Separation and concatenation hierarchies (Part I)

Thomas Place
Joint work with Marc Zeitoun

LaBRI, Bordeaux University

July 14, 2017

Investigating Logics over Words

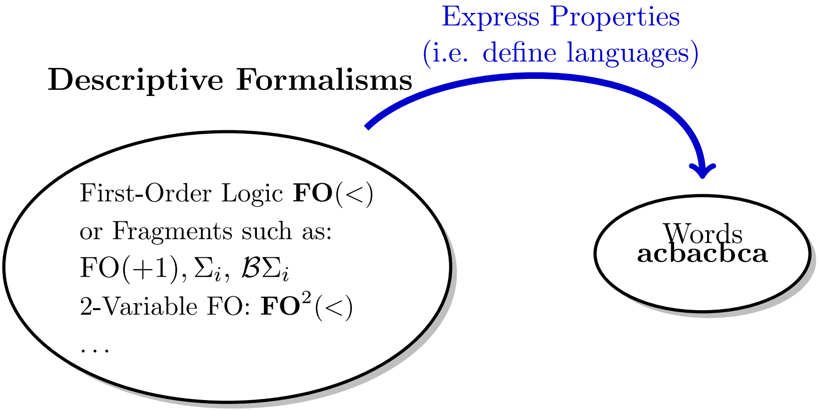
Main Objective

Descriptive Formalisms

First-Order Logic $\mathbf{FO}(<)$
or Fragments such as:
$\mathrm{FO}(+1), \Sigma_i, \mathcal{B}\Sigma_i$
2-Variable FO: $\mathbf{FO}^2(<)$
$\ldots$

Words
acbacbca

Main Objective



Main Objective

Express Properties
(i.e. define languages)

Descriptive Formalisms

First-Order Logic $\mathbf{FO}(<)$
or Fragments such as:
$\mathrm{FO}(+1), \Sigma_i, \mathcal{B}\Sigma_i$
2-Variable FO: $\mathbf{FO}^2(<)$
$\dots$

Words
acbacbca

Objective: For each fragment, **understand** what it can express.

i.e. **What languages belong to the associated class $\mathcal{C}$?**

First-Order Logic over Words (FO(<))

$$\begin{array}{cccccccccccc} a & b & b & b & c & a & a & a & c & a & \in A^* \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \end{array}$$

- A word is a sequence of labeled positions.
- Positions can be quantified.

First-Order Logic over Words (FO(<))

$$\begin{array}{cccccccccccc} a & b & b & b & c & a & a & a & c & a & \in A^* \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & \end{array}$$

- A word is a sequence of labeled positions.
- Positions can be quantified.
- Two kinds of predicates:
  1. Given $a \in A$, $a(x)$ selects positions x whose label is a.
  2. Binary predicate for the (strict) order: $x < y$.

First-Order Logic over Words (FO(<))

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

- A word is a sequence of labeled positions.
- Positions can be quantified.
- Two kinds of predicates:
  1. Given $a \in A$, $a(x)$ selects positions x whose label is a.
  2. Binary predicate for the (strict) order: $x < y$.

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

"for any a in the word, there is a b to its left"

Each sentence defines a language

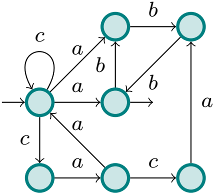
$\Rightarrow$ FO(<) defines a **class of languages**.

We want to **understand classes of languages** (defined by logic)

Methodology: The membership problem

Given such a class $\mathcal{C}$, the goal is to solve the associated **membership problem**:

L a regular language

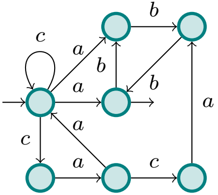


Does L belong to the class $\mathcal{C}$?

Methodology: The membership problem

Given such a class $\mathcal{C}$, the goal is to solve the associated **membership problem**:

L a regular language



Does L belong to the class $\mathcal{C}$?

There are two stages to the problem:

- Stage 1: get an **algorithm** that decides it.
- Stage 2: find a generic way to **construct a sentence** witnessing membership of L in $\mathcal{C}$ when it exists.

Example - McNaughton-Papert-Schützenberger

Given a regular language L, the following properties are equivalent:

- L is definable in FO($<$)
- The minimal automaton of L is counter-free
- The syntactic monoid of L is aperiodic

Example - McNaughton-Papert-Schützenberger

Given a regular language L, the following properties are equivalent:

- L is definable in FO($<$) } **semantic** hard to decide
- The minimal automaton of L is counter-free
- The syntactic monoid of L is aperiodic

} **syntactic** easy to decide

Example - McNaughton-Papert-Schützenberger

Given a regular language L, the following properties are equivalent:

- L is definable in FO($<$)
- The minimal automaton of L is counter-free
- The syntactic monoid of L is aperiodic

semantic hard to decide

syntactic easy to decide

Why is it interesting ?

1. The theorem itself is an effective description of the class FO($<$).
2. The proofs are constructive: if we have the minimal automaton in hand, we can **construct a sentence** for L by induction.
   $\Rightarrow$ We get normal forms for FO($<$) sentences over words.

Altogether, we learn a lot about FO($<$) from this theorem

Summary - Membership

- Understanding a class $\mathcal{C}$ = solving $\mathcal{C}$-**membership**.
- Proof provides a **canonical representation** of languages in $\mathcal{C}$.

Summary - Membership

- Understanding a class $\mathcal{C}$ = solving **$\mathcal{C}$-membership**.
- Proof provides a **canonical representation** of languages in $\mathcal{C}$.

- **Successful methodology since the 70s**, reproduced
  - For other logical classes on words (eg, several restrictions of FO).
  - For other structures: infinite words, finite trees.

Summary - Membership

- Understanding a class $\mathcal{C}$ = solving $\mathcal{C}$-**membership**.
- Proof provides a **canonical representation** of languages in $\mathcal{C}$.

- **Successful methodology since the 70s**, reproduced
  - For other logical classes on words (eg, several restrictions of FO).
  - For other structures: infinite words, finite trees.

- Still, the methodology **fails for important classes**.

The big problem: quantifier alternation

Quantifer Alternation: Classifying Sentences

A simple sentence:

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

Quantifer Alternation: Classifying Sentences

A simple sentence:

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

A more involved one:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 \forall x_7 \exists x_8 \; \varphi(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$$

Quantifer Alternation: Classifying Sentences

A simple sentence:

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

A more involved one:

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 \forall x_7 \exists x_8 \; \varphi(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$$

Complicated = High Quantifier Alternation

Quantifer Alternation: Classifying Sentences

A simple sentence: $\Sigma_2(<)$

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

A more involved one: $\Sigma_7(<)$

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 \forall x_7 \exists x_8 \; \varphi(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$$

Level n: $\Sigma_n(<)$ sentence (in prenex normal form)

$$\underbrace{\exists x_1, \ldots, x_{n_1} \forall y_1, \ldots, y_{n_2} \cdots\cdots}_{n \text{ blocks (starting with } \exists)} \; \underbrace{\varphi(\bar{x}, \bar{y}, \ldots)}_{\text{quantifier-free}}$$

Quantifer Alternation: Classifying Sentences

A simple sentence: $\Sigma_2(<)$

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

A more involved one: $\Sigma_7(<)$

$$\exists x_1 \forall x_2 \exists x_3 \forall x_4 \exists x_5 \forall x_6 \forall x_7 \exists x_8 \; \varphi(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$$

