction** from membership. (special case $L_1 = A^* \setminus L_0$)

The **separation problem** for a class of languages $\mathcal{C}$

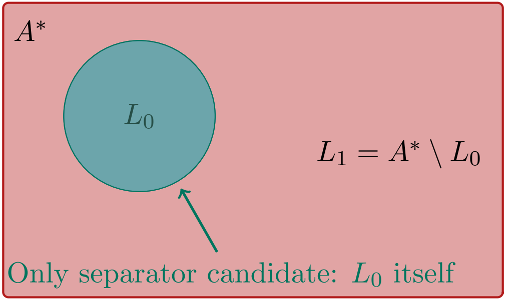
INPUT: **two** regular languages.



QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

We want $K \in \mathcal{C}$ such that,

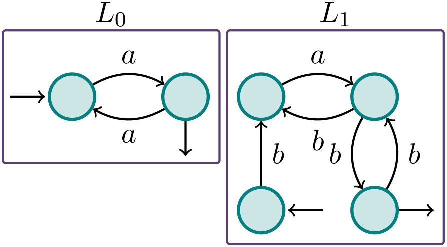
- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.



- **Reduction** from membership. (special case $L_1 = A^* \setminus L_0$)

The **separation problem** for a class of languages $\mathcal{C}$

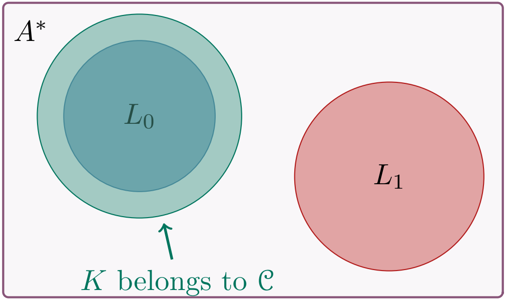
INPUT: **two** regular languages.



QUESTION: Is L_0 $\mathcal{C}$-separable from L_1?

We want $K \in \mathcal{C}$ such that,

- $L_0 \subseteq K$.
- $L_1 \cap K = \emptyset$.



- **Reduction** from membership. (special case $L_1 = A^* \setminus L_0$)

- Boils down to **interpolation**. Interpolate L_0 and $A^* \setminus L_1$: $K \in \mathcal{C}$ s.t. $L_0 \subseteq K \subseteq A^* \setminus L_1$.

The case for separation

Separation: natural generalization of membership.

- Negative aspect:
  - ☹ Usually harder than membership.
- Positive aspects:
  - ☺ **More rewarding** with respect to the insight gained on classes:
    $\mathcal{C}$-membership: detects the languages in $\mathcal{C}$.
    $\mathcal{C}$-**separation**: interaction of $\mathcal{C}$ with **all regular languages**.

The case for separation

Separation: natural generalization of membership.

- Negative aspect:
  - ☹ Usually harder than membership.
- Positive aspects:
  - ☺ **More rewarding** with respect to the insight gained on classes:
    $\mathcal{C}$-membership: detects the languages in $\mathcal{C}$.
    $\mathcal{C}$-**separation**: interaction of $\mathcal{C}$ with **all regular languages**.
  - ☺ "**transfer results**" for operators (*e.g.*, star-free closure).

Let us look at the second point.

Generic characterization
of star-free closure using **separation**

Generic characterization of star-free closure ($\mathrm{SF}(\mathcal{C})$)

Characterization for an arbitrary input $\mathcal{C}$ (with mild hypotheses)

L a regular language. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C})$.
2. $L \in \mathrm{FO}(\mathbb{I}_{\mathcal{C}})$ ($\mathbb{I}_{\mathcal{C}}$: generic signature built from $\mathcal{C}$).
3. The minimal automaton of L **does not contain a $\mathcal{C}$-counter**.

Generic characterization of star-free closure ($\mathrm{SF}(\mathcal{C})$)

Characterization for an arbitrary input $\mathcal{C}$ (with mild hypotheses)

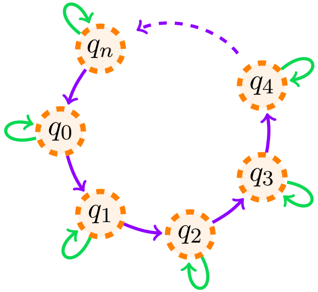
L a regular language. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C})$.
2. $L \in \mathrm{FO}(\mathbb{I}_{\mathcal{C}})$ ($\mathbb{I}_{\mathcal{C}}$: generic signature built from $\mathcal{C}$).
3. The minimal automaton of L **does not contain a $\mathcal{C}$-counter**.

