Separation and Concatenations

Thomas Place
Joint work with Marc Zeitoun (Supreme leader of DELTA)

LaBRI, Bordeaux University

February 9, 2017

Certified “100% transducer-free”

Investigating Logics over Words

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.

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. A binary for the (strict) order: $x < y$.

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

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

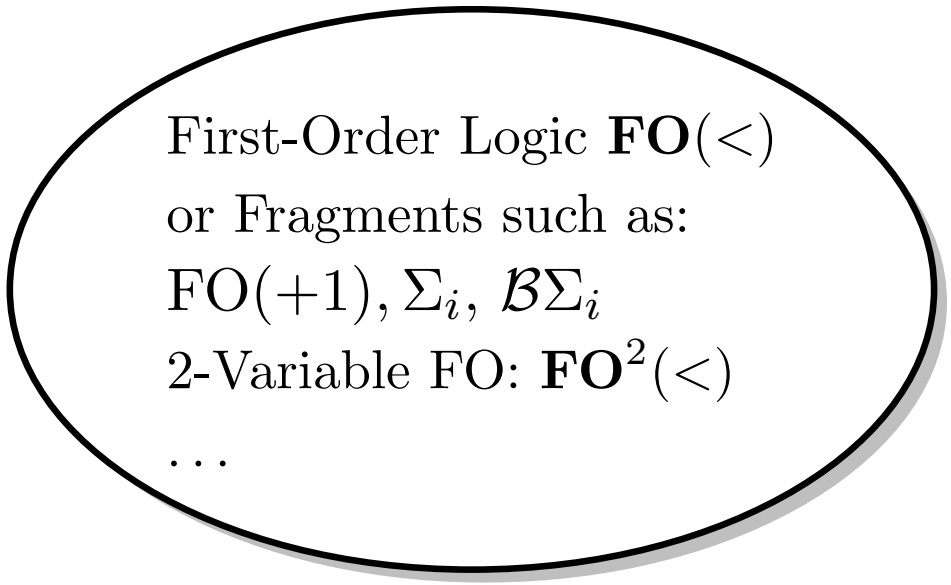
- A word is a sequence of labeled positions.
- Positions can be quantified.
- Two kinds of predicates:
  1. Given $a \in A$, $a(x)$ selects positions x whose label is a.
  2. A binary for the (strict) order: $x < y$.

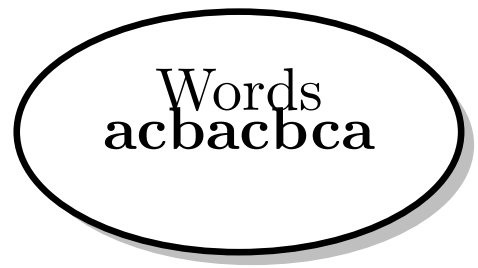
$$\forall x\ (a(x) \Rightarrow \exists y\ (b(y) \wedge (y < x)))$$
"for any a in the word, there is a b to its left"

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

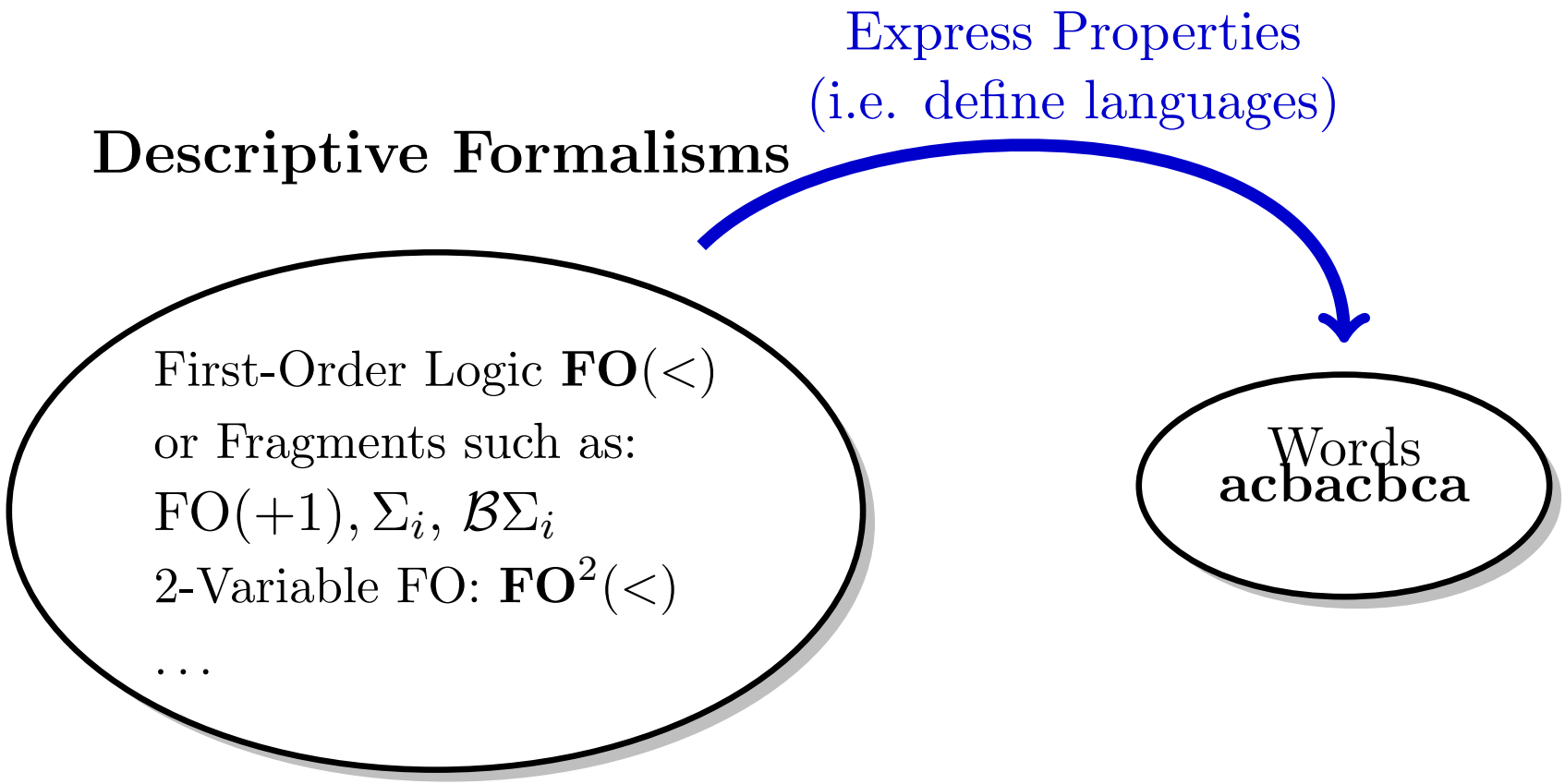
Main Objective

Descriptive Formalisms

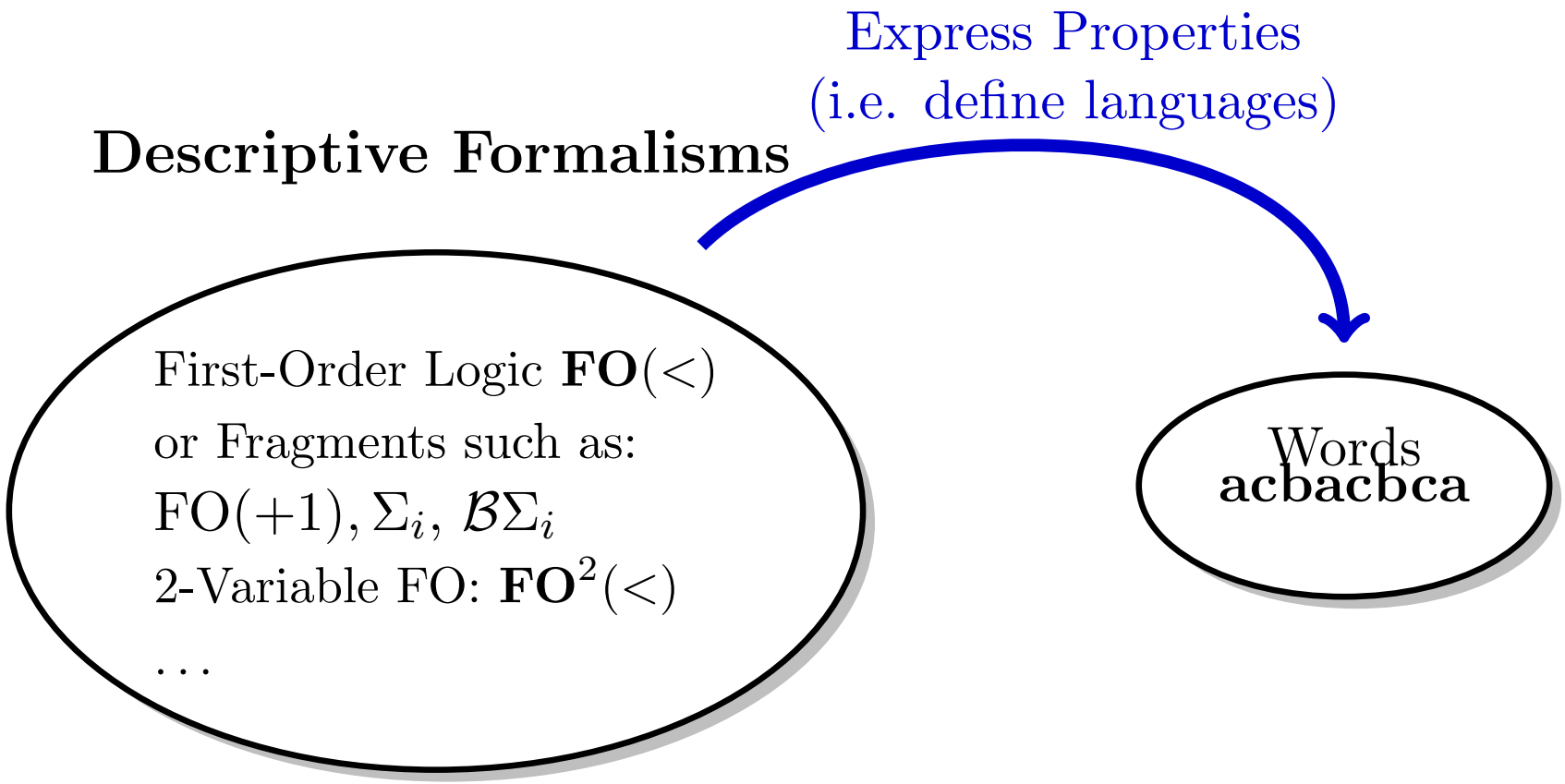




Main Objective



Main Objective



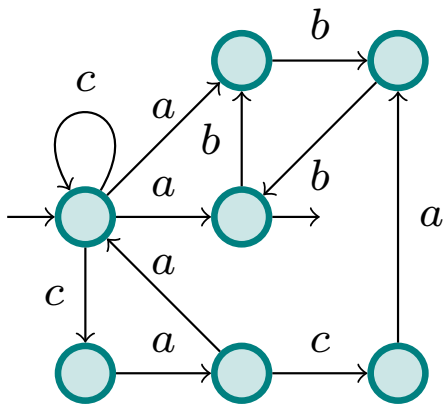
Objective: For each fragment, understand what it can express.

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

Methodology: The membership problem

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

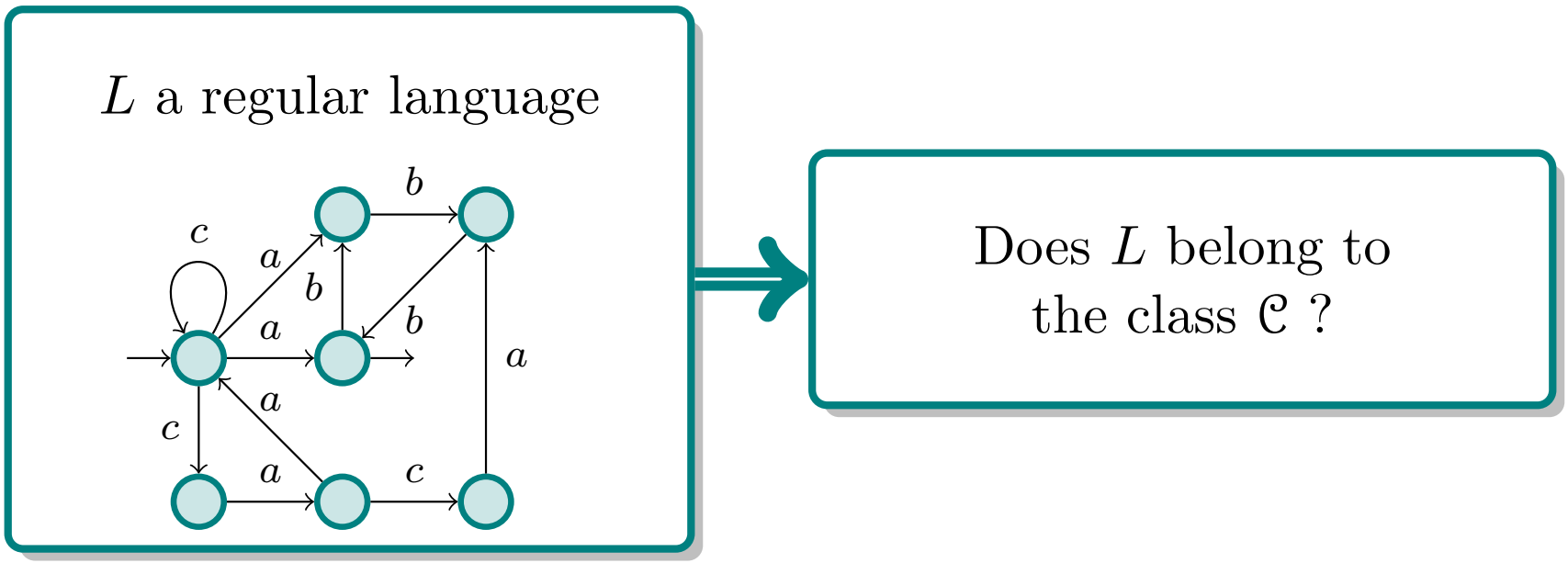
L a regular language



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

Methodology: The membership problem

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



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 $\mathrm{FO}(<)$
- 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($<$) — **semantic**, hard to decide
- The minimal automaton of L is counter-free — **syntactic**, easy to decide
- The syntactic monoid of L is aperiodic — **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 Formulas

What is a simple formula ?

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

What is a complicated formula ?

Quantifer Alternation: Classifying Formulas

What is a simple formula ?

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

What is a complicated formula ?

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

Complicated = High Quantifier Alternation

Quantifer Alternation: Classifying Formulas

What is a simple formula ?

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

$\Rightarrow \Pi_2$ formula.

What is a complicated formula ?