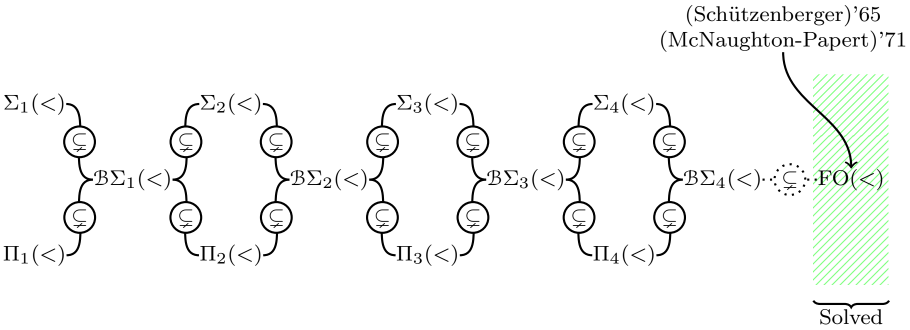
Level n: $\Sigma_n(<)$ sentence (in prenex normal form)

$$\underbrace{\exists x_1, \ldots, x_{n_1} \forall y_1, \ldots, y_{n_2} \cdots\cdots}_{n \text{ blocks (starting with } \exists)} \; \underbrace{\varphi(\bar{x}, \bar{y}, \ldots)}_{\text{quantifier-free}}$$

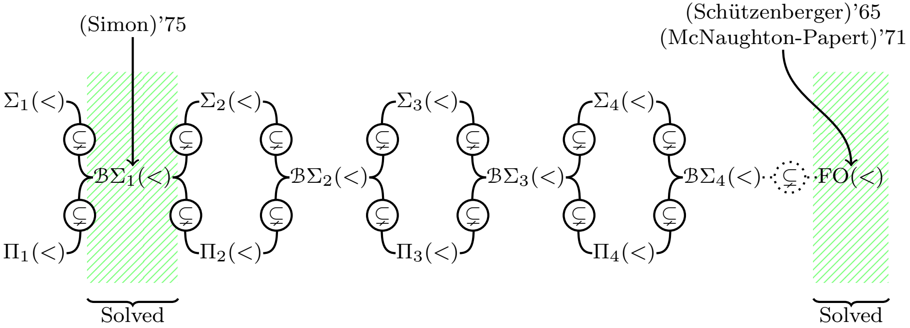
$\Sigma_n(<)$ not closed under complement $\Rightarrow \mathcal{B}\Sigma_n(<)$

$\mathcal{B}\Sigma_n(<)$ sentence $=$ **Boolean combination** of $\Sigma_n(<)$ sentences.

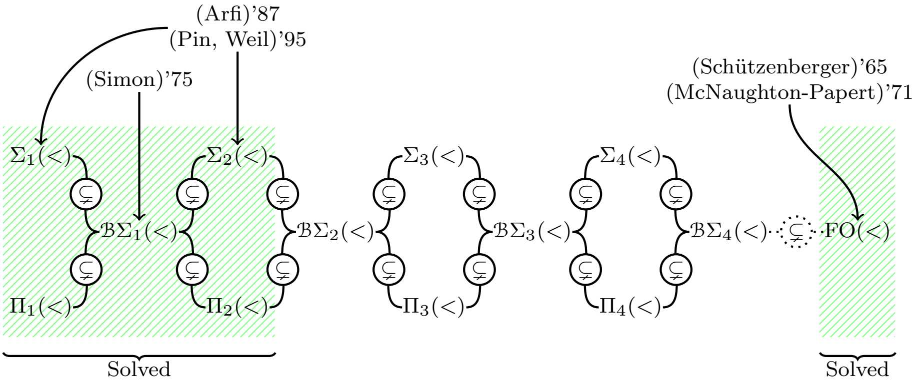
Quantifier Alternation: Membership state of the art



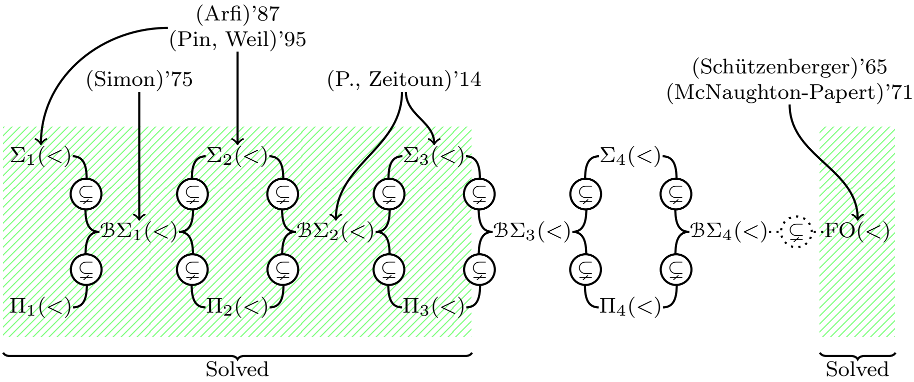
Quantifier Alternation: Membership state of the art



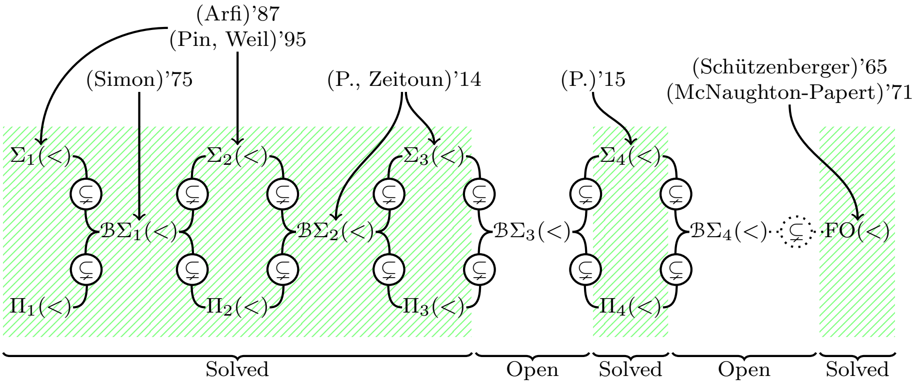
Quantifier Alternation: Membership state of the art



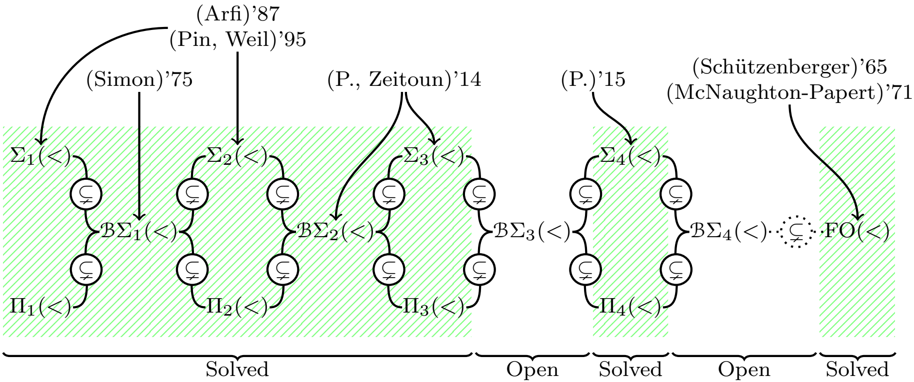
Quantifier Alternation: Membership state of the art