What is a $\mathcal{C}$-counter inside an arbitrary DFA $\mathcal{A}$?

Sequence of states $q_0, \dots, q_n$ such that,

- **Non-trivial** ($n \geq 1$).
- **Pairwise distinct** ($q_i \neq q_j$ for $i \neq j$).
-



Generic characterization of star-free closure ($\mathrm{SF}(\mathcal{C})$)

Characterization for an arbitrary input $\mathcal{C}$ (with mild hypotheses)

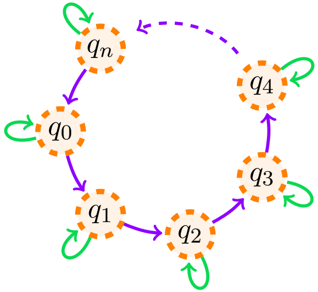
L a regular language. The following properties are equivalent.

1. $L \in \mathrm{SF}(\mathcal{C})$.
2. $L \in \mathrm{FO}(\mathbb{I}_{\mathcal{C}})$ ($\mathbb{I}_{\mathcal{C}}$: generic signature built from $\mathcal{C}$).
3. The minimal automaton of L **does not contain a $\mathcal{C}$-counter**.

What is a $\mathcal{C}$-counter inside an arbitrary DFA $\mathcal{A}$?

Sequence of states $q_0, \dots, q_n$ such that,

- **Non-trivial** ($n \geq 1$).
- **Pairwise distinct** ($q_i \neq q_j$ for $i \neq j$).
- $\boldsymbol{\bigcap_{i \leq n} L(q_i, q_i)}$ **not $\mathcal{C}$-separable from** $\boldsymbol{\bigcap_{i < n} L(q_i, q_{i+1}) \cap L(q_n, q_0)}$



$(L(q, r) = \{w \in A^* \mid q \xrightarrow{w} r\})$

Generic characterization of star-free closure ($SF(\mathcal{C})$)

Characterization for an arbitrary input $\mathcal{C}$ (with mild hypotheses)

L a regular language. The following properties are equivalent.

1. $L \in SF(\mathcal{C})$.
2. $L \in FO(\mathbb{I}_{\mathcal{C}})$ ($\mathbb{I}_{\mathcal{C}}$: generic signature built from $\mathcal{C}$).
3. The minimal automaton of L **does not contain a $\mathcal{C}$-counter**.

Consequence

Membership for $SF(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

The proof of $3 \Rightarrow 1$ is particularly interesting:

- If there is no $\mathcal{C}$-counter, build a "$SF(\mathcal{C})$ expression" for L.
- First step: **choose the basic languages in $\mathcal{C}$**.
- These are the languages in $\mathcal{C}$ that separate languages of the form,

$$\bigcap_{(q,r) \in P} L(q,r) \quad P \text{ a set of pairs of states.}$$

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto SF(\mathcal{C})$).

  Membership for $SF(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto SF(\mathcal{C})$).

  Membership for $SF(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).

- **Polynomial closure** ($\mathcal{C} \mapsto Pol(\mathcal{C})$).

  Membership for $Pol(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2014) - Can also be deduced from Pin, Weil (1997).

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto SF(\mathcal{C})$).

  Membership for $SF(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).
- **Polynomial closure** ($\mathcal{C} \mapsto Pol(\mathcal{C})$).

  Membership for $Pol(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2014) - Can also be deduced from Pin, Weil (1997).
- **Boolean polynomial closure** ($\mathcal{C} \mapsto BPol(\mathcal{C})$).

  Membership for $BPol(\mathcal{C})$ boils down to **covering** for $\mathcal{C}$.

  P., Zeitoun (2024)

  Covering: generalizes separation to **more than two inputs**.

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$).

  Membership for $\mathrm{SF}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).
- **Polynomial closure** ($\mathcal{C} \mapsto \mathrm{Pol}(\mathcal{C})$).

  Membership for $\mathrm{Pol}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2014) - Can also be deduced from Pin, Weil (1997).
- **Boolean polynomial closure** ($\mathcal{C} \mapsto \mathrm{BPol}(\mathcal{C})$).

  Membership for $\mathrm{BPol}(\mathcal{C})$ boils down to **covering** for $\mathcal{C}$.

  P., Zeitoun (2024)

  Covering: generalizes separation to **more than two inputs**.

$$\boldsymbol{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<) - \Sigma_4(<) \cdots\cdots\cdots}$$

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$).

  Membership for $\mathrm{SF}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).
- **Polynomial closure** ($\mathcal{C} \mapsto \mathrm{Pol}(\mathcal{C})$).