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

$\Rightarrow \Sigma_7$ formula (φ quantifier-free)

Complicated = High Quantifier Alternation

FO Quantifier Alternation Hierarchy

Level i: Σ_i

For all i, a Σ_i formula is (in prenex normal form)

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

FO Quantifier Alternation Hierarchy

Level i: Σ_i

For all i, a Σ_i formula is (in prenex normal form)

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

Σ_i is not closed under complement $\Rightarrow$ we get two other classes:

Level i: Π_i

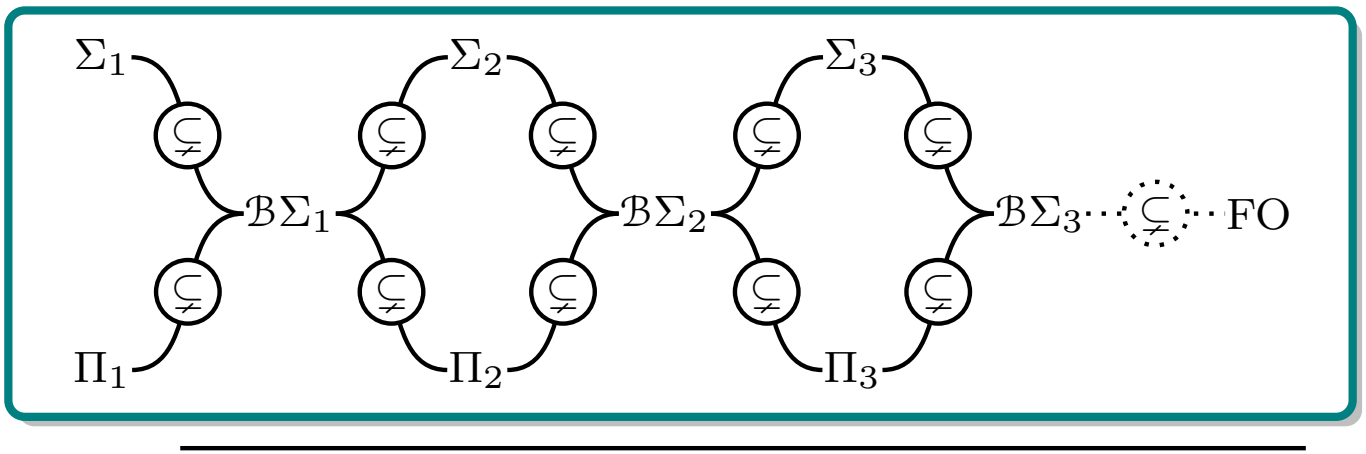
Negation of a Σ_i formula:

$$\underbrace{\forall x_1, \ldots, x_{n_1} \exists y_1, \ldots, y_{n_2} \cdots}_{i \text{ blocks (starting with } \forall)} \quad \varphi$$

Level i: $\mathcal{B}\Sigma_i$

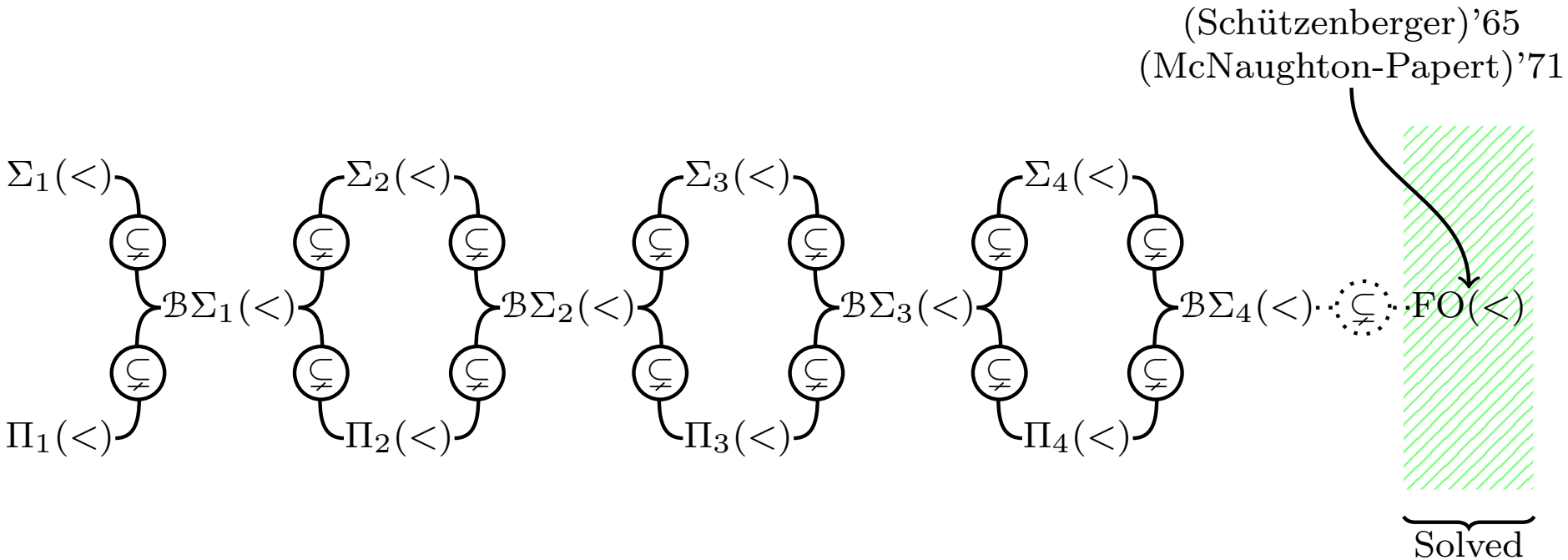
Boolean combinations of Σ_i (and Π_i) formulas.

FO(<) Quantifier Alternation Hierarchy: The Problem

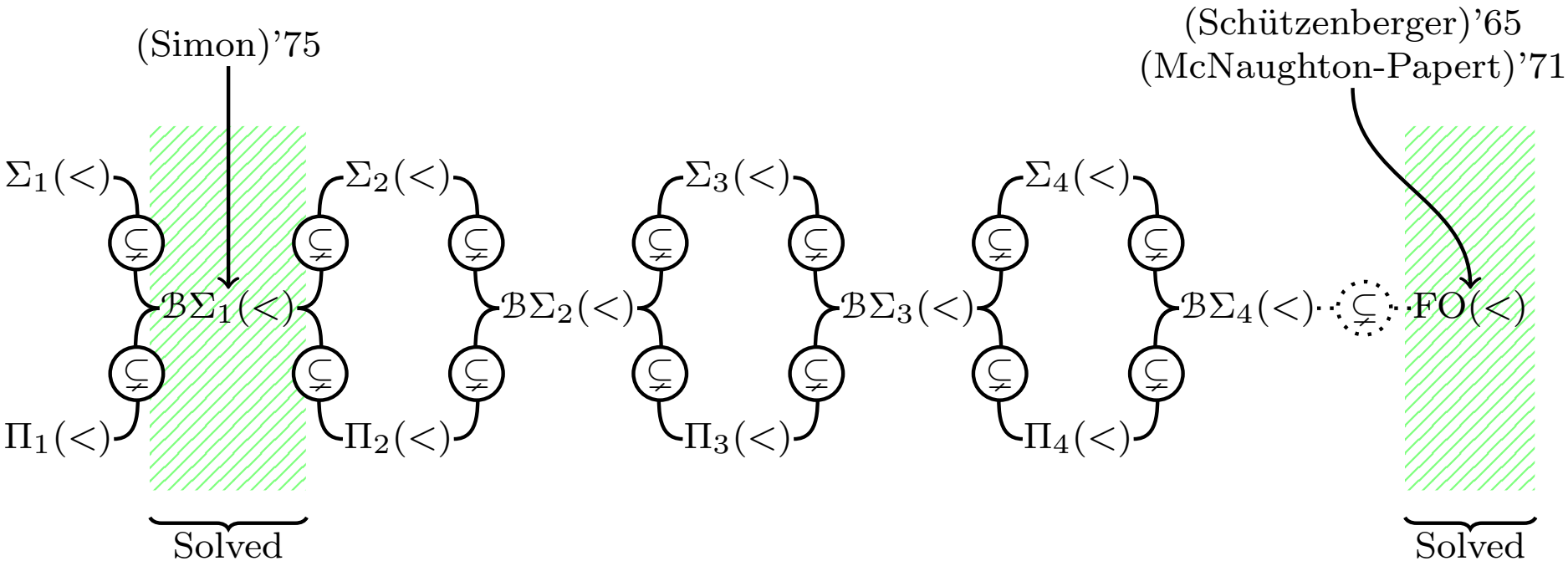


Objective: get membership algorithms for all levels.

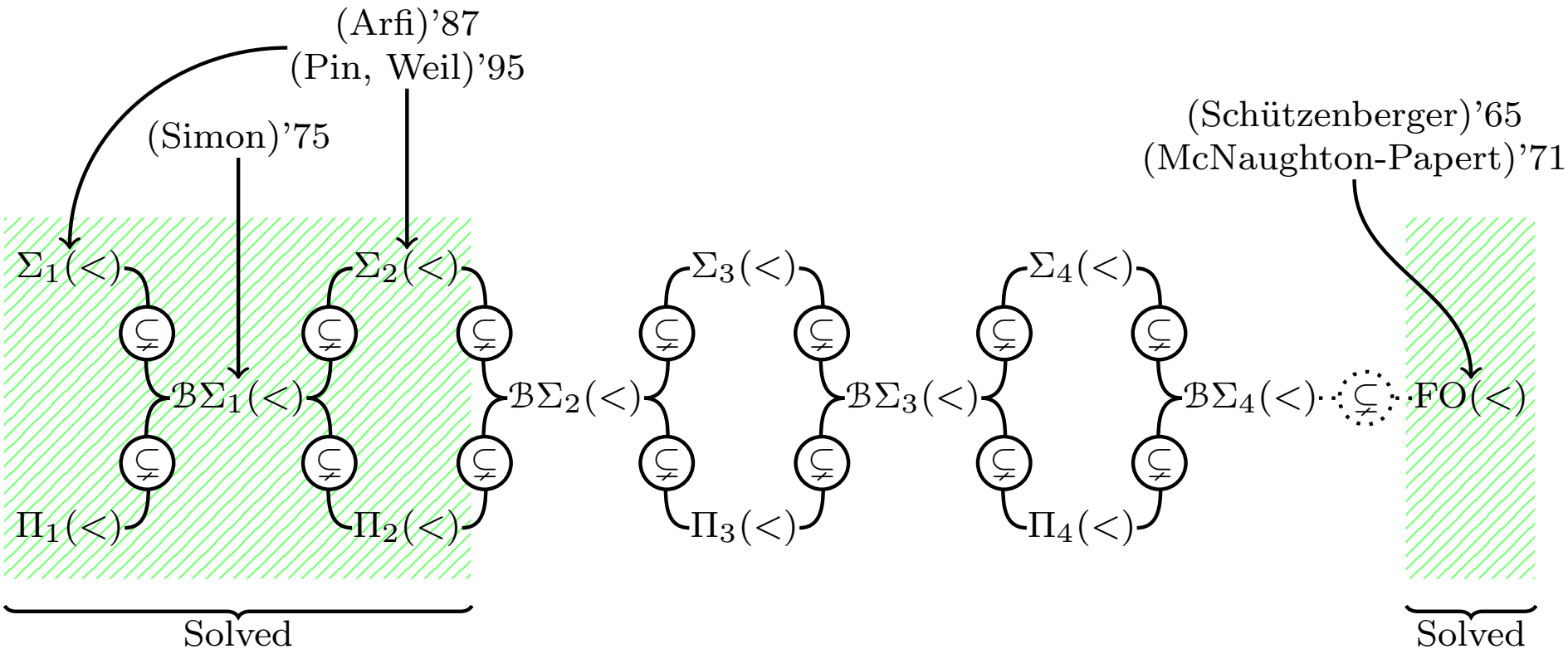
Quantifier Alternation: Membership state of the art



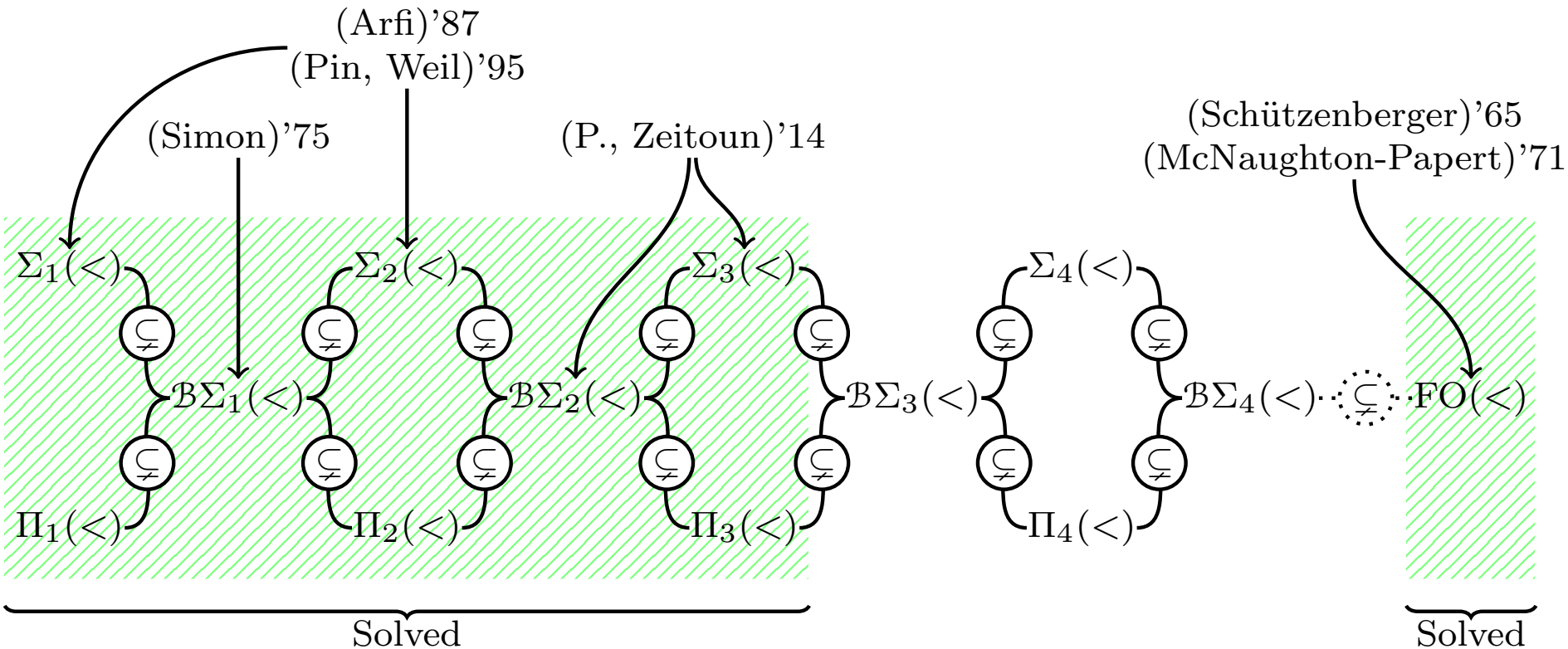
Quantifier Alternation: Membership state of the art