Quantifier Alternation: Membership state of the art

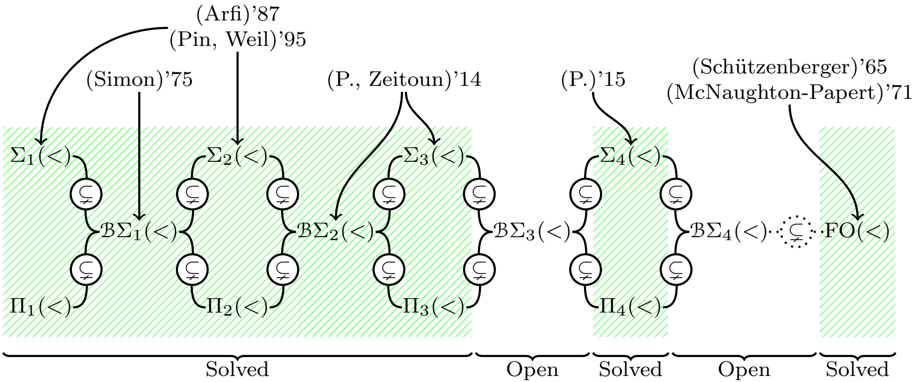


Quantifier Alternation: Membership state of the art



How are this results obtained ?

Quantifier Alternation: Membership state of the art



How are this results obtained ?

The previous slides only present a **third of the story** (at best).

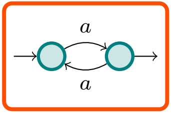
The Separation Problem

Definition

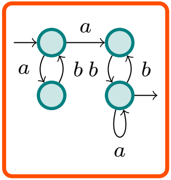
Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:

L_1, L_2 two regular languages

L_1



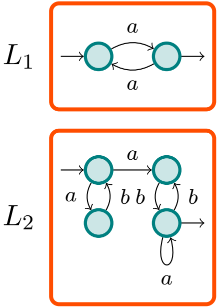
L_2



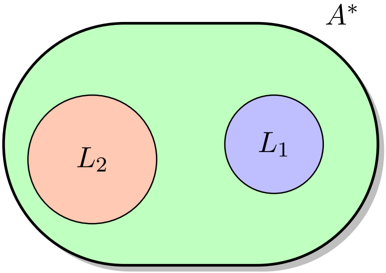
Definition

Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:

L_1, L_2 two regular languages



Can L_1 be separated from L_2 with a language of $\mathcal{C}$?

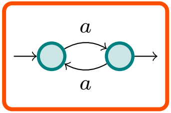


Definition

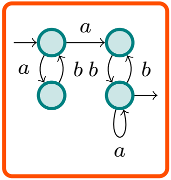
Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:

L_1, L_2 two regular languages

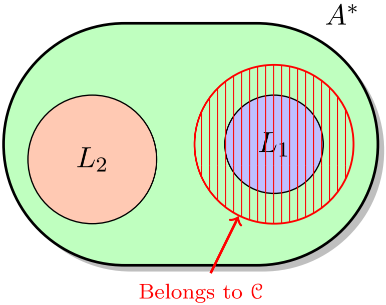
L_1



L_2



Can L_1 be separated from L_2 with a language of $\mathcal{C}$?



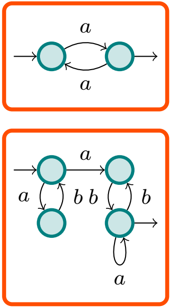
Definition

Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:

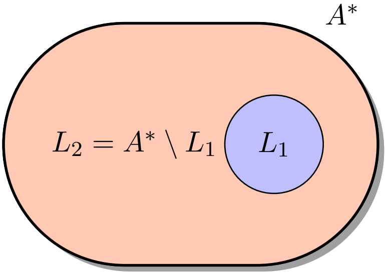
L_1, L_2 two regular languages

L_1

L_2



Can L_1 be separated from L_2 with a language of $\mathcal{C}$?

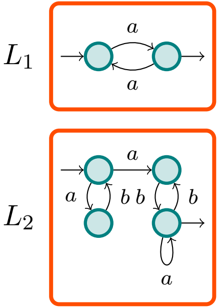


Membership can be formally reduced to separation

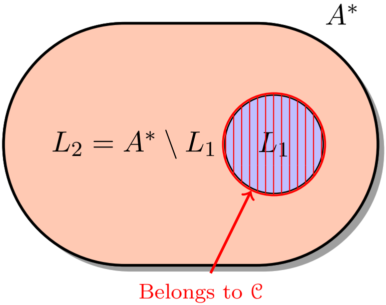
Definition

Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:

L_1, L_2 two regular languages



Can L_1 be separated from L_2 with a language of $\mathcal{C}$?



Membership can be formally reduced to separation

Motivation for Separation (1)

Negative aspect:

- Usually harder than membership.

Motivation for Separation (1)

Negative aspect:

- Usually harder than membership.

Positive aspects:

- More rewarding with respect to the investigated class.

Motivation for Separation (1)

Negative aspect:

- ☹ Usually harder than membership.

Positive aspects:

- ☺ More rewarding with respect to the investigated class.
- ☺ Membership for $\mathcal{C}$ = Techniques applying to languages in $\mathcal{C}$ only.
  Separation for $\mathcal{C}$ = Techniques applying to **all languages**.

Membership for $\mathcal{C}$

Given a language L:

1. Does $L \in \mathcal{C}$?
2. If so, compute a description of L in $\mathcal{C}$.

Separation for $\mathcal{C}$

Given two languages L_1, L_2:

1. Can we **approximate** L_1 with some $K \in \mathcal{C}$? (allowed approximations given by L_2)
2. If so, compute $K \in \mathcal{C}$ realizing this approximation.

Motivation for Separation (2)

All results that we have today for the hierarchy are **based on separation** (or more general problems):

Motivation for Separation (2)

All results that we have today for the hierarchy are **based on separation** (or more general problems):

- While harder, separation provides a better and more robust framework for this investigation.
- Moreover, **interaction** between membership and separation.

Motivation for Separation (2)

All results that we have today for the hierarchy are **based on separation** (or more general problems):

- While harder, separation provides a better and more robust framework for this investigation.
- Moreover, **interaction** between membership and separation.

Transfer theorem (P.,Zeitoun)'14

For all $n \geq 1$,

$$\Sigma_n\text{-separation decidable} \Rightarrow \Sigma_{n+1}\text{-membership decidable}$$

Important Remark

Separation is **harder** than membership. The above above does **not** solve the whole hierarchy.

Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

Notation, for two states p, q: $L_{p,q} = \{w \mid p \xrightarrow{w} q\}$

Forbidden Patterns and Separation

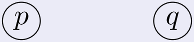
A regular language **is definable in $\Sigma_\mathbf{n}$** iff its minimal automaton has **no pattern**:

Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

Notation, for two states p, q: $L_{p,q} = \{w \mid p \xrightarrow{w} q\}$

Forbidden Patterns and Separation

A regular language **is definable in $\mathbf{\Sigma_n}$** iff its minimal automaton has **no pattern**:

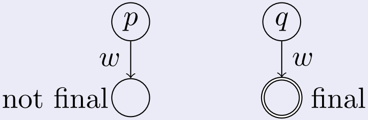


Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

Notation, for two states p, q: $L_{p,q} = \{w \mid p \xrightarrow{w} q\}$

Forbidden Patterns and Separation

A regular language **is definable in Σ_n** iff its minimal automaton has **no pattern**:

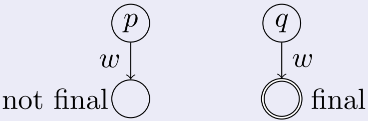


Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

Notation, for two states p, q: $L_{p,q} = \{w \mid p \xrightarrow{w} q\}$

Forbidden Patterns and Separation

A regular language **is definable in Σ_n** iff its minimal automaton has **no pattern**:



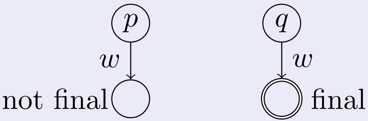
where $L_{p,q}$ is **not Σ_{n-1}-separable** from $L_{p,p} \cap L_{q,q}$

Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

Notation, for two states p, q: $L_{p,q} = \{w \mid p \xrightarrow{w} q\}$

Forbidden Patterns and Separation

A regular language **is definable in $\Sigma_{\mathbf{n}}$** iff its minimal automaton has **no pattern**:



where $L_{p,q}$ is **not $\Sigma_{\mathbf{n-1}}$-separable** from $L_{p,p} \cap L_{q,q}$

Corollary

Solving **$\Sigma_{\mathbf{n-1}}$-separation** yields a solution for **$\Sigma_{\mathbf{n}}$-membership**.

Limits of this approach

We have the following:

$$\Sigma_n\text{-separation decidable} \Rightarrow \Sigma_{n+1}\text{-membership decidable}$$

No similar result with separation on the right side.

Limits of this approach

We have the following:

$$\Sigma_n\text{-separation decidable} \Rightarrow \Sigma_{n+1}\text{-membership decidable}$$

No similar result with separation on the right side.

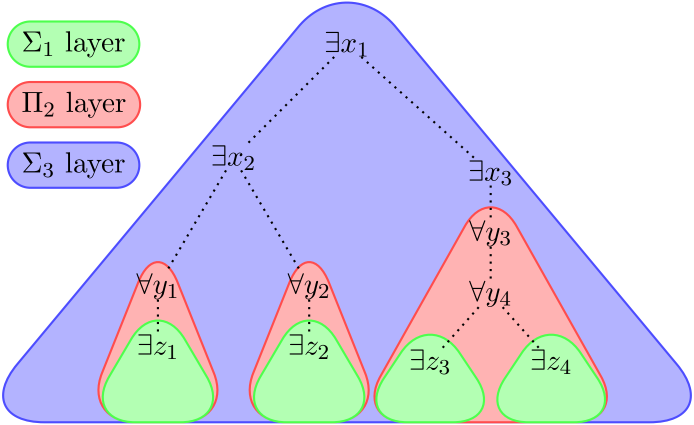
Let us explain why.

Hard part for both membership and separation:
Generic construction of **descriptions in $\mathcal{C}$**.

This is also the case for the transfer theorem.

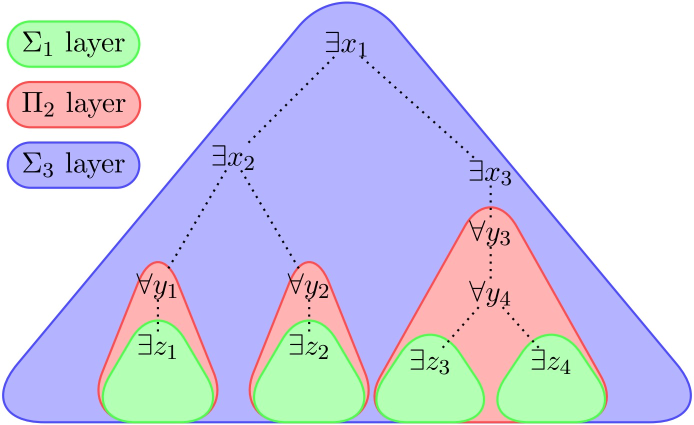
Construction of Σ_n sentences

A Σ_n sentence is **layered**: Consider a Σ_3 sentence



Construction of Σ_n sentences

A Σ_n sentence is **layered**: Consider a Σ_3 sentence



A generic construction should have several phases:
one for each layer

Construction of Σ_n sentences in the transfer theorem

Starting from a **DFA** $\mathcal{A}$ satisfying the transfer theorem, one builds a Σ_n sentence as follows:

- All languages needed for the **layers below Σ_{n-1}** are Σ_{n-1}-separators of $L_{p,q}$ from $L_{p,p} \cap L_{q,q}$ for some states p, q of $\mathcal{A}$.
- One builds the topmost layer (Σ_n) from them by induction on $\mathcal{A}$.

Construction of Σ_n sentences in the transfer theorem

Starting from a **DFA** $\mathcal{A}$ satisfying the transfer theorem, one builds a Σ_n sentence as follows:

- All languages needed for the **layers below Σ_{n-1}** are Σ_{n-1}-separators of $L_{p,q}$ from $L_{p,p} \cap L_{q,q}$ for some states p, q of $\mathcal{A}$.
- One builds the topmost layer (Σ_n) from them by induction on $\mathcal{A}$.

Key ideas

- **We already have the languages of the Σ_n layer in hand**: they are all recognized by $\mathcal{A}$.
- The lower layers are built by **approximating these languages** with Σ_{n-1}-separation.

Construction of Σ_n sentences in the transfer theorem

Starting from a **DFA** $\mathcal{A}$ satisfying the transfer theorem, one builds a Σ_n sentence as follows:

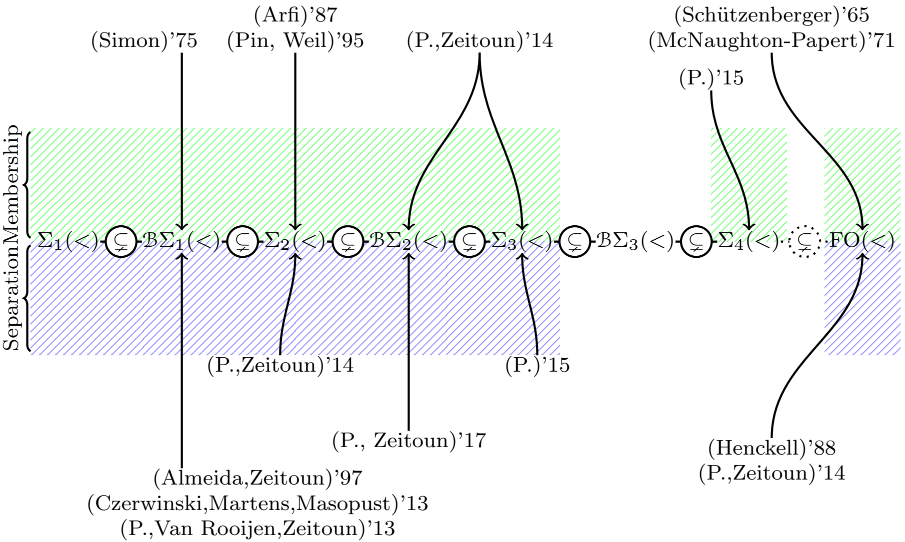
- All languages needed for the **layers below Σ_{n-1}** are Σ_{n-1}-separators of $L_{p,q}$ from $L_{p,p} \cap L_{q,q}$ for some states p, q of $\mathcal{A}$.
- One builds the topmost layer (Σ_n) from them by induction on $\mathcal{A}$.