  Membership for $\mathrm{Pol}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2014) - Can also be deduced from Pin, Weil (1997).
- **Boolean polynomial closure** ($\mathcal{C} \mapsto \mathrm{BPol}(\mathcal{C})$).

  Membership for $\mathrm{BPol}(\mathcal{C})$ boils down to **covering** for $\mathcal{C}$.

  P., Zeitoun (2024)

  Covering: generalizes separation to **more than two inputs**.

$$\underbrace{\boldsymbol{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<)}}_{\text{Separation and covering decidable}} - \underbrace{\boldsymbol{\mathcal{B}\Sigma_3(<) - \Sigma_4(<)}}_{\text{Membership decidable}} \cdots\cdots\cdots$$

The known transfer results

- **Star-free closure** ($\mathcal{C} \mapsto \mathrm{SF}(\mathcal{C})$).

  Membership for $\mathrm{SF}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2019) - Can also be deduced from Straubing (1979).
- **Polynomial closure** ($\mathcal{C} \mapsto \mathrm{Pol}(\mathcal{C})$).

  Membership for $\mathrm{Pol}(\mathcal{C})$ boils down to **separation** for $\mathcal{C}$.

  P., Zeitoun (2014) - Can also be deduced from Pin, Weil (1997).
- **Boolean polynomial closure** ($\mathcal{C} \mapsto \mathrm{BPol}(\mathcal{C})$).

  Membership for $\mathrm{BPol}(\mathcal{C})$ boils down to **covering** for $\mathcal{C}$.

  P., Zeitoun (2024)

  Covering: generalizes separation to **more than two inputs**.

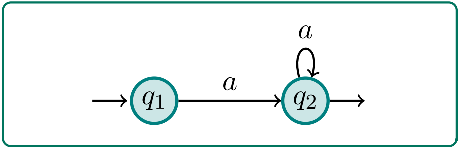
Transfer theorems for Pol and BPol

$$\underbrace{\boldsymbol{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<)}}_{\text{Separation and covering decidable}} - \underbrace{\boldsymbol{\mathcal{B}\Sigma_3(<) - \Sigma_4(<)}}_{\text{Membership decidable}} \cdots\cdots\cdots$$

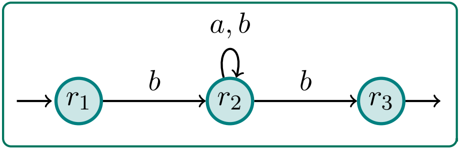
So, how do we deal with this separation stuff ?

Tackling the **separation problem**

The typical separation procedure for class $\mathcal{C}$



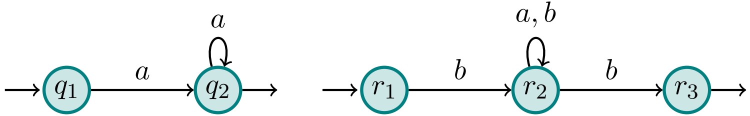
Language L_0



Language L_1

We want to **test $\mathcal{C}$-separability** for these two inputs.

The typical separation procedure for class $\mathcal{C}$



Language L_0

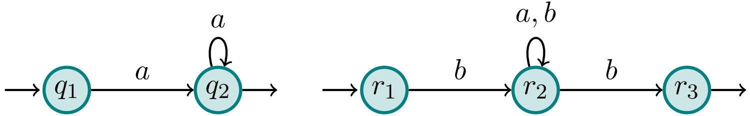
Language L_1

We want to **test $\mathcal{C}$-separability** for these two inputs.

Preliminary step:

View the **two automata as a single one** $\mathcal{A} = (Q, \delta)$.

The typical separation procedure for class $\mathcal{C}$



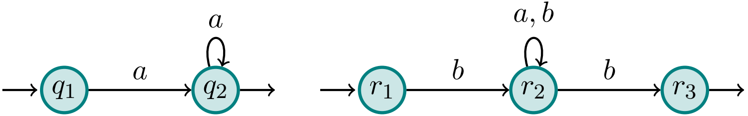
Language L_0 Language L_1

We want to **test $\mathcal{C}$-separability** for these two inputs.

Preliminary step:
View the **two automata as a single one** $\mathcal{A} = (Q, \delta)$.

Main procedure (specific to $\mathcal{C}$):
Compute the set $\mathcal{I}_{\mathcal{C}}[\mathcal{A}] \subseteq Q^4$ of **non-separable quadruples.**
all $(q, r, s, t) \in Q^4$ such that $L(q, r)$ not $\mathcal{C}$-separable from $L(s, t)$

The typical separation procedure for class $\mathcal{C}$



Language L_0 Language L_1

We want to **test $\mathcal{C}$-separability** for these two inputs.