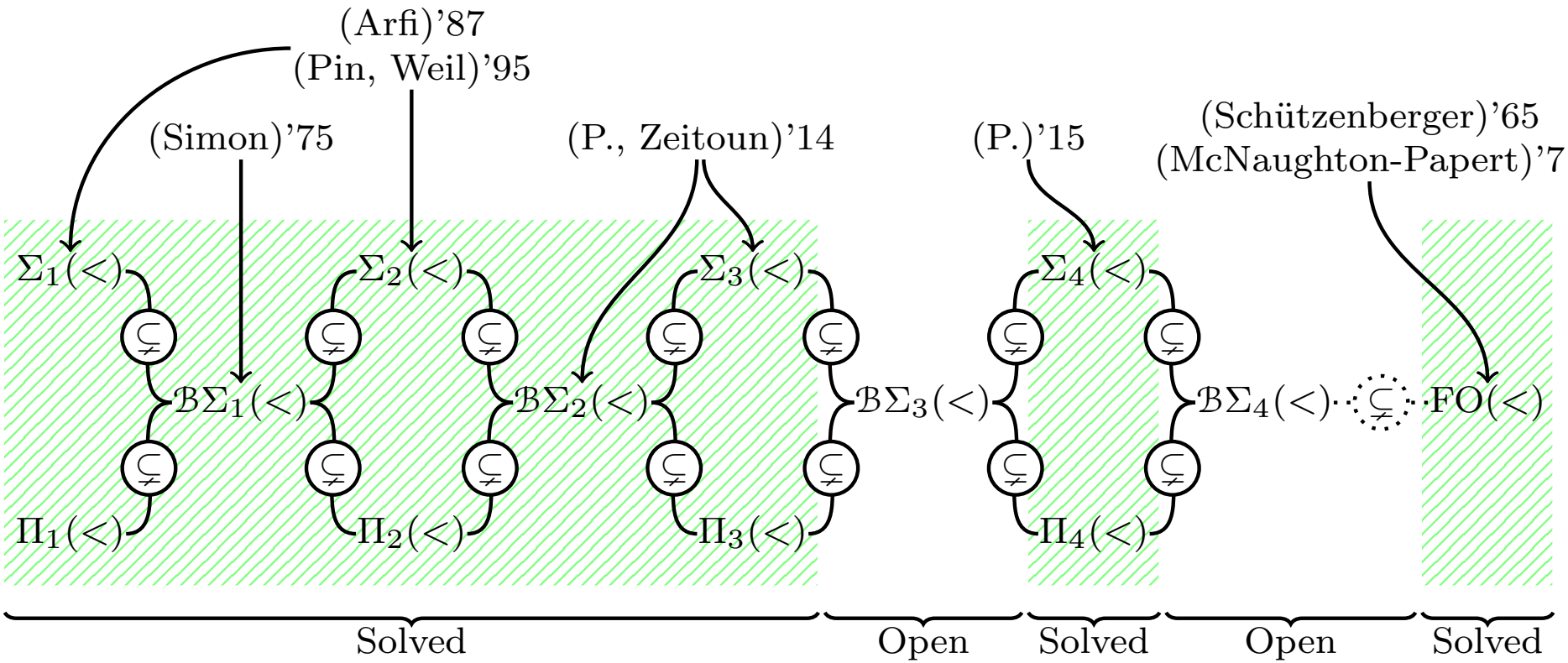
Quantifier Alternation: Membership state of the art



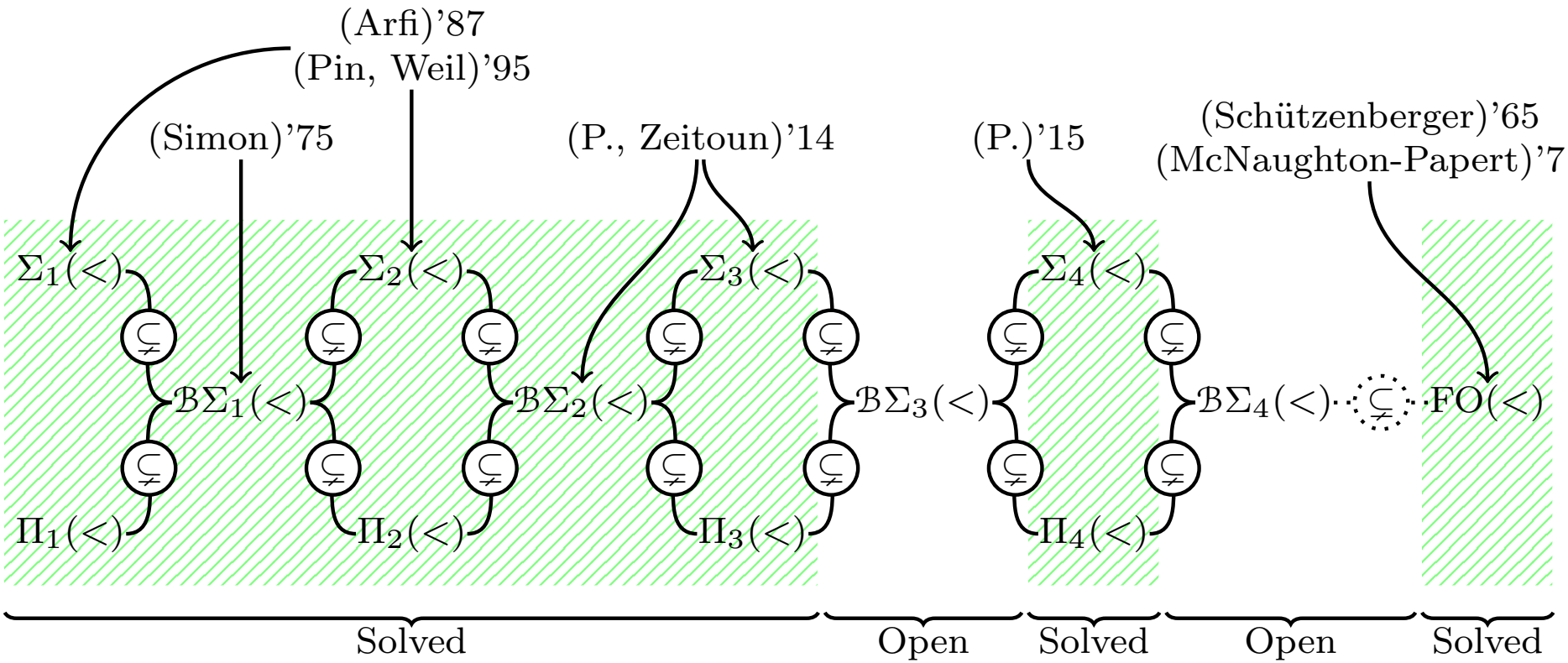
Quantifier Alternation: Membership state of the art



Quantifier Alternation: Membership state of the art

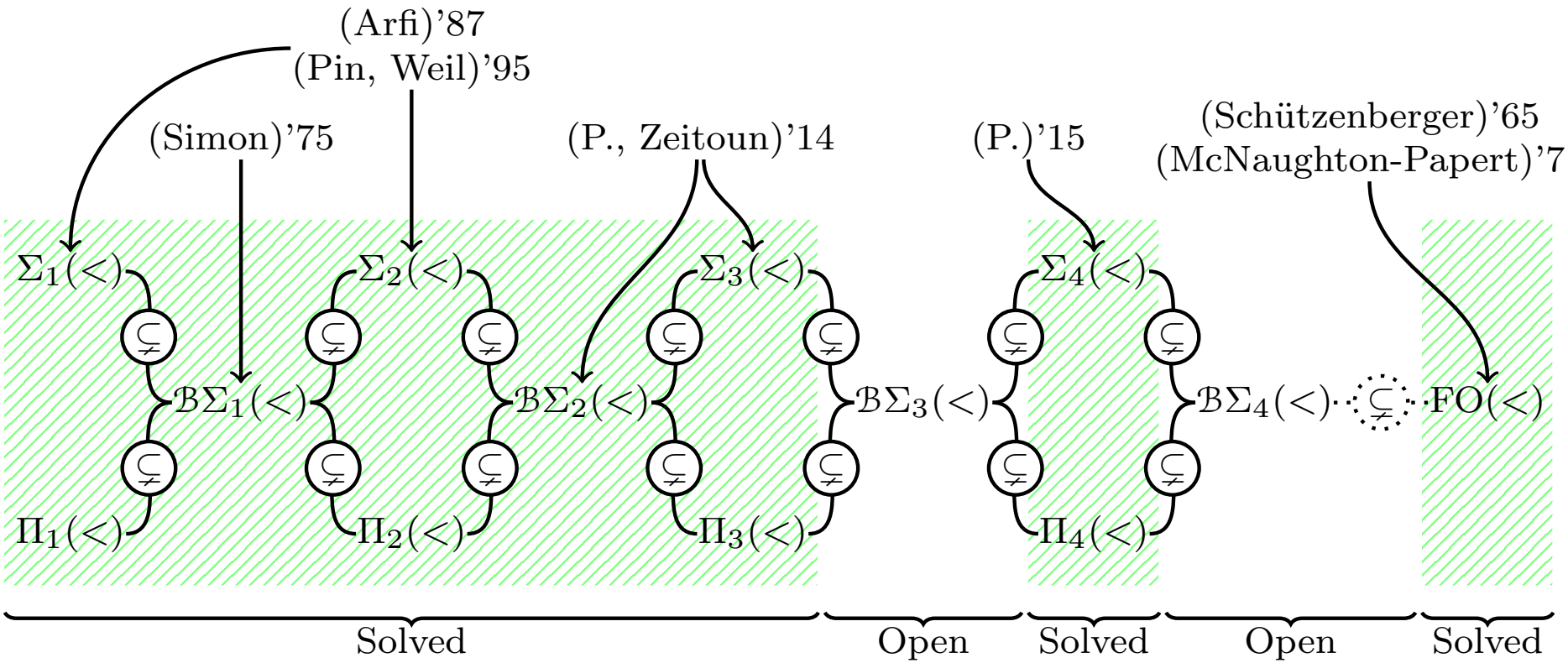


Quantifier Alternation: Membership state of the art



How are this results obtained ?

Quantifier Alternation: Membership state of the art



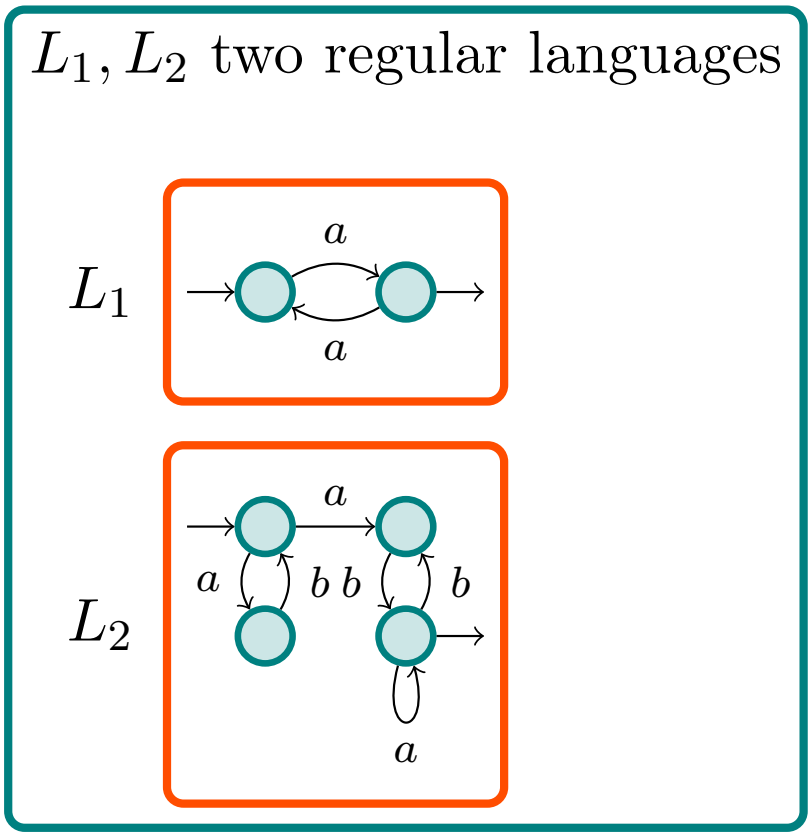
How are this results obtained ?

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

The Separation Problem

Definition

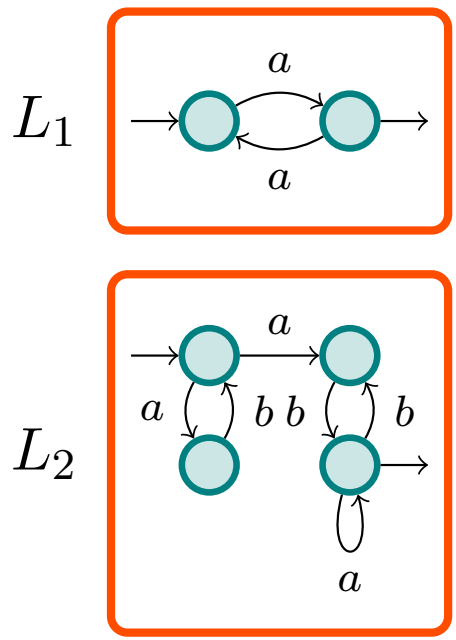
Given a class of languages $\mathcal{C}$ (for example a level in the hierarchy), decide the following problem:



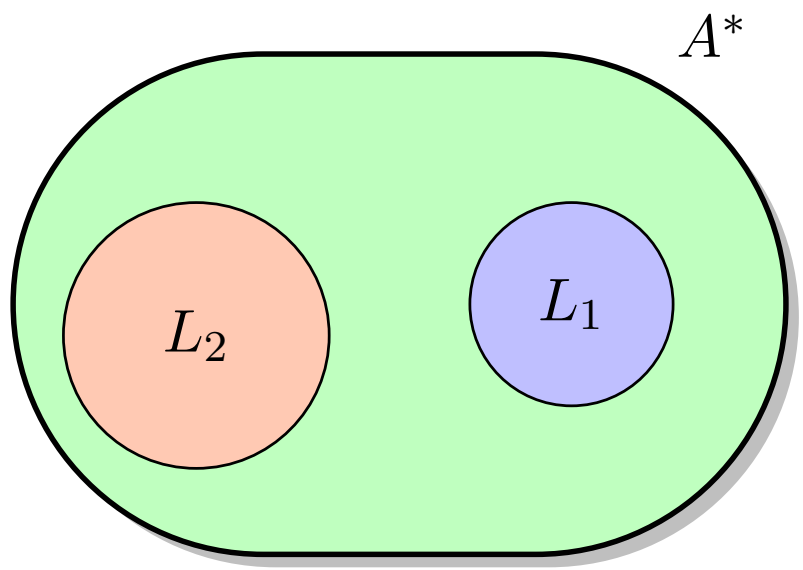
Definition

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



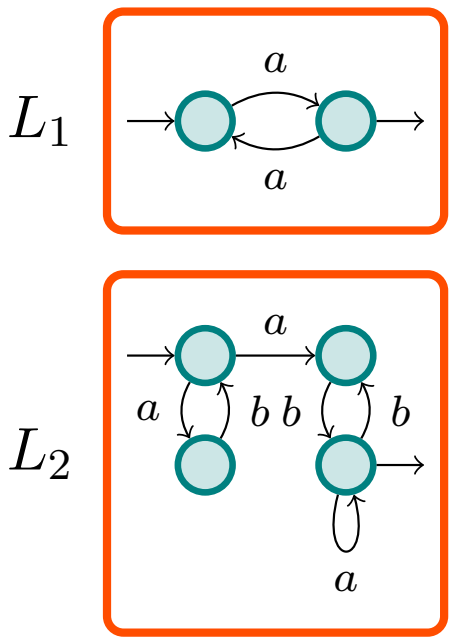
Can L_1 be separated from L_2 with a language of $\mathcal{C}$?