Key ideas

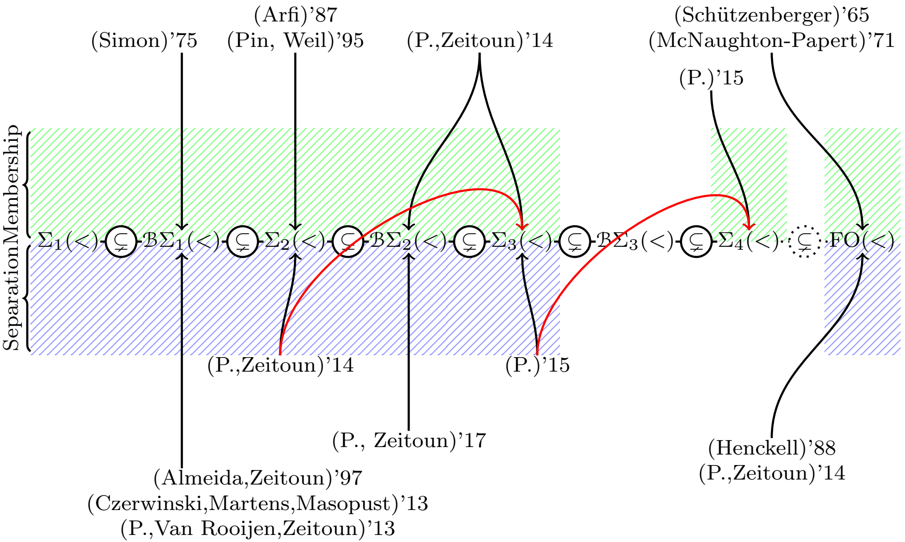
- **We already have the languages of the Σ_n layer in hand**: they are all recognized by $\mathcal{A}$.
- The lower layers are built by **approximating these languages** with Σ_{n-1}-separation.

Separation is different: we do not have the Σ_n-layer in hand.
$\Rightarrow$ All layers must be considered simultaneously.

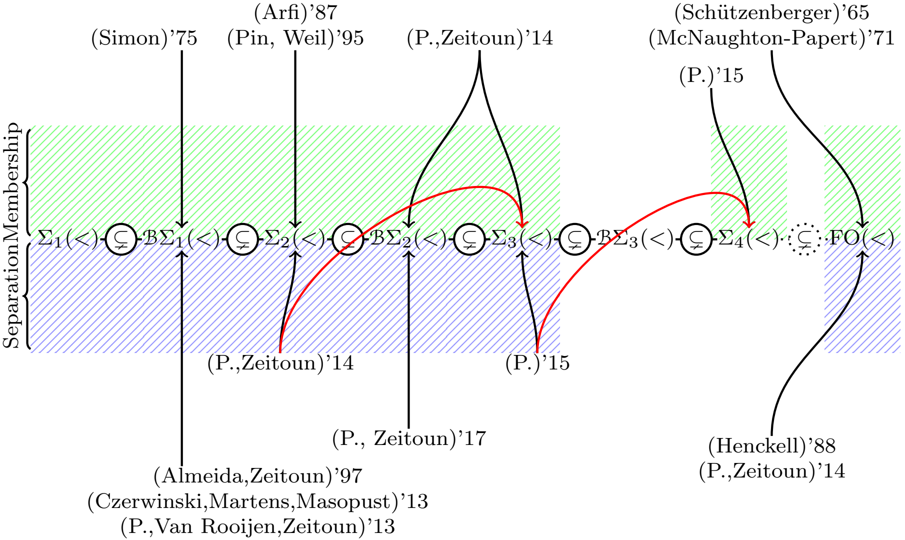
Current state of the art: Separation



Current state of the art: Separation



Current state of the art: Separation



We are still missing **one third of the story**.

Concatenation hierarchies

Star-free languages (1)

McNaughton-Papert'71

Given a regular language L, the following properties are equivalent:

- L may be defined by an FO($<$) sentence.
- L is **star-free**.

Star-free languages (1)

McNaughton-Papert'71

Given a regular language L, the following properties are equivalent:

- L may be defined by an FO($<$) sentence.
- L is **star-free**.

Star-free languages

- $\emptyset$ and A^* are star-free.

  $\Rightarrow$ Corresponds to the FO($<$) sentences $\perp$ and $\top$.

Star-free languages (1)

McNaughton-Papert'71

Given a regular language L, the following properties are equivalent:

- L may be defined by an FO($<$) sentence.
- L is **star-free**.

Star-free languages

- $\emptyset$ and A^* are star-free.
  $\Rightarrow$ Corresponds to the FO($<$) sentences $\perp$ and $\top$.
- Closed under **union**, **union** and **complement**.
  $\Rightarrow$ Corresponds to Boolean connectives in FO($<$).

Star-free languages (1)

McNaughton-Papert'71

Given a regular language L, the following properties are equivalent:

- L may be defined by an FO($<$) sentence.
- L is **star-free**.

Star-free languages

- $\emptyset$ and A^* are star-free.
  $\Rightarrow$ Corresponds to the FO($<$) sentences $\perp$ and $\top$.
- Closed under **union**, **union** and **complement**.
  $\Rightarrow$ Corresponds to Boolean connectives in FO($<$).
- Closed under **marked concatenation**:

$$\text{Given } a \in A \quad K, L, a \mapsto KaL$$

  $\Rightarrow$ Corresponds to existential quantification in FO($<$).

$$\exists x \; a(x) \wedge \varphi_K^{<x}(x) \wedge \varphi_L^{>x}(x)$$

Star-free languages (2)

- Going from star-free languages to FO($<$) is easy:
  Star-free description is a **FO($<$) sentence in normal form**.

Star-free languages (2)

- Going from star-free languages to FO($<$) is easy:

  Star-free description is a **FO($<$) sentence in normal form**.

- Other direction is less immediate:

  More syntactical freedom in FO($<$) sentences.

Star-free languages (2)

- Going from star-free languages to FO($<$) is easy:

  Star-free description is a **FO($<$) sentence in normal form**.
- Other direction is less immediate:

  More syntactical freedom in FO($<$) sentences.

However, in generic constructions of FO($<$) sentences, this additional **freedom is never used**.

For building FO($<$) languages, one always starts from $\emptyset$ and A^* using only **Boolean operations** and **marked concatenations**.

Star-free languages (2)

- Going from star-free languages to FO($<$) is easy:

  Star-free description is a **FO($<$) sentence in normal form**.

- Other direction is less immediate:

  More syntactical freedom in FO($<$) sentences.

However, in generic constructions of FO($<$) sentences, this additional **freedom is never used**.

For building FO($<$) languages, one always starts from $\emptyset$ and A^* using only **Boolean operations** and **marked concatenations**.

This is also the case for classes in the quantifier alternation hierarchy of FO($<$).