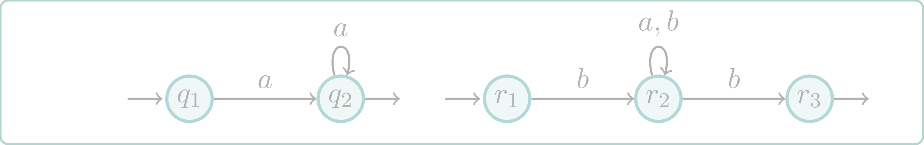
Preliminary step:
View the **two automata as a single one** $\mathcal{A} = (Q, \delta)$.

Main procedure (specific to $\mathcal{C}$):
Compute the set $\mathcal{I}_{\mathcal{C}}[\mathcal{A}] \subseteq Q^4$ of **non-separable quadruples.**
all $(q, r, s, t) \in Q^4$ such that $L(q, r)$ not $\mathcal{C}$-separable from $L(s, t)$

L_0 is **not** $\mathcal{C}$-separable from L_1 iff there exists $(q, r, s, t) \in \mathcal{I}_{\mathcal{C}}[\mathcal{A}]$ s.t.,

- q initial for L_0 and r final for L_0.
- s initial for L_1 and t final for L_1.

The typical separation procedure for class $\mathcal{C}$



Language L_0 Language L_1

Prel…

View…

Typically: main procedure based on a least fixpoint.

Main procedure (specific to $\mathcal{C}$):

Compute the set $\mathcal{I}_{\mathcal{C}}[\mathcal{A}] \subseteq Q^4$ of **non-separable quadruples.**

all $(q, r, s, t) \in Q^4$ such that $L(q, r)$ not $\mathcal{C}$-separable from $L(s, t)$

L_0 is **not** $\mathcal{C}$-separable from L_1 iff there exists $(q, r, s, t) \in \mathcal{I}_{\mathcal{C}}[\mathcal{A}]$ s.t.,

- q initial for L_0 and r final for L_0.
- s initial for L_1 and t final for L_1.

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Generic to all classes $\mathcal{C}$

Intersecting languages are not $\mathcal{C}$-separable

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q,\boldsymbol{r},s,\boldsymbol{t}),(\boldsymbol{r},u,\boldsymbol{t},v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Generic to all practical* classes $\mathcal{C}$

If L_0 is not $\mathcal{C}$-separable from L_1 and H_0 is not $\mathcal{C}$-separable from H_1, then L_0H_0 is not $\mathcal{C}$-separable from L_1H_1.

* mild hypotheses on $\mathcal{C}$ are needed.

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q,\boldsymbol{r},s,\boldsymbol{t}),(\boldsymbol{r},u,\boldsymbol{t},v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(3) If $q,r,s \in Q$ and $L(r,s) \neq \emptyset$, then $(q,q,r,s) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Specific to the particular class $\Sigma_1(<)$

A^* is the only language in $\Sigma_1(<)$ containing the empty word $\varepsilon \in L(q,q)$.

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

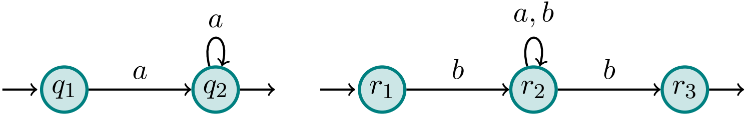
One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q,\boldsymbol{r},s,\boldsymbol{t}),(\boldsymbol{r},u,\boldsymbol{t},v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(3) If $q,r,s \in Q$ and $L(r,s) \neq \emptyset$, then $(q,q,r,s) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Let us look at the example



Language L_0 Language L_1

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

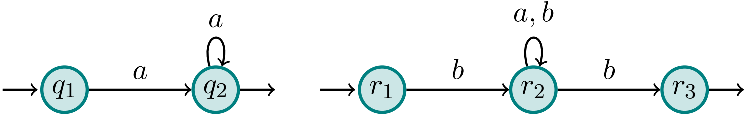
One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q, \boldsymbol{r}, s, \boldsymbol{t}), (\boldsymbol{r}, u, \boldsymbol{t}, v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(3) If $q, r, s \in Q$ and $L(r,s) \neq \emptyset$, then $(q,q,r,s) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Let us look at the example



Language L_0 Language L_1

- By **(1)**, $(q_1, q_2, r_2, r_2) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$ since $a \in L(q_1,q_2) \cap L(r_1,r_2)$.