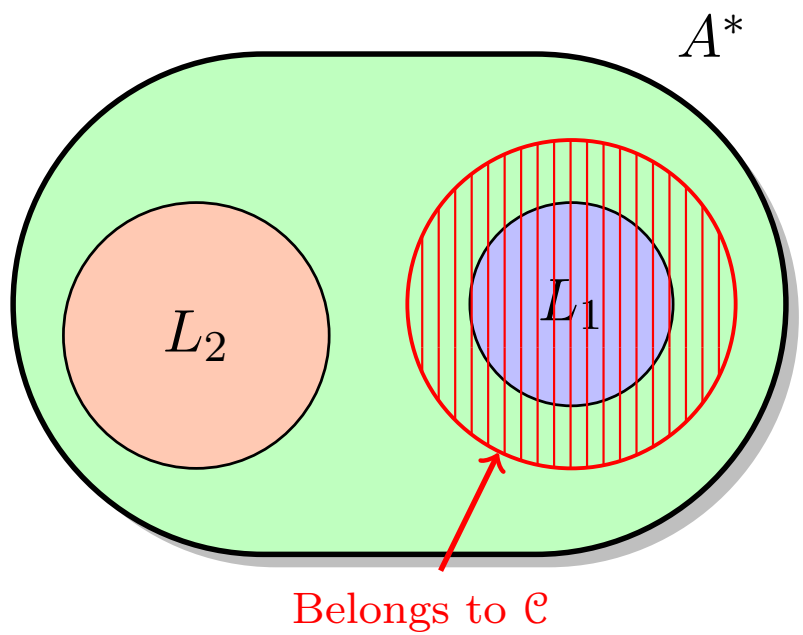
Definition

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



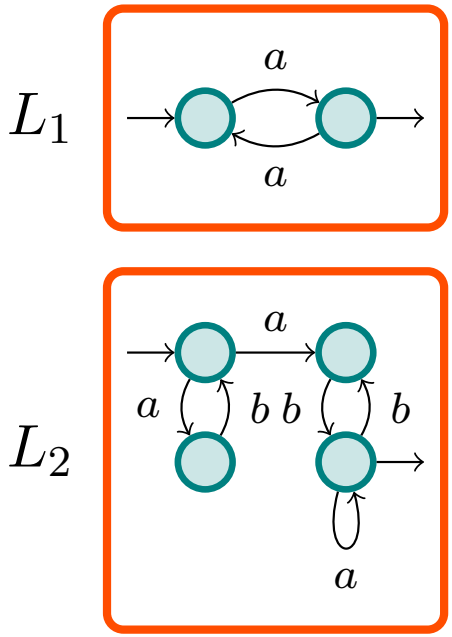
Can L_1 be separated from L_2 with a language of $\mathcal{C}$?



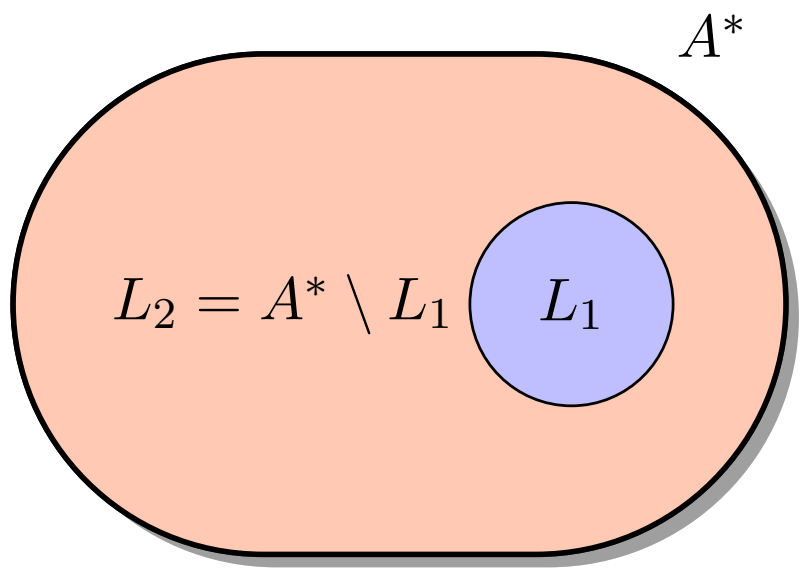
Definition

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



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

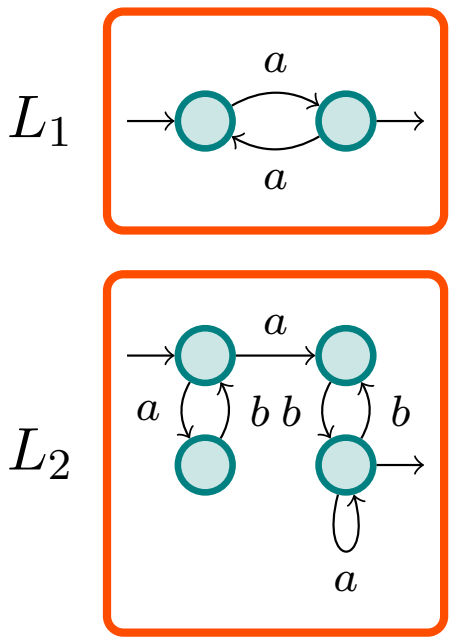


Membership can be formally reduced to separation

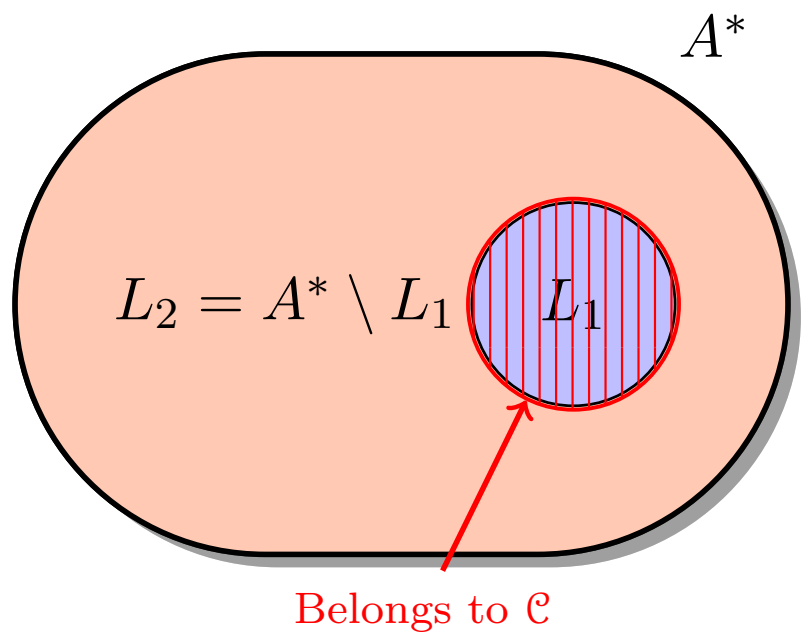
Definition

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

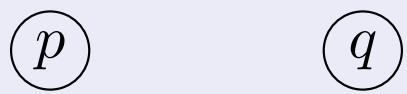
A regular language **is definable in Σ_n** iff its minimal automaton has **no pattern**:

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**:

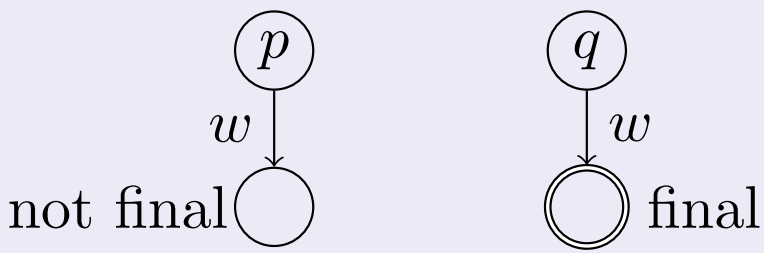


Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

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**:

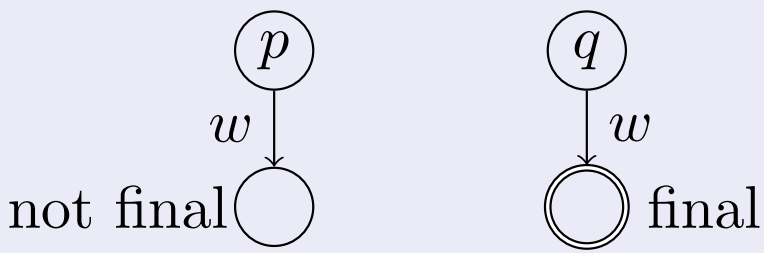


Transfer theorem: Σ_{n-1}-separation $\Rightarrow$ Σ_n-membership

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**:



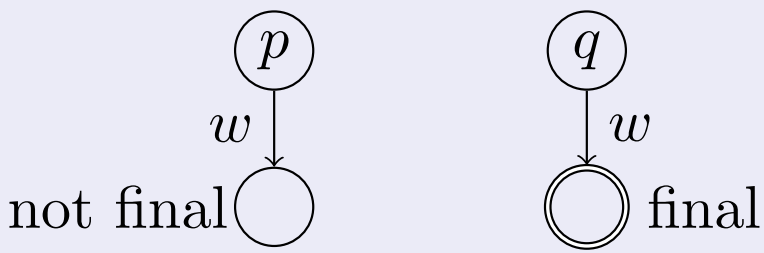
where $L_{p,q}$ is **not Σ_{n-1}-separable** from $L_{p,p} \cap L_{q,q}$

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 Σ_n** iff its minimal automaton has **no pattern**:



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

Corollary

Solving **Σ_{n-1}-separation** yields a solution for **Σ_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.

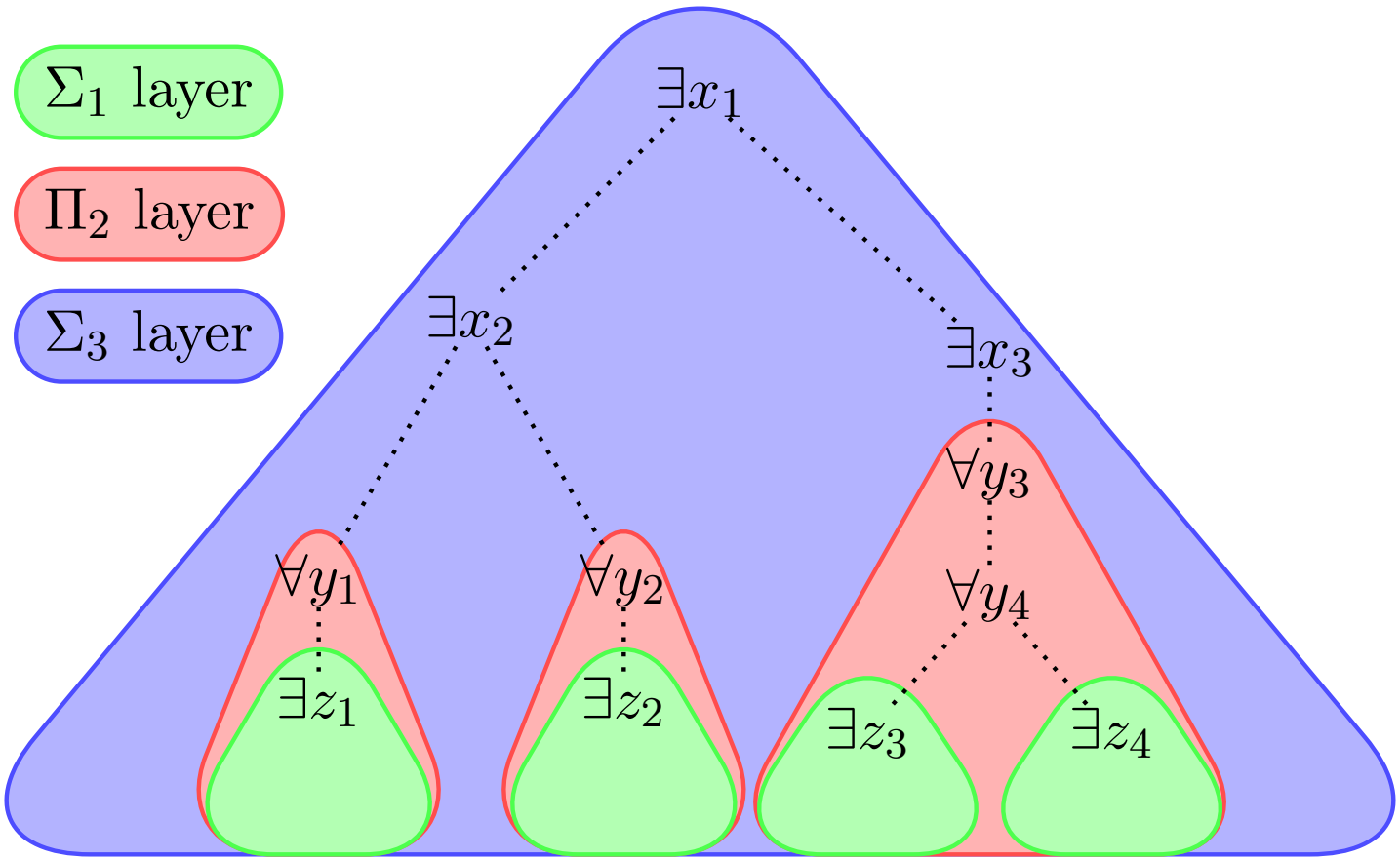
Let us explain why.

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

This is also the case for the transfer theorem.

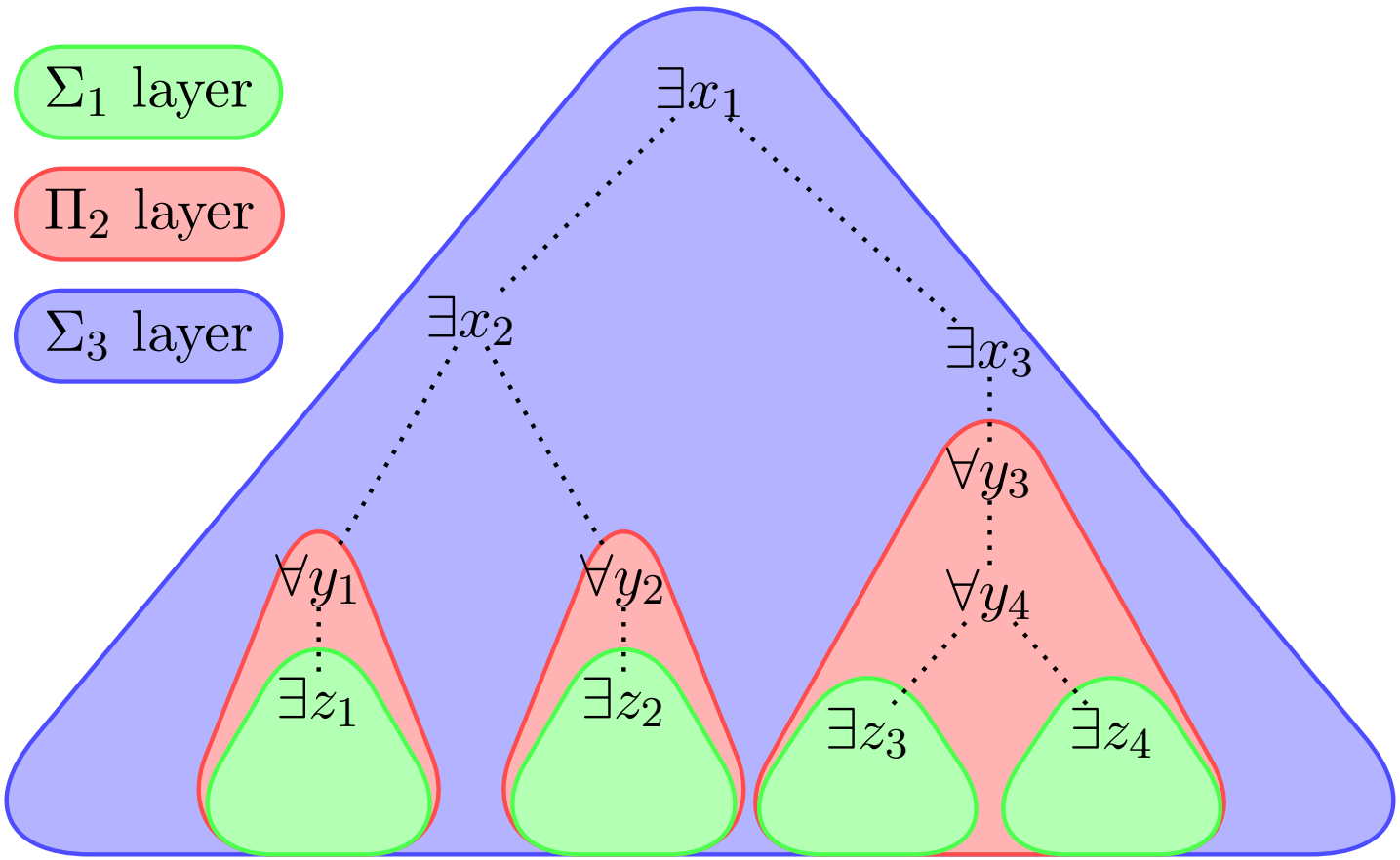
Construction of Σ_n sentences

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 $\mathcal{A}$.