The Straubing Thérien Hierarchy'81

Classifies the star-free languages into half and full levels:

$$0 \longrightarrow \tfrac{1}{2} \longrightarrow 1 \longrightarrow \tfrac{3}{2} \longrightarrow 2 \longrightarrow \tfrac{5}{2} \longrightarrow 3 \cdots\cdots$$

The Straubing Thérien Hierarchy'81

Classifies the star-free languages into half and full levels:

$$0 \longrightarrow \tfrac{1}{2} \longrightarrow 1 \longrightarrow \tfrac{3}{2} \longrightarrow 2 \longrightarrow \tfrac{5}{2} \longrightarrow 3 \cdots\cdots$$

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

The Straubing Thérien Hierarchy'81

Classifies the star-free languages into half and full levels:

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \longrightarrow 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \longrightarrow 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \longrightarrow 3 \cdots\cdots$$

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

Polynomial closure

$Pol(\mathcal{C})$ built by closing the class $\mathcal{C}$ under:

- Union ($\bigcup$).
- Intersection ($\bigcap$).
- Marked concatenation ($K, L, a \mapsto KaL$).

The Straubing Thérien Hierarchy'81

Classifies the star-free languages into half and full levels:

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

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

Polynomial closure

$Pol(\mathcal{C})$ built by closing the class $\mathcal{C}$ under:

- Union ($\bigcup$).
- Intersection ($\bigcap$).
- Marked concatenation ($K, L, a \mapsto KaL$).

Boolean closure

$Bool(\mathcal{C})$ built by closing the class $\mathcal{C}$ under:

- Union ($\bigcup$).
- Intersection ($\bigcap$).
- Complement ($L \mapsto A^* \setminus L$).

The Straubing Thérien Hierarchy'81

Classifies the star-free languages into half and full levels:

$$
\begin{tikzcd}
0 \arrow[r, "\mathrm{Pol}"] & \frac{1}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[d, leftrightarrow] & 1 \arrow[r, "\mathrm{Pol}"] \arrow[d, leftrightarrow] & \frac{3}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[d, leftrightarrow] & 2 \arrow[r, "\mathrm{Pol}"] \arrow[d, leftrightarrow] & \frac{5}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[d, leftrightarrow] & 3 \cdots\cdots \arrow[d, leftrightarrow] \\
\{\emptyset, A^*\} & & & & & & \\
\mathcal{B}\Sigma_0(<) & \Sigma_1(<) & \mathcal{B}\Sigma_1(<) & \Sigma_2(<) & \mathcal{B}\Sigma_2(<) & \Sigma_3(<) & \mathcal{B}\Sigma_3(<)
\end{tikzcd}
$$

((Thomas)'82 (Perrin,Pin)'86)

Polynomial closure

$Pol(\mathcal{C})$ built by closing the class $\mathcal{C}$ under:

- Union ($\bigcup$).
- Intersection ($\bigcap$).
- Marked concatenation $(K, L, a \mapsto KaL)$.

Boolean closure

$Bool(\mathcal{C})$ built by closing the class $\mathcal{C}$ under:

- Union ($\bigcup$).
- Intersection ($\bigcap$).
- Complement $(L \mapsto A^* \setminus L)$.

Generic template: Concatenation hierarchies

Previous slide is an example of a **generic** construction.

Generic template: Concatenation hierarchies

0
Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

$$w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$$

$$Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2}$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$

  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$

  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2}$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

$$w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$$

$$Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,
  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$
  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2}$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$

  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,
  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$
  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,

$$w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$$

$$Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$$

All results for quantifier alternation **can be lifted as generic results** for concatenation hierarchies whose basis is finite.

Generic template: Concatenation hierarchies

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

$\mathcal{C}$ must be closed under:

- **Boolean operations**.
- **Quotients**. For $L \in \mathcal{C}$, $w \in A^*$,
  $w^{-1}L \stackrel{\text{def}}{=} \{u \in A^* \mid wu \in L\} \in \mathcal{C}$
  $Lw^{-1} \stackrel{\text{def}}{=} \{u \in A^* \mid uw \in L\} \in \mathcal{C}$

All results for quantifier alternation **can be lifted as generic results** for concatenation hierarchies whose basis is finite.

Before we explain how, let us further motivate the introduction of concatenation hierarchies.

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant FO($\mathcal{C}$) of first-order logic** equipped with the following signature:

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant FO($\mathcal{C}$) of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant $\mathbf{FO}(\mathcal{C})$ of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

- **Infix** $I_L(x, y)$ (binary):

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

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant $\mathbf{FO}(\mathcal{C})$ of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

- **Infix** $I_L(x, y)$ (binary):
$$a_1 \cdots a_n \models I_L(i, j) \text{ iff } i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L$$
- **Prefix** $P_L(x)$ (unary):
$$a_1 \cdots a_n \models P_L(i) \text{ iff } a_1 \cdots a_{i-1} \in L$$

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant $\mathbf{FO}(\mathcal{C})$ of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

- **Infix** $I_L(x, y)$ (binary):
$$a_1 \cdots a_n \models I_L(i, j) \text{ iff } i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L$$
- **Prefix** $P_L(x)$ (unary):
$$a_1 \cdots a_n \models P_L(i) \text{ iff } a_1 \cdots a_{i-1} \in L$$
- **Suffix** $S_L(x)$ (unary):
$$a_1 \cdots a_n \models S_L(i) \text{ iff } a_{i+1} \cdots a_n \in L$$

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant $\mathbf{FO}(\mathcal{C})$ of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

- **Infix** $I_L(x, y)$ (binary):

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

- **Prefix** $P_L(x)$ (unary):

$$a_1 \cdots a_n \models P_L(i) \text{ iff } a_1 \cdots a_{i-1} \in L$$

- **Suffix** $S_L(x)$ (unary):

$$a_1 \cdots a_n \models S_L(i) \text{ iff } a_{i+1} \cdots a_n \in L$$

- **Whole word** N_L (nullary):

$$a_1 \cdots a_n \models N_L \text{ iff } a_1 \cdots a_n \in L$$

The logical connection is generic (1)

Given a basis $\mathcal{C}$, we define a **variant $\mathrm{FO}(\mathcal{C})$ of first-order logic** equipped with the following signature:

- Label predicates: $a(x), b(x), \ldots$.

For each $L \in \mathcal{C}$, we add **four predicates**:

- **Infix** $I_L(x, y)$ (binary):
$$a_1 \cdots a_n \models I_L(i, j) \text{ iff } i < j \text{ and } a_{i+1} \cdots a_{j-1} \in L$$
- **Prefix** $P_L(x)$ (unary):
$$a_1 \cdots a_n \models P_L(i) \text{ iff } a_1 \cdots a_{i-1} \in L$$
- **Suffix** $S_L(x)$ (unary):
$$a_1 \cdots a_n \models S_L(i) \text{ iff } a_{i+1} \cdots a_n \in L$$
- **Whole word** N_L (nullary):
$$a_1 \cdots a_n \models N_L \text{ iff } a_1 \cdots a_n \in L$$

The concatenation hierarchy of basis $\mathcal{C}$ corresponds to the quantifier alternation hierarchy within $\mathrm{FO}(\mathcal{C})$.