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

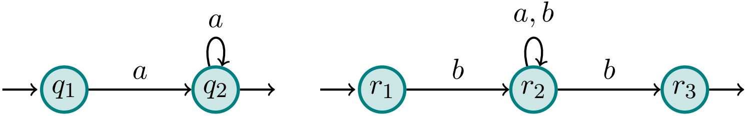
One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q,\boldsymbol{r},s,\boldsymbol{t}),(\boldsymbol{r},u,\boldsymbol{t},v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(3) If $q,r,s \in Q$ and $L(r,s) \neq \emptyset$, then $(q,q,r,s) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Let us look at the example



Language L_0 Language L_1

- By **(1)**, $(q_1,q_2,r_2,r_2) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$ since $a \in L(q_1,q_2) \cap L(r_1,r_2)$.
- By **(3)**, $(q_1,q_1,r_1,r_2) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$ and $(q_2,q_2,r_2,r_3) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$.

An example, the class $\Sigma_1(<)$ (existential FO: $\exists^*$)

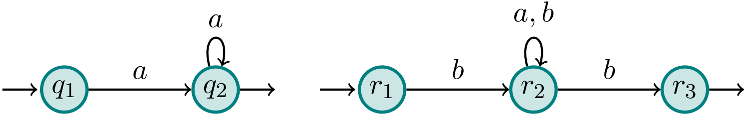
One can prove that $\mathcal{I}_{\Sigma_1}[\mathcal{A}]$ is the least subset of Q^4 such that:

(1) If $L(q,r) \cap L(s,t) \neq \emptyset$, then $(q,r,s,t) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(2) If $(q,\boldsymbol{r},s,\boldsymbol{t}),(\boldsymbol{r},u,\boldsymbol{t},v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$, then $(q,u,s,v) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

(3) If $q,r,s \in Q$ and $L(r,s) \neq \emptyset$, then $(q,q,r,s) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$

Let us look at the example



Language L_0 Language L_1

- By **(1)**, $(q_1,q_2,r_2,r_2) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$ since $a \in L(q_1,q_2) \cap L(r_1,r_2)$.
- By **(3)**, $(q_1,q_1,r_1,r_2) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$ and $(q_2,q_2,r_2,r_3) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$.
- By **(2)**, it then follows that $(q_1,q_2,r_1,r_3) \in \mathcal{I}_{\Sigma_1}[\mathcal{A}]$.

Thus, L_0 is **not** $\Sigma_1(<)$-separable from L_1.

The difficulty of separation: a fixpoint concern

- Problem: the pattern seen for membership repeats itself...

Previous question: tied to a single fixed class $\mathcal{D}$

Does $\mathcal{D}$ have decidable **separation** ?

New question: for an operator $\mathcal{C} \mapsto \boldsymbol{Op}(\mathcal{C})$ (family of classes)

Find hypotheses on $\mathcal{C}$ ensuring that $Op(\mathcal{C})$ has decidable **separation**.

The difficulty of separation: a fixpoint concern

- Problem: the pattern seen for membership repeats itself...

Previous question: tied to a single fixed class $\mathcal{D}$

Does $\mathcal{D}$ have decidable **separation** ?

New question: for an operator $\mathcal{C} \mapsto Op(\mathcal{C})$ (family of classes)

Find hypotheses on $\mathcal{C}$ ensuring that $Op(\mathcal{C})$ has decidable **separation**.

This is difficult.

- **Least fixpoints** often need **more information** than separation.
- $\Rightarrow$ **go beyond separation** (*e.g.*, covering and beyond).

The difficulty of separation: a fixpoint concern

- Problem: the pattern seen for membership repeats itself. . .

Previous question: tied to a single fixed class $\mathcal{D}$

Does $\mathcal{D}$ have decidable **separation** ?

New question: for an operator $\mathcal{C} \mapsto Op(\mathcal{C})$ (family of classes)

Find hypotheses on $\mathcal{C}$ ensuring that $Op(\mathcal{C})$ has decidable **separation**.

This is difficult.

- **Least fixpoints** often need **more information** than separation.
- $\Rightarrow$ **go beyond separation** (*e.g.*, covering and beyond).

Current results are restrict to **very specific kinds of input classes**.
(*e.g.* low levels in alternation hierarchies).

Thank you !

$$\underbrace{\boldsymbol{\Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<)}}_{\text{Separation and covering decidable}} - \underbrace{\boldsymbol{\mathcal{B}\Sigma_3(<) - \Sigma_4(<)}}_{\text{Membership decidable}} \cdots\cdots\cdots$$