Conclusion

- **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:

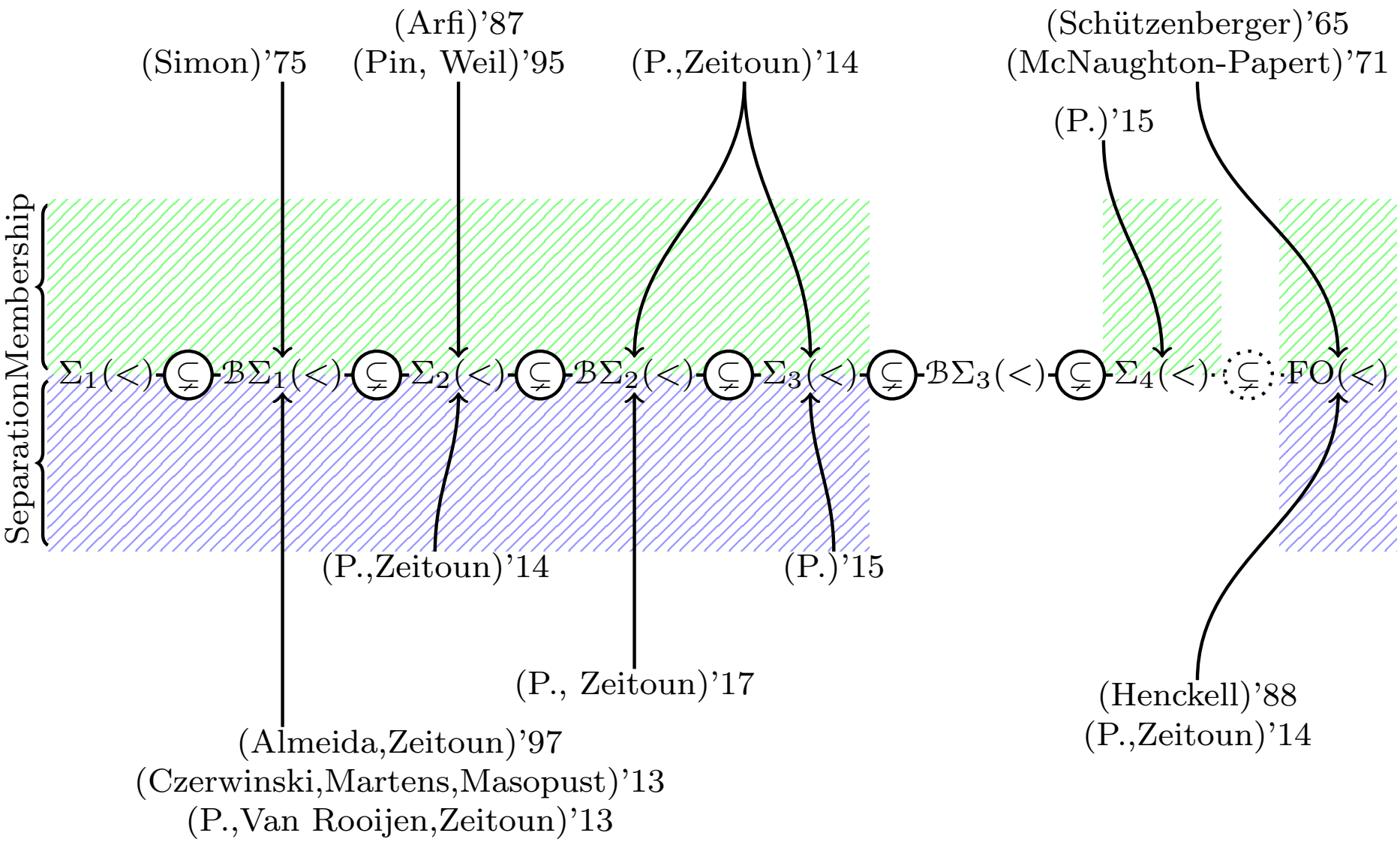
- All languages needed for the **layers below** Σ_{n-1} are Σ_{n-1}-separators of $L_{p,q}$ from $L_{p,p} \cap L_{q,q}$ for some states p, q of $\mathcal{A}$.
- One builds the topmost layer (Σ_n) from them by induction on $\mathcal{A}$.

Conclusion

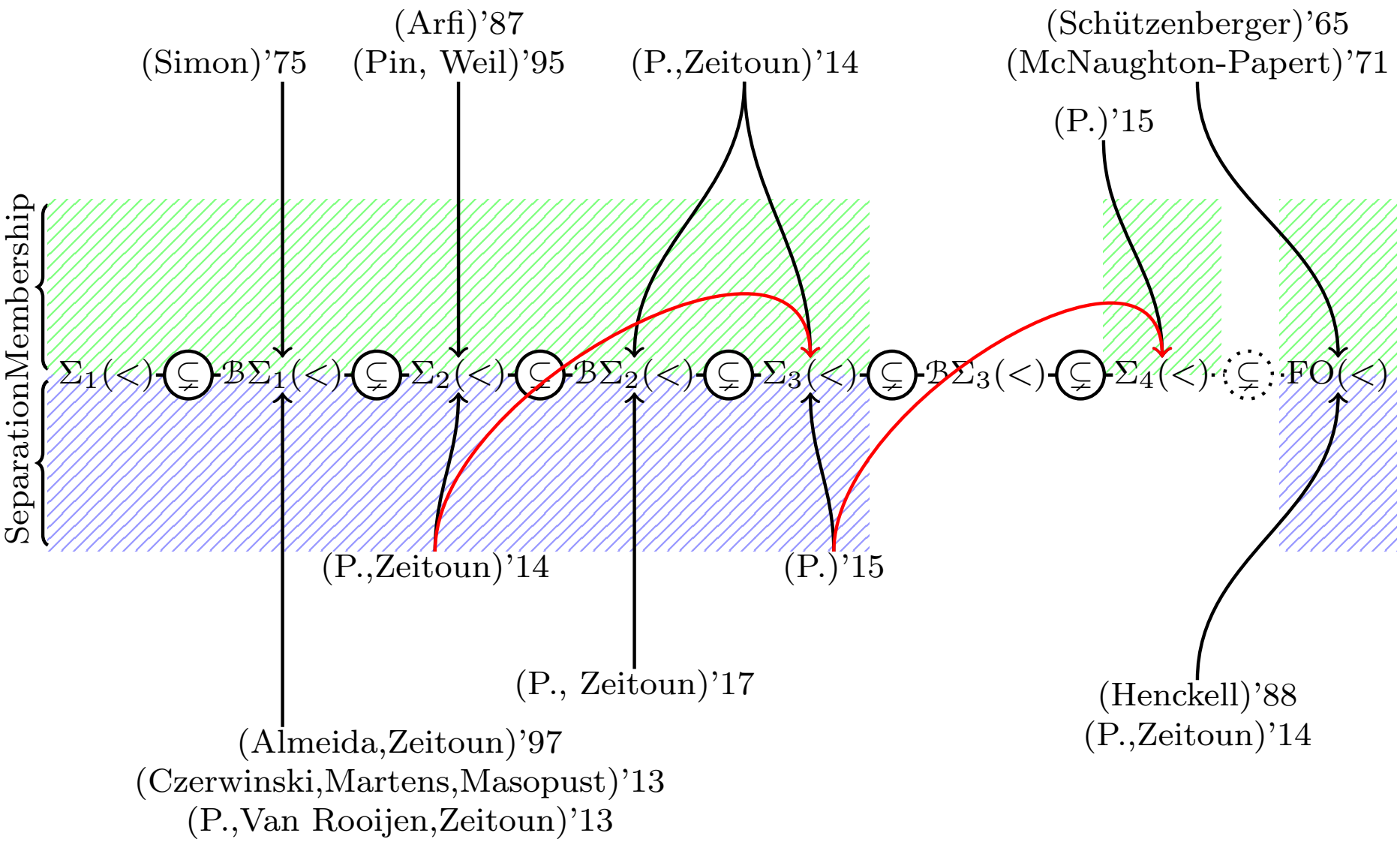
- **We already have the languages of the Σ_n layer in hand**: they are all recognized by $\mathcal{A}$.
- The lower layers are built by **approximating these languages** with Σ_{n-1}-separation.

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

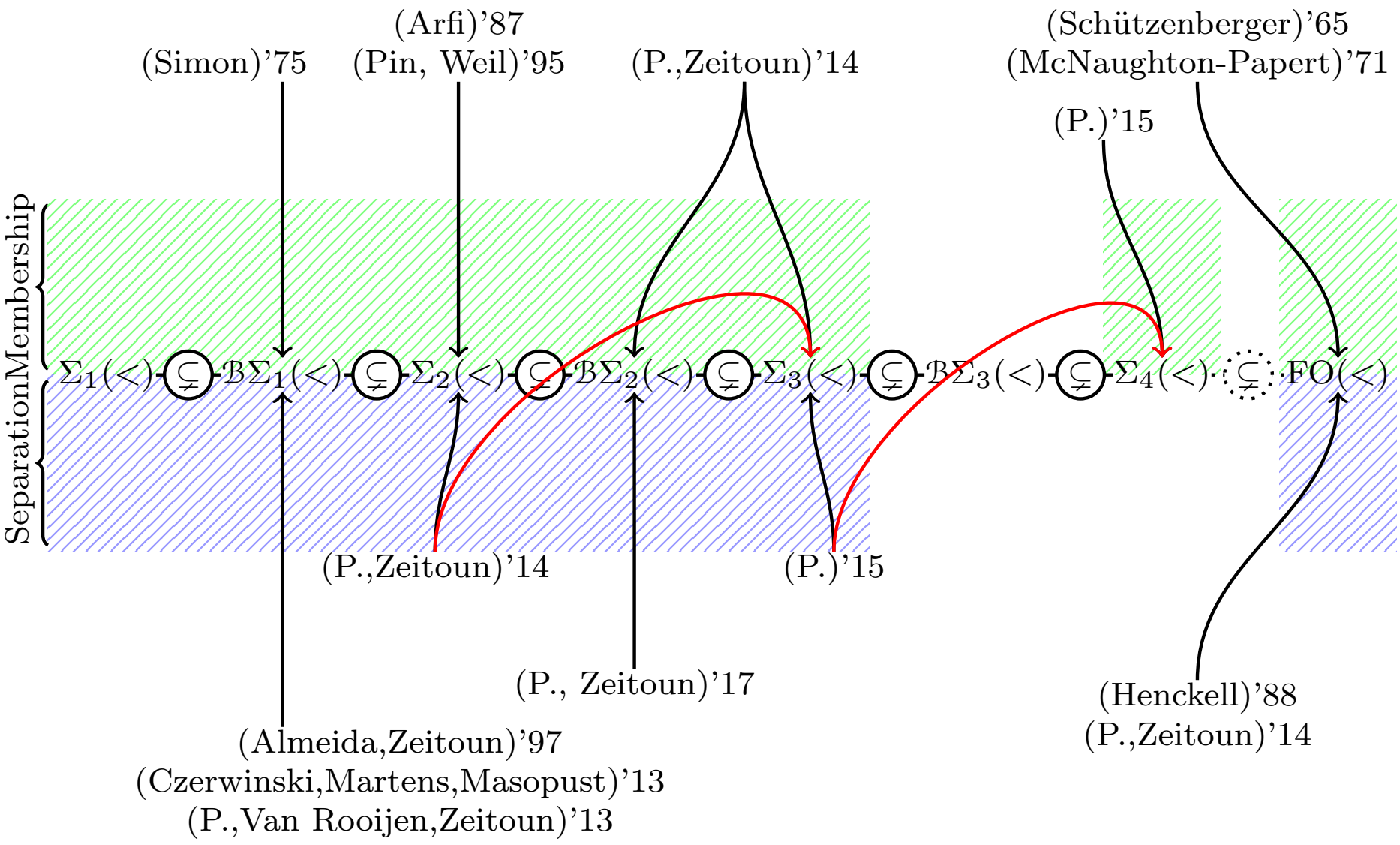
Current state of the art: Separation



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 $\mathrm{FO}(<)$ 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 $\bot$ 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 $\bot$ 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 $\bot$ 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:

$$\{\emptyset, A^*\} \quad 0 \xrightarrow{\text{Pol}} \tfrac{1}{2} \xrightarrow[\text{Bool}]{} 1 \xrightarrow{\text{Pol}} \tfrac{3}{2} \xrightarrow[\text{Bool}]{} 2 \xrightarrow{\text{Pol}} \tfrac{5}{2} \xrightarrow[\text{Bool}]{} 3 \cdots\cdots$$

Polynomial closure

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

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

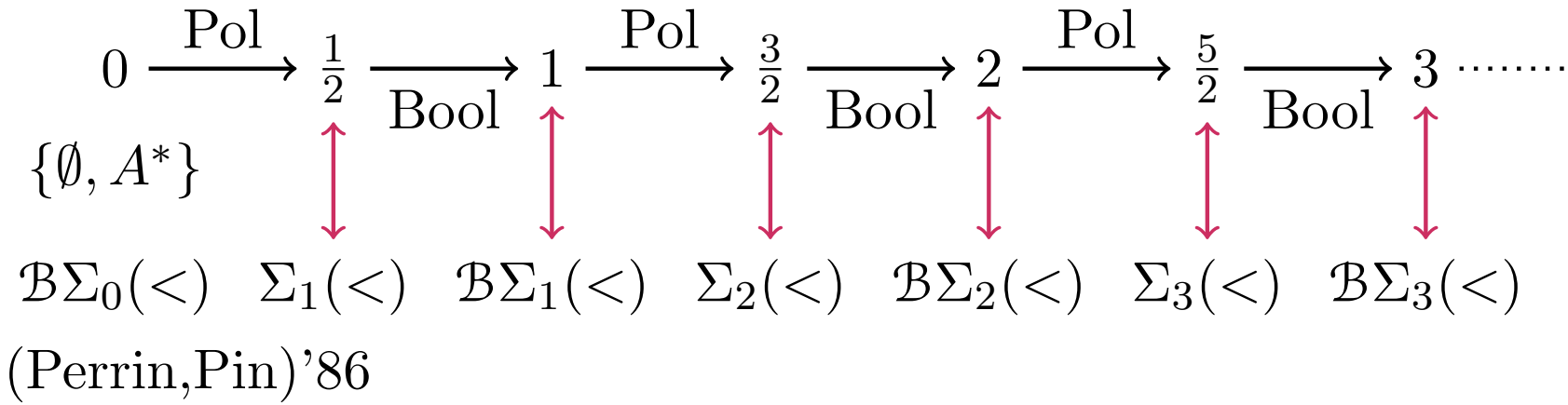
Boolean closure

$Bool(\mathcal{C})$ 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:



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

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

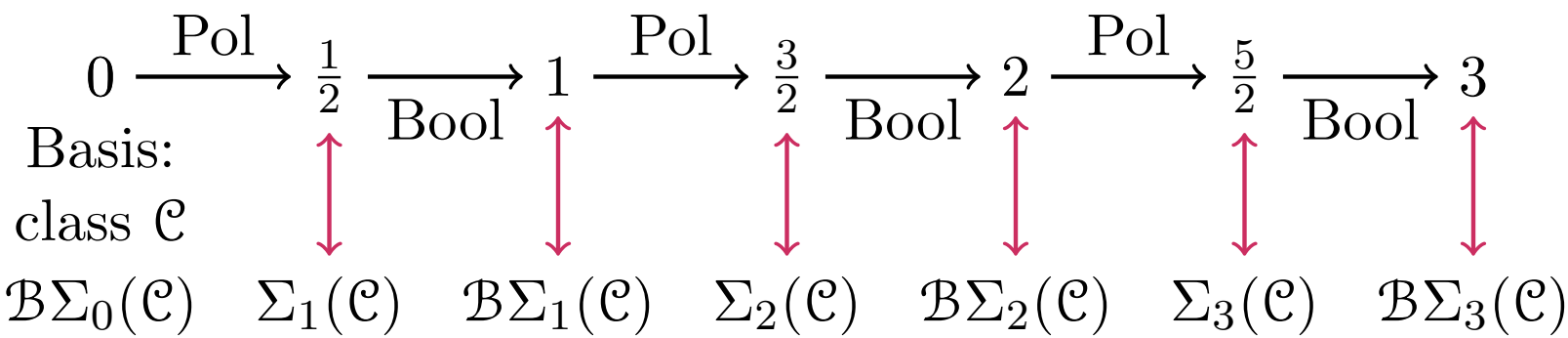
For each $L \in \mathcal{C}$, we add **four predicates**:

- **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 FO($\mathcal{C}$).