The logical connection is generic (2)

The concatenation hierarchy of basis $\mathcal{C}$ corresponds to the quantifier alternation hierarchy within $\mathrm{FO}(\mathcal{C})$.

$$\begin{tikzcd}
0 \arrow[r, "\mathrm{Pol}"] & \frac{1}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[dd, leftrightarrow] & 1 \arrow[r, "\mathrm{Pol}"] \arrow[dd, leftrightarrow] & \frac{3}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[dd, leftrightarrow] & 2 \arrow[r, "\mathrm{Pol}"] \arrow[dd, leftrightarrow] & \frac{5}{2} \arrow[r, "\mathrm{Bool}"'] \arrow[dd, leftrightarrow] & 3 \arrow[dd, leftrightarrow] \\
\text{Basis: class } \mathcal{C} & & & & & & \\
\mathcal{B}\Sigma_0(\mathcal{C}) & \Sigma_1(\mathcal{C}) & \mathcal{B}\Sigma_1(\mathcal{C}) & \Sigma_2(\mathcal{C}) & \mathcal{B}\Sigma_2(\mathcal{C}) & \Sigma_3(\mathcal{C}) & \mathcal{B}\Sigma_3(\mathcal{C})
\end{tikzcd}$$

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow \mathrm{FO}(<)$

- $I_{A^*}(x, y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x, y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\bot$

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow \mathrm{FO}(<)$

- $I_{A^*}(x,y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x,y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\perp$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow \mathrm{FO}(<)$

- $I_{A^*}(x,y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x,y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\perp$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$

- $I_\varepsilon(x,y)$ is $x + 1 = y$.

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow$ FO(<)

- $I_{A^*}(x, y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x, y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\bot$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$

- $I_\varepsilon(x, y)$ is $x + 1 = y$.
- $P_\varepsilon(x)$ and $S_\varepsilon(x)$ are $min(x)$ and $max(x)$.

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow \mathrm{FO}(<)$

- $I_{A^*}(x, y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x, y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\bot$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$

- $I_\varepsilon(x, y)$ is $x + 1 = y$.
- $P_\varepsilon(x)$ and $S_\varepsilon(x)$ are $min(x)$ and $max(x)$.
- N_ε is ε.

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow$ FO($<$)

- $I_{A^*}(x,y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x,y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\perp$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$

- $I_\varepsilon(x,y)$ is $x + 1 = y$.
- $P_\varepsilon(x)$ and $S_\varepsilon(x)$ are $min(x)$ and $max(x)$.
- N_ε is ε.
- Predicates obtained from A^+ are expressed from the others.

The logical connection is generic: Examples

The Straubing-Thérien hierarchy

Basis $\mathcal{C} = \{\emptyset, A^*\} \Rightarrow \mathrm{FO}(<)$

- $I_{A^*}(x, y)$ is $x < y$.
- $P_{A^*}(x), S_{A^*}(x), N_{A^*}$ are equivalent to $\top$.
- $I_\emptyset(x, y), P_\emptyset(x), S_\emptyset(x), N_\emptyset$ are equivalent to $\bot$

The dot-depth hierarchy (Brzozowski,Cohen)'71

Basis $\mathcal{C} = \{\emptyset, \{\varepsilon\}, A^+, A^*\} \Rightarrow \mathrm{FO}(<, +1, min, max, \varepsilon)$ (Thomas)'82

- $I_\varepsilon(x, y)$ is $x + 1 = y$.
- $P_\varepsilon(x)$ and $S_\varepsilon(x)$ are $min(x)$ and $max(x)$.
- N_ε is ε.
- Predicates obtained from A^+ are expressed from the others.

Generic separation results

Generic separation results (1)

$$\underset{\substack{\text{Basis:}\\ \text{class } \mathcal{C}}}{0} \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Generic Separation Results

If $\mathcal{C}$ is **finite**, then **separation is decidable** for the following,

Generic separation results (1)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Generic Separation Results

If $\mathcal{C}$ is **finite**, then **separation is decidable** for the following,

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

Generic separation results (1)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Generic Separation Results

If $\mathcal{C}$ is **finite**, then **separation is decidable** for the following,

1. $Pol(\mathcal{C})$.
2. $BPol(\mathcal{C})$ (i.e. $Bool(Pol(\mathcal{C}))$).

Generic separation results (1)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Generic Separation Results

If $\mathcal{C}$ is **finite**, then **separation is decidable** for the following,

1. $Pol(\mathcal{C})$.
2. $BPol(\mathcal{C})$ (i.e. $Bool(Pol(\mathcal{C}))$).
3. $Pol(BPol(\mathcal{C}))$.

Generic separation results (2)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Advantages:

- 🙂 These results treat many classes.
- 🙂 **All we know** about separation is captured by **three results**.
- 🙂 We **pinpoint the hypotheses** which we really need.

Generic separation results (2)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Advantages:

- 🙂 These results treat many classes.
- 🙂 **All we know** about separation is captured by **three results**.
- 🙂 We **pinpoint the hypotheses** which we really need.

Downside:

- 🙁 Generic proof are harder than specific ones.

Generic separation results (2)

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Basis:
class $\mathcal{C}$

Advantages:

- These results treat many classes.
- **All we know** about separation is captured by **three results**.
- We **pinpoint the hypotheses** which we really need.

Downside:

- Generic proof are harder than specific ones.

Wait ! The speaker lied ! Refund my 53 €!!!!!!

The results for FO($<$) went **one full level higher**, didn't they ?

The almighty alphabet argument

Logical point of view (hierarchy within FO($<$)):

$$\mathcal{B}\Sigma_0(<) - \Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<)$$

The almighty alphabet argument

Logical point of view (hierarchy within FO(<)):

$$\mathcal{B}\Sigma_0(<) - \Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<)$$

Languages point of view (Straubing-Thérien hierarchy):

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots$$

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

The almighty alphabet argument

Logical point of view (hierarchy within FO($<$)):

$$\mathcal{B}\Sigma_0(<) — \Sigma_1(<) — \mathcal{B}\Sigma_1(<) — \Sigma_2(<) — \mathcal{B}\Sigma_2(<) — \Sigma_3(<) — \mathcal{B}\Sigma_3(<)$$

Languages point of view (Straubing-Thérien hierarchy):

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots$$

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

Finite class AT
(**Alphabet testable**)
Boolean combinations of languages of the form A^*aA^* for some $a \in A$

The almighty alphabet argument

Logical point of view (hierarchy within FO(<)):

$$\mathcal{B}\Sigma_0(<) - \Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<)$$

Languages point of view (Straubing-Thérien hierarchy):

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots$$

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

Pol

(Pin,Straubing)'81

Finite class AT
(**Alphabet testable**)
Boolean combinations of languages of the form A^*aA^* for some $a \in A$

New level 0

The almighty alphabet argument

Logical point of view (hierarchy within FO(<)):

$$\mathcal{B}\Sigma_0(<) - \Sigma_1(<) - \mathcal{B}\Sigma_1(<) - \Sigma_2(<) - \mathcal{B}\Sigma_2(<) - \Sigma_3(<) - \mathcal{B}\Sigma_3(<)$$

Languages point of view (Straubing-Thérien hierarchy):

$$0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots$$

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

Pol

$\frac{1}{2}$ 1 $\frac{3}{2}$

(Pin,Straubing)'81

Finite class AT
(**Alphabet testable**)
Boolean combinations of languages of the form A^*aA^* for some $a \in A$

New level 0

Summary

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.

Summary

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.

Summary

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.