The logical connection is generic (2)

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



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 $\bot$

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 $\bot$

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

- $I_{A^*}(x, y)$ 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 $max(x)$.

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 $\bot$

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.
- 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 $\bot$

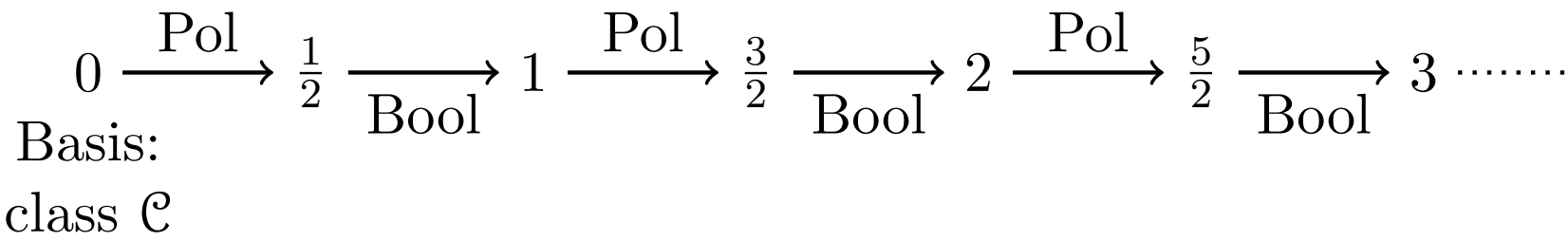
The dot-depth hierarchy (Brzozowski,Cohen)'71

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

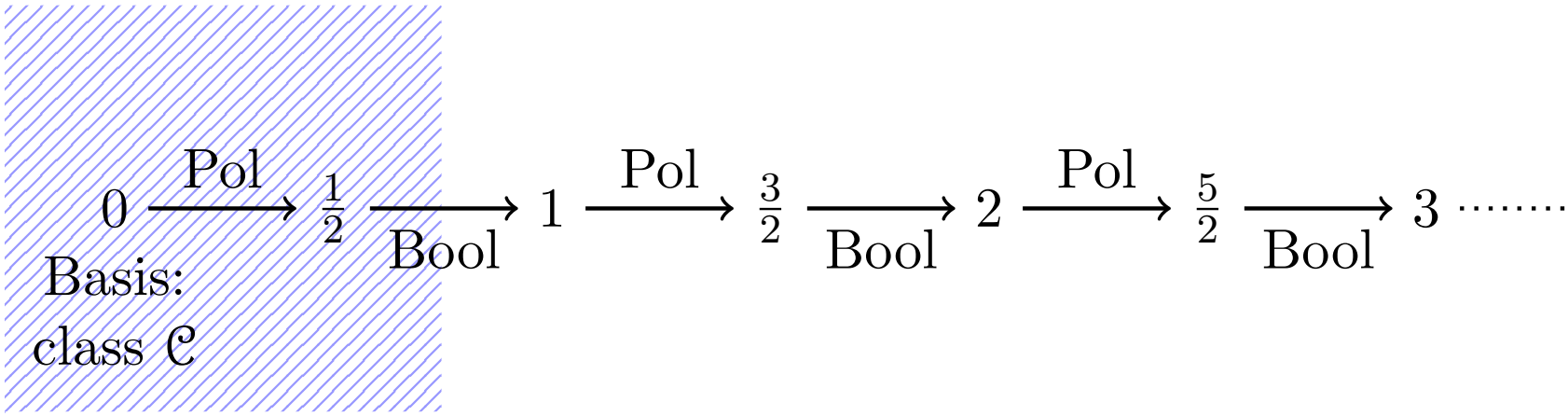
Generic separation results (1)



Generic Separation Results (P.,Zeitoun)'17

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

Generic separation results (1)

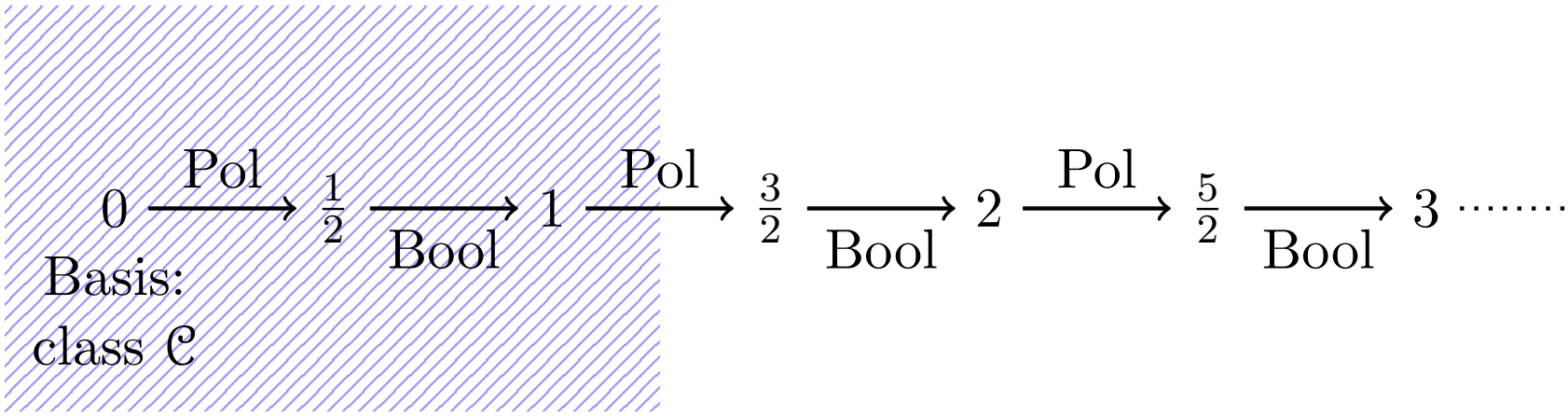


Generic Separation Results (P.,Zeitoun)'17

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

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

Generic separation results (1)

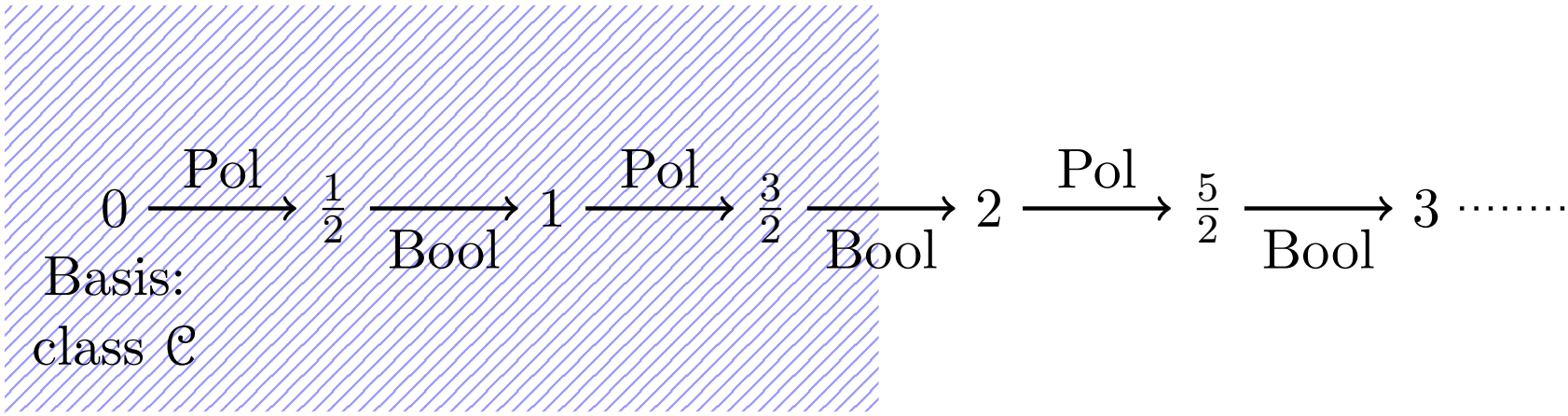


Generic Separation Results (P.,Zeitoun)'17

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)

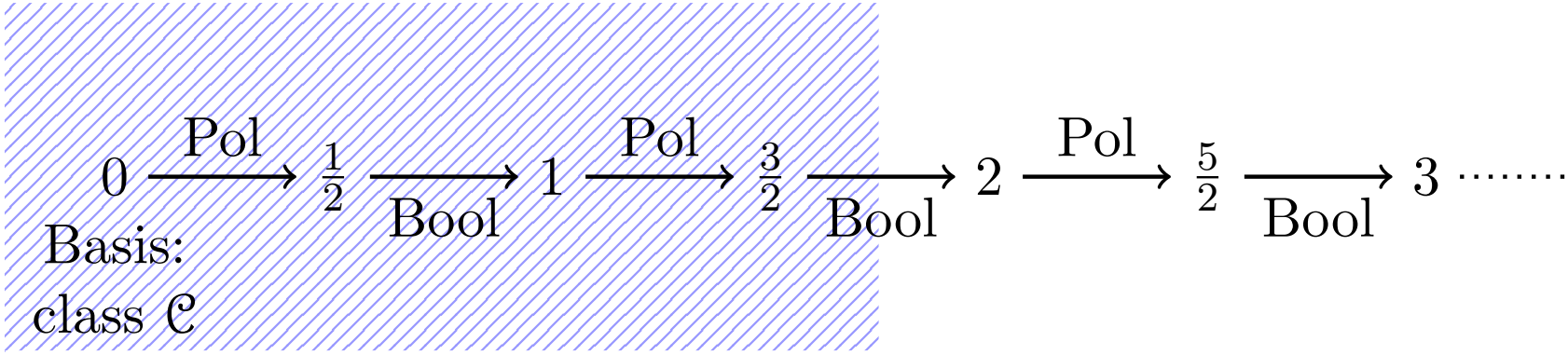


Generic Separation Results (P.,Zeitoun)'17

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

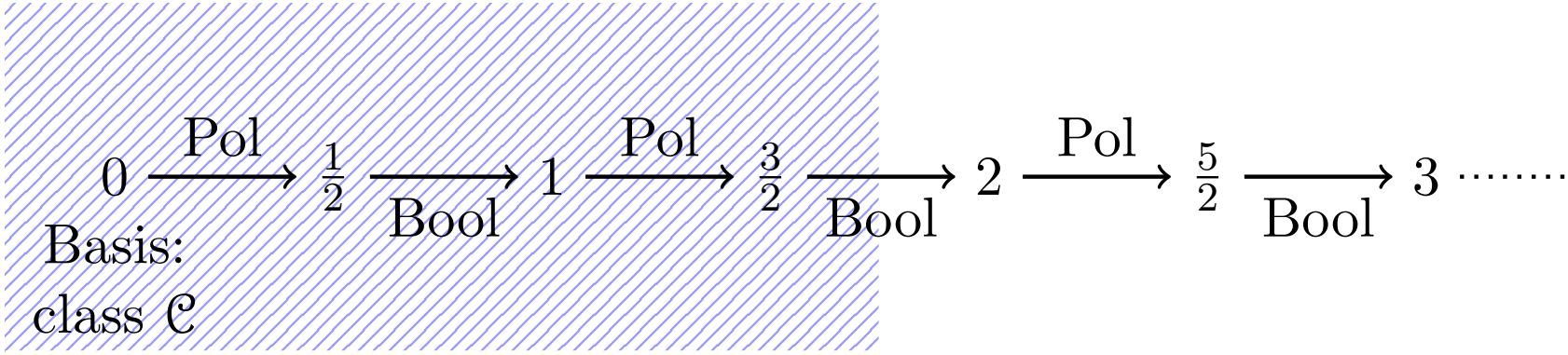
Generic separation results (2)



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)



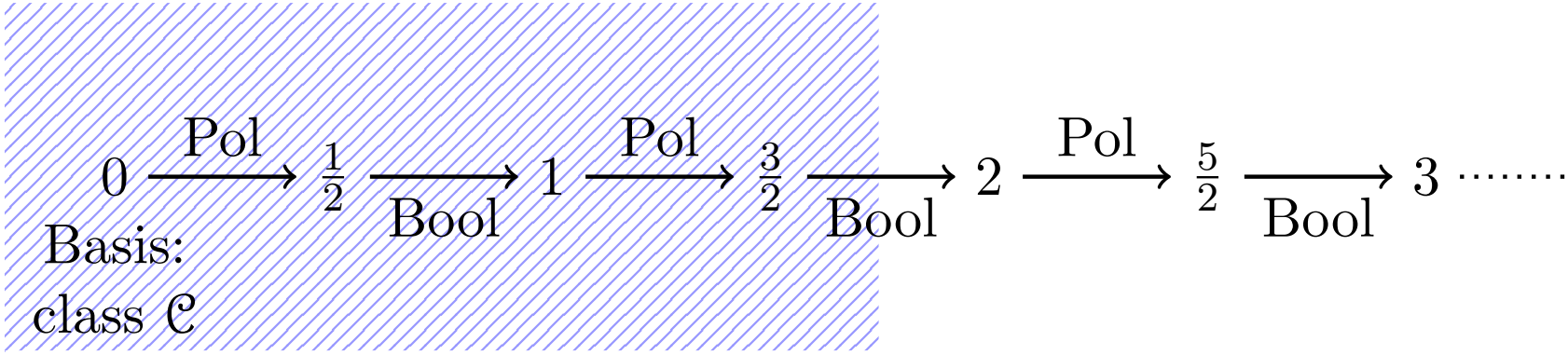
Advantages:

- 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)



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 to us ! Refund !!!!!!

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

The almighty alphabet argument

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

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

The almighty alphabet argument

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

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

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 $a \in A$

The almighty alphabet argument

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

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

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 $a \in A$

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.

We now focus on $BPol(\mathcal{C})$**-separation**.

Separation for $BPol(\mathcal{C})$ when $\mathcal{C}$ is finite

$BPol(\mathcal{C})$-separation: Three main steps

The algorithm is based on three main steps.

$BPol(\mathcal{C})$-separation: Three main steps

The algorithm is based on three main steps.

- All steps are very involved (and hide sub-steps).
- We explain their purpose at a high level.

$BPol(\mathcal{C})$-separation: Three main steps

The algorithm is based on three main steps.

- All steps are very involved (and hide sub-steps).
- We explain their purpose at a high level.

Warning

Many issues are hidden to simplify the presentation.

First step - preliminary remark

Our two closure operations have different properties:

- $Pol(\mathcal{D})$ is closed under union, intersection and **marked concatenation**.

First step - preliminary remark

Our two closure operations have different properties:

- $Pol(\mathcal{D})$ is closed under union, intersection and **marked concatenation**.
- $Bool(\mathcal{D})$ is closed under all Boolean operations but **not marked concatenation**.

First step - preliminary remark

Our two closure operations have different properties:

- $Pol(\mathcal{D})$ is closed under union, intersection and **marked concatenation**.
- $Bool(\mathcal{D})$ is closed under all Boolean operations but **not marked concatenation**.

Our **techniques rely heavily on concatenation**:
$\Rightarrow$ we like $Pol(\mathcal{C})$ and hate $BPol(\mathcal{C})$.

First step - preliminary remark

Our two closure operations have different properties:

- $Pol(\mathcal{D})$ is closed under union, intersection and **marked concatenation**.
- $Bool(\mathcal{D})$ is closed under all Boolean operations but **not marked concatenation**.

Our **techniques rely heavily on concatenation**:
$\Rightarrow$ we like $Pol(\mathcal{C})$ and hate $BPol(\mathcal{C})$.

Consequence

Even if our goal is $BPol(\mathcal{C})$-separation, we prefer working with $Pol(\mathcal{C})$.

$BPol(\mathcal{C})$-separation: Three main steps (1)

Getting rid of Boolean closure

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.
- Given such a tuple $(L_1, L_2, \ldots, L_n)$ and a language K, we write,

$$(L_1, L_2, \ldots, L_n) \cap K = (L_1 \cap K, L_2 \cap K, \ldots, L_n \cap K)$$

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.
- Given such a tuple $(L_1, L_2, \ldots, L_n)$ and a language K, we write,

$$(L_1, L_2, \ldots, L_n) \cap K = (L_1 \cap K, L_2 \cap K, \ldots, L_n \cap K)$$

$\mathcal{D}$-separability: inductive definition

Let $\mathcal{D}$ be class of languages. $(L_1, \ldots, L_n)$ is **$\mathcal{D}$-separable** iff:

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.
- Given such a tuple $(L_1, L_2, \ldots, L_n)$ and a language K, we write,

$$(L_1, L_2, \ldots, L_n) \cap K = (L_1 \cap K, L_2 \cap K, \ldots, L_n \cap K)$$

$\mathcal{D}$-separability: inductive definition

Let $\mathcal{D}$ be class of languages. $(L_1, \ldots, L_n)$ is **$\mathcal{D}$-separable** iff:

- $n = 1$ and $L_1 = \emptyset$ or,

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.
- Given such a tuple $(L_1, L_2, \ldots, L_n)$ and a language K, we write,

$$(L_1, L_2, \ldots, L_n) \cap K = (L_1 \cap K, L_2 \cap K, \ldots, L_n \cap K)$$

$\mathcal{D}$-separability: inductive definition

Let $\mathcal{D}$ be class of languages. $(L_1, \ldots, L_n)$ is **$\mathcal{D}$-separable** iff:

- $n = 1$ and $L_1 = \emptyset$ or,
- $n \geq 2$ and there exists $K \in \mathcal{D}$ such that,

$$L_1 \subseteq K \quad \text{and} \quad (L_2, \ldots, L_n) \cap K \text{ is } \mathcal{D}\text{-separable}$$

$Bool(\mathcal{D})$-separation $\Rightarrow$ generalized separation for $\mathcal{D}$

- We generalize the notion of separability to **tuples of languages**.
- Given such a tuple $(L_1, L_2, \ldots, L_n)$ and a language K, we write,

$$(L_1, L_2, \ldots, L_n) \cap K = (L_1 \cap K, L_2 \cap K, \ldots, L_n \cap K)$$

$\mathcal{D}$-separability: inductive definition

Let $\mathcal{D}$ be class of languages. $(L_1, \ldots, L_n)$ is **$\mathcal{D}$-separable** iff:

- $n = 1$ and $L_1 = \emptyset$ or,
- $n \geq 2$ and there exists $K \in \mathcal{D}$ such that,

$$L_1 \subseteq K \quad \text{and} \quad (L_2, \ldots, L_n) \cap K \text{ is } \mathcal{D}\text{-separable}$$

Remark

When $n = 2$, we recover the classical notion.

Boolean closure theorem

For any two languages L_1, L_2 and $k \geq 1$, we denote by $(L_1, L_2)^k$, the tuple:

$$\underbrace{(L_1, L_2, \ldots, L_1, L_2)}_{\text{length } 2k}$$

Boolean closure theorem

For any two languages L_1, L_2 and $k \geq 1$, we denote by $(L_1, L_2)^k$, the tuple:

$$\underbrace{(L_1, L_2, \ldots, L_1, L_2)}_{\text{length } 2k}$$

$Bool(\mathcal{D})$-separation theorem

Given a lattice $\mathcal{D}$ and two languages L_1, L_2, t.f.a.e.,

1. L_1 is $Bool(\mathcal{D})$-separable from L_2.
2. There exists $k \geq 1$ s.t. $(L_1, L_2)^k$ is $\mathcal{D}$-separable.

Boolean closure theorem

For any two languages L_1, L_2 and $k \geq 1$, we denote by $(L_1, L_2)^k$, the tuple:

$$\underbrace{(L_1, L_2, \ldots, L_1, L_2)}_{\text{length } 2k}$$

$Bool(\mathcal{D})$-separation theorem

Given a lattice $\mathcal{D}$ and two languages L_1, L_2, t.f.a.e.,

1. L_1 is $Bool(\mathcal{D})$-separable from L_2.
2. There exists $k \geq 1$ s.t. $(L_1, L_2)^k$ is $\mathcal{D}$-separable.

Mission accomplished !

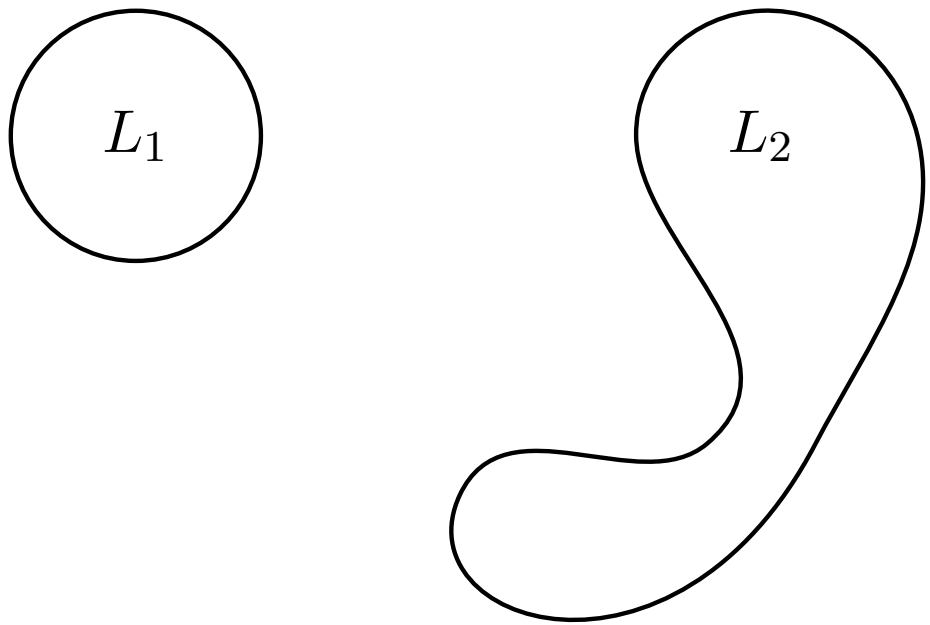
When $\mathcal{D} = Pol(\mathcal{C})$, this is a reduction from $BPol(\mathcal{C})$-separation to a problem for $Pol(\mathcal{C})$.

Boolean closure theorem: proof

We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2

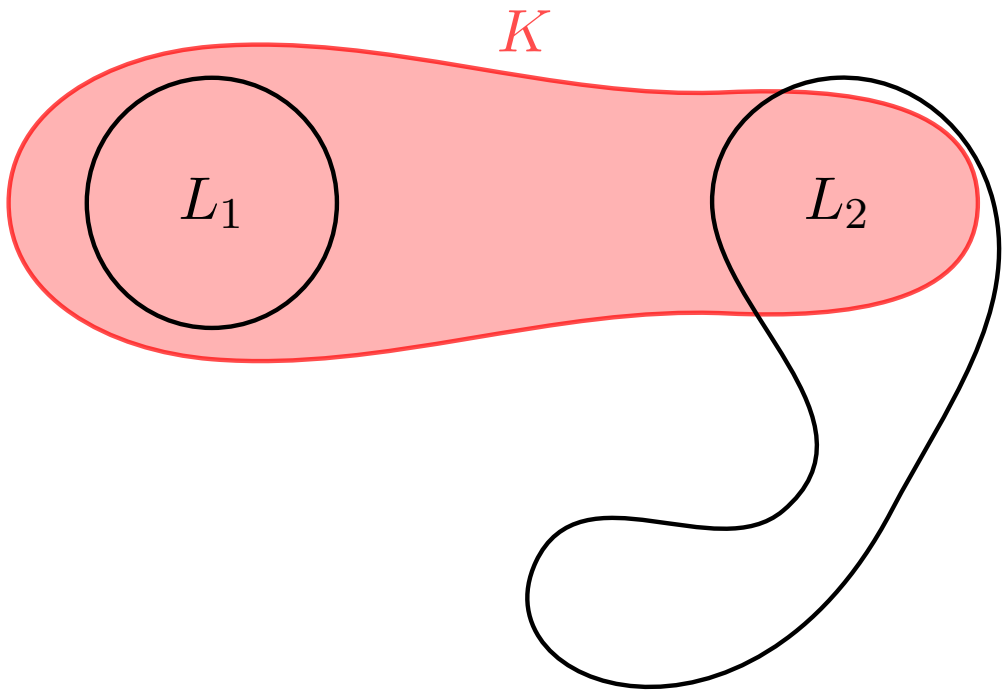
Boolean closure theorem: proof

We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2



Boolean closure theorem: proof

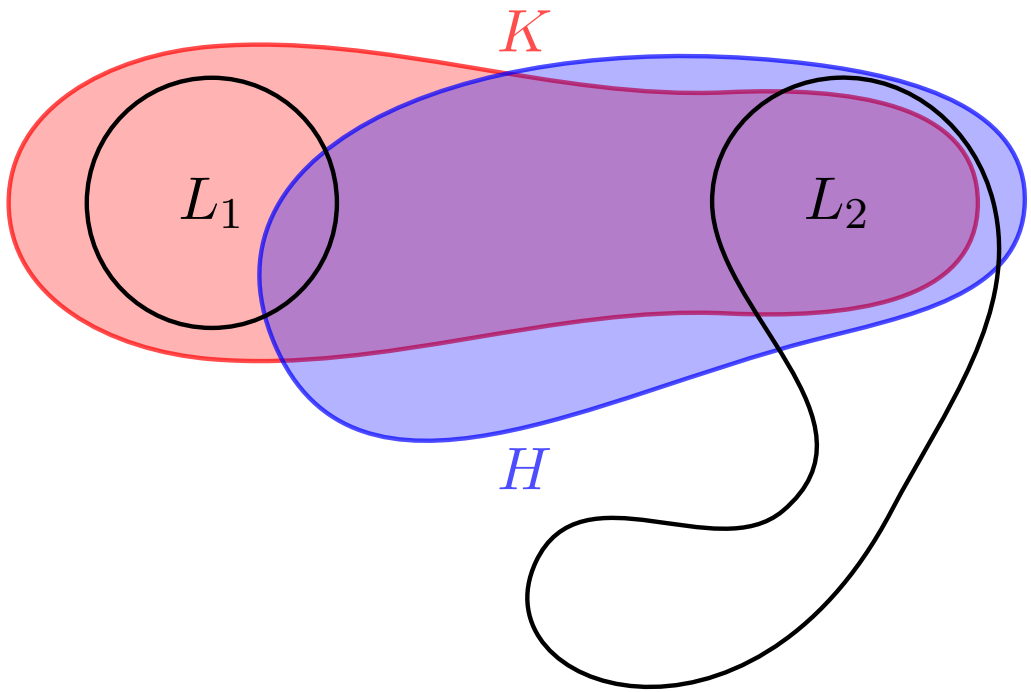
We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2



- $(L_2, (L_1, L_2)^{k-1}) \cap K$ is $\mathcal{D}$-separable

Boolean closure theorem: proof

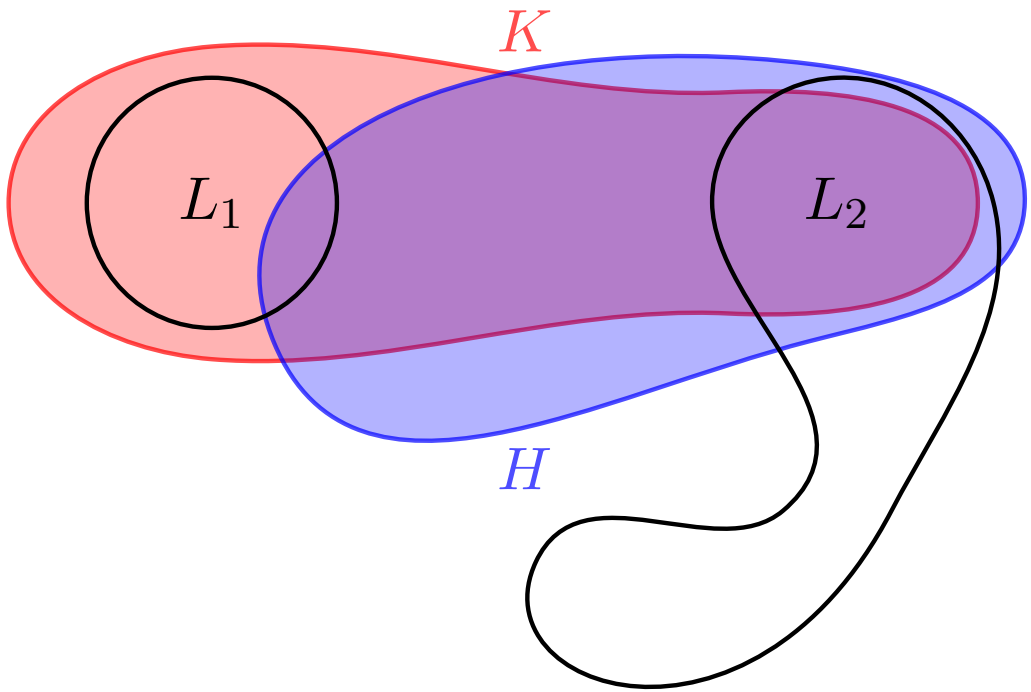
We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2



- $(L_2, (L_1, L_2)^{k-1}) \cap K$ is $\mathcal{D}$-separable
- $(L_1, L_2)^{k-1} \cap K \cap H$ is $\mathcal{D}$-separable