Summary

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Summary

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Additional important result:

5. Generic reduction. For any half or full level n:

Transfer of separation

Level n in the dot-depth reduces to level n in the Straubing-Thérien.

Conclusion

Conclusion

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Conclusion

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Some words about **complexity**:

1. Complexity depends on $|\mathcal{C}|$ (tied to the implicit alphabet).

Conclusion

Everything we know is captured by only **four generic results**:

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Some words about **complexity**:

1. Complexity depends on $|\mathcal{C}|$ (tied to the implicit alphabet).
2. $Pol(\text{AT})$ and $BPol(\text{AT})$ are **PSpace**(-complete).

Conclusion

Everything we know is captured by only **four generic results**:

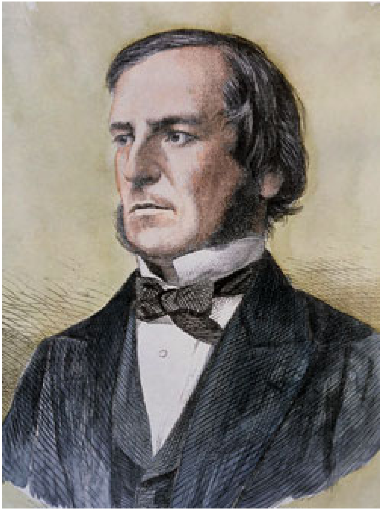
1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.
4. For any $\mathcal{C}$,
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

Some words about **complexity**:

1. Complexity depends on $|\mathcal{C}|$ (tied to the implicit alphabet).
2. $Pol(\mathrm{AT})$ and $BPol(\mathrm{AT})$ are **PSpace**(-complete).
3. If the alphabet is fixed, or $|\mathcal{C}|$ is constant,

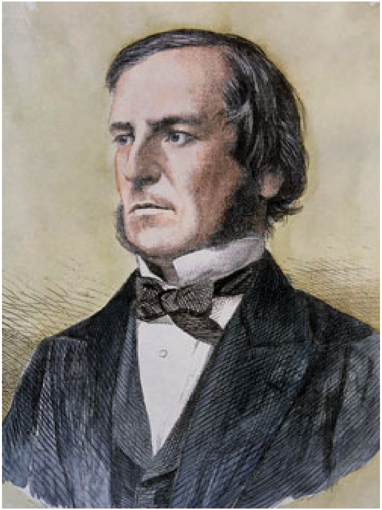
   $Pol(\mathcal{C})$-separation and $BPol(\mathcal{C})$-separation are in **PTime**

What you should know



George Boole
(1815-1864)

What you should know



George Boole
(1815-1864)

This guy is evil

We don't understand negation

$$\Sigma_1(<) \subsetneq \mathcal{B}\Sigma_1(<) \subsetneq \Sigma_2(<) \subsetneq \mathcal{B}\Sigma_2(<) \subsetneq \Sigma_3(<) \subsetneq \mathcal{B}\Sigma_3(<) \subsetneq \Sigma_4(<) \cdots \subsetneq \cdots \mathrm{FO}(<)$$

this picture is misleading

We don't understand negation

$$\Sigma_1(<) \subsetneq \mathcal{B}\Sigma_1(<) \subsetneq \Sigma_2(<) \subsetneq \mathcal{B}\Sigma_2(<) \subsetneq \Sigma_3(<) \subsetneq \mathcal{B}\Sigma_3(<) \subsetneq \Sigma_4(<) \cdots \subsetneq \cdots \mathrm{FO}(<)$$

this picture is misleading

Only three results

1. $\mathcal{C}$ finite $\Rightarrow$ $Pol(\mathcal{C})$-separation decidable.
2. $\mathcal{C}$ finite $\Rightarrow$ $BPol(\mathcal{C})$-separation decidable.
3. $\mathcal{C}$ finite $\Rightarrow$ $Pol(BPol(\mathcal{C}))$-separation decidable.

We are only able to handle **one** negation.

We don't understand negation (2)

Let's consider two other examples

Two-variables first-order logic ($FO^2(<)$): plenty of negation

Separation is decidable. Operations used to build separators:

- Union.
- Concatenations.

We don't understand negation (2)

Let's consider two other examples

Two-variables first-order logic ($FO^2(<)$): plenty of negation

Separation is decidable. Operations used to build separators:

- Union.
- Concatenations.

First-order logic ($FO(<)$): even more negation

Separation still decidable. Operations used to build separators:

- Union.
- Concatenations.
- Kleene star (simulated with negation in special situations)

We don't understand negation (2)

Let's consider two other examples

Two-variables first-order logic (FO^2($<$)): plenty of negation

Separation is decidable. Operations used to build separators:

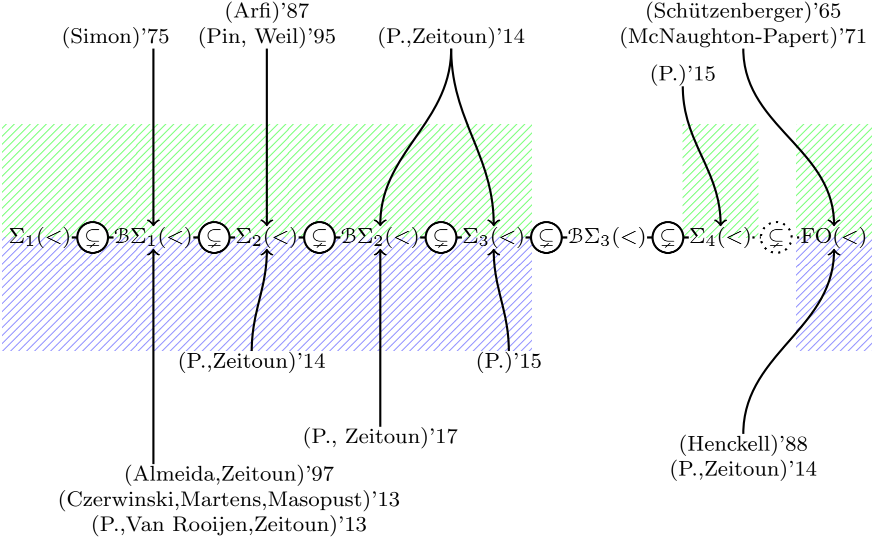
- Union.
- Concatenations.

First-order logic (FO($<$)): even more negation

Separation still decidable. Operations used to build separators:

- Union.
- Concatenations.
- Kleene star (simulated with negation in special situations)

There are **three** operations that we understand: union, concatenation and (to a lesser extent) Kleene star. Complement is evil.



(Arfi)'87
(Simon)'75
(Pin, Weil)'95
(P.,Zeitoun)'14
(Schützenberger)'65
(McNaughton-Papert)'71
(P.)'15
$\Sigma_1(<) \subsetneq \mathcal{B}\Sigma_1(<) \subsetneq \Sigma_2(<) \subsetneq \mathcal{B}\Sigma_2(<) \subsetneq \Sigma_3(<) \subsetneq \mathcal{B}\Sigma_3(<) \subsetneq \Sigma_4(<) \subsetneq FO(<)$
(P.,Zeitoun)'14
(P.)'15
(P., Zeitoun)'17
(Henckell)'88
(P.,Zeitoun)'14
(Almeida,Zeitoun)'97
(Czerwinski,Martens,Masopust)'13
(P.,Van Rooijen,Zeitoun)'13

Thank You