Boolean closure theorem: proof

We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2

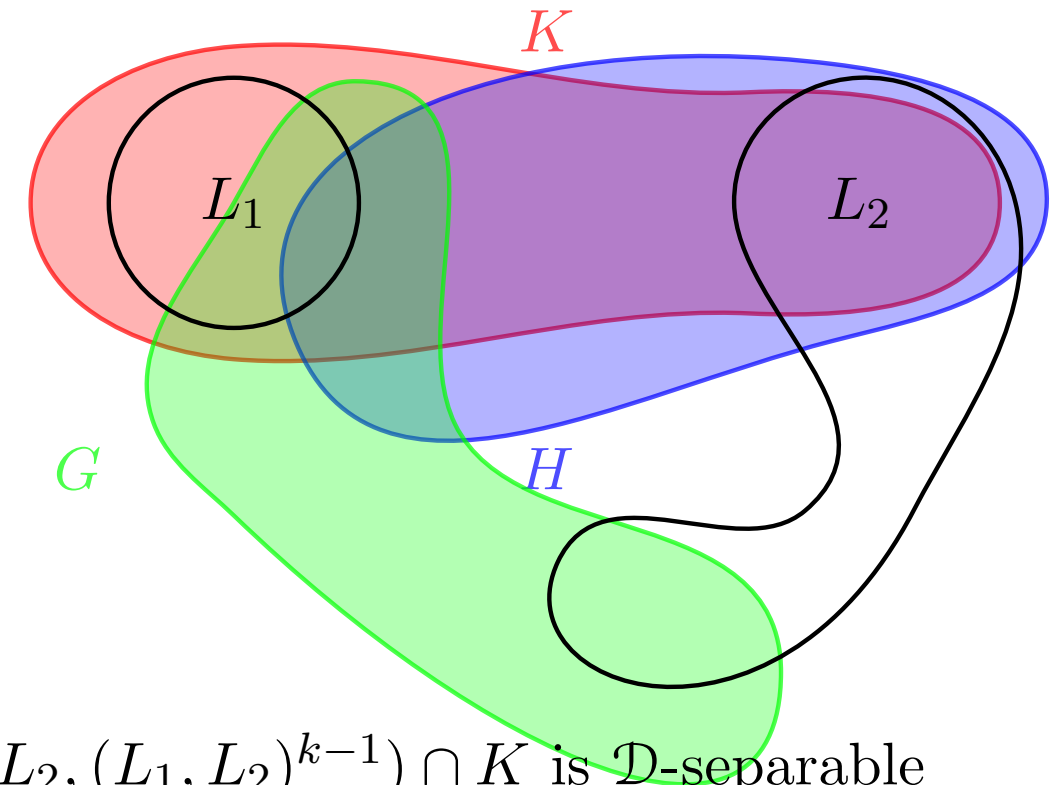


- $(L_2, (L_1, L_2)^{k-1}) \cap K$ is $\mathcal{D}$-separable
- $(L_1, L_2)^{k-1} \cap K \cap H$ is $\mathcal{D}$-separable

Induction $\Rightarrow$ $G \in Bool(\mathcal{D})$ separates $L_1 \cap K \cap H$ from $L_2 \cap K \cap H$

Boolean closure theorem: proof

We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2

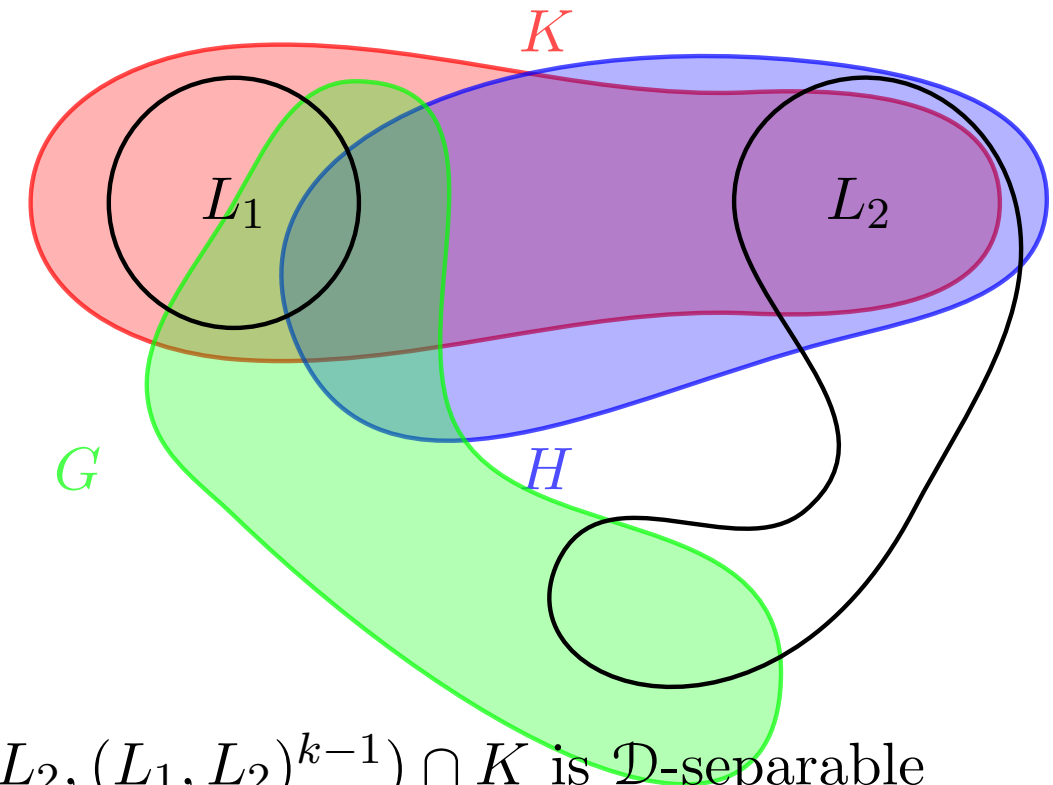


- $(L_2, (L_1, L_2)^{k-1}) \cap K$ is $\mathcal{D}$-separable
- $(L_1, L_2)^{k-1} \cap K \cap H$ is $\mathcal{D}$-separable

Induction $\Rightarrow$ $G \in Bool(\mathcal{D})$ separates $L_1 \cap K \cap H$ from $L_2 \cap K \cap H$

Boolean closure theorem: proof

We prove 2. $\Rightarrow$ 1. We have $k \geq 1$ such that $(L_1, L_2)^k$ is $\mathcal{D}$-separable.
We show by **induction on k** that L_1 is $Bool(\mathcal{D})$-separable from L_2



- $(L_2, (L_1, L_2)^{k-1}) \cap K$ is $\mathcal{D}$-separable
- $(L_1, L_2)^{k-1} \cap K \cap H$ is $\mathcal{D}$-separable

Induction $\Rightarrow$ $G \in Bool(\mathcal{D})$ separates $L_1 \cap K \cap H$ from $L_2 \cap K \cap H$

$$(G \cap K) \cup (K \setminus H) \in Bool(\mathcal{D}) \text{ separates } L_1 \text{ from } L_2$$

$BPol(\mathcal{C})$-separation: Three main steps (2)

Getting rid of Boolean closure

$BPol(\mathcal{C})$-separation: Three main steps (2)

Step 1

Step 2

Step 3

Getting rid of Boolean closure

(K, L) $BPol(\mathcal{C})$-separable
iff
$\exists k$ $(K, L)^k$ $Pol(\mathcal{C})$-separable

$BPol(\mathcal{C})$-separation: Three main steps (2)

Step 1

Step 2

Step 3

Getting rid of Boolean closure

(K, L) $BPol(\mathcal{C})$-separable
iff
$\exists k$ $(K, L)^k$ $Pol(\mathcal{C})$-separable

Solving generalized $Pol(\mathcal{C})$-separation:
Input: $(L_1, \ldots, L_n)$ (regular)
Output: Is $(L_1, \ldots, L_n)$ $Pol(\mathcal{C})$-separable ?

Generalized $Pol(\mathcal{C})$-separation: Approach (1)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

We do **not work directly with these languages**.

Generalized $Pol(\mathcal{C})$-separation: Approach (1)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

We do **not work directly with these languages**.

Rule number one for separation-like problems:

One always looks at several inputs simultaneously.

Generalized $Pol(\mathcal{C})$-separation: Approach (1)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

We do **not work directly with these languages**.

Rule number one for separation-like problems:

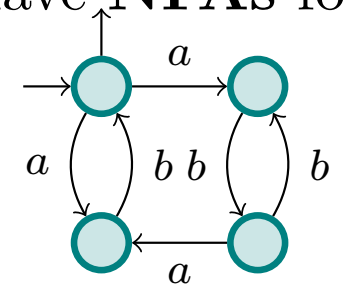
One always looks at several inputs simultaneously.

We use as **set of inputs** which has a special structure.

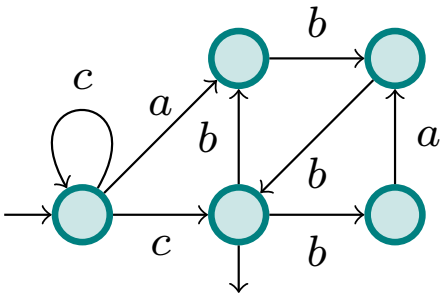
Generalized $Pol(\mathcal{C})$-separation: Approach (2)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

$\Rightarrow$ We have **NFAs** for these languages:



L_1

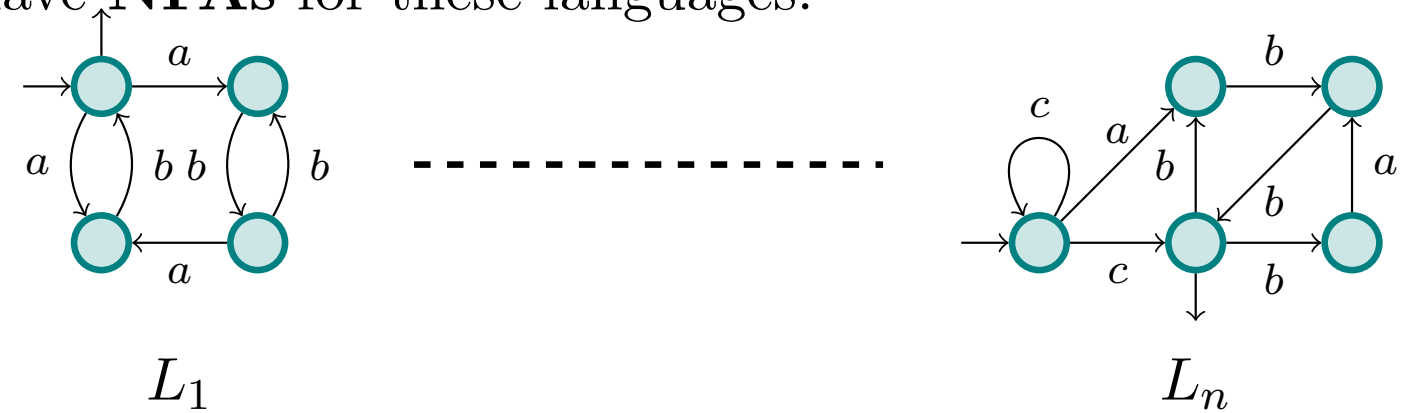


L_n

Generalized $Pol(\mathcal{C})$-separation: Approach (2)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

$\Rightarrow$ We have **NFAs** for these languages:



L_1 L_n

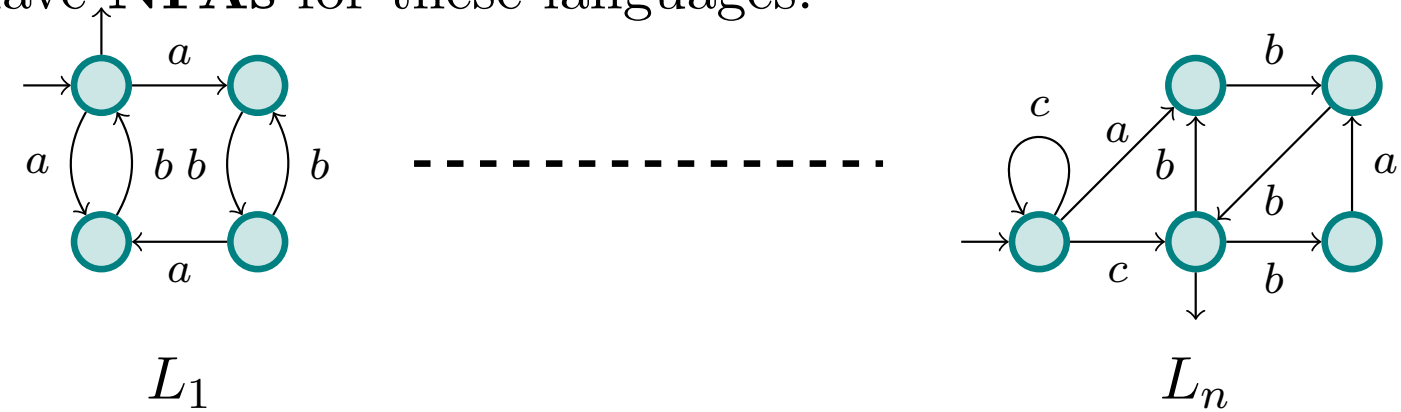
We work with a **set of languages L** containing all languages

$$L_{p,q} = \{w \mid p \xrightarrow{w} q\} \quad (p, q \text{ two states of these } \mathbf{NFAs})$$

Generalized $Pol(\mathcal{C})$-separation: Approach (2)

Our input $(L_1, \ldots, L_n)$ is a **tuple of n regular languages**.

$\Rightarrow$ We have **NFAs** for these languages:



L_1 $\qquad\qquad$ L_n

We work with a **set of languages L** containing all languages

$$L_{p,q} = \{w \mid p \xrightarrow{w} q\} \quad (p, q \text{ two states of these } \mathbf{NFAs})$$

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not** $Pol(\mathcal{C})$**-separable**.

Answer for $(L_1, \ldots, L_n)$ can then be extracted from this information.

Generalized $Pol(\mathcal{C})$-separation: Approach (3)

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not $Pol(\mathcal{C})$-separable**.

The algorithm needs two hypotheses on $\mathbf{L}$:

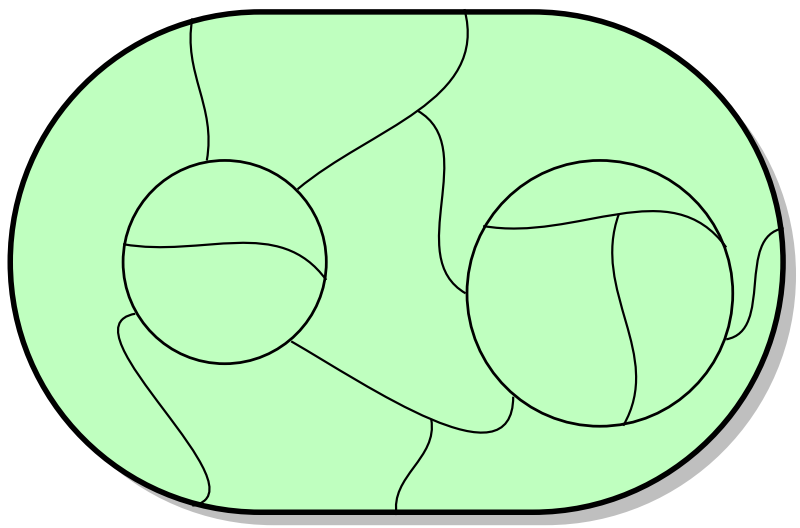
- It has an algebraic structure (given by the fact that it is the set of languages recognized by **NFAs**).

Generalized $Pol(\mathcal{C})$-separation: Approach (3)

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not $Pol(\mathcal{C})$-separable**.

The algorithm needs two hypotheses on $\mathbf{L}$:

- It has an algebraic structure (given by the fact that it is the set of languages recognized by **NFAs**).
- It is **$\mathcal{C}$-compatible**:
  $\mathcal{C}$ is finite $\Rightarrow$ exists **finest partition** of A^* into languages of $\mathcal{C}$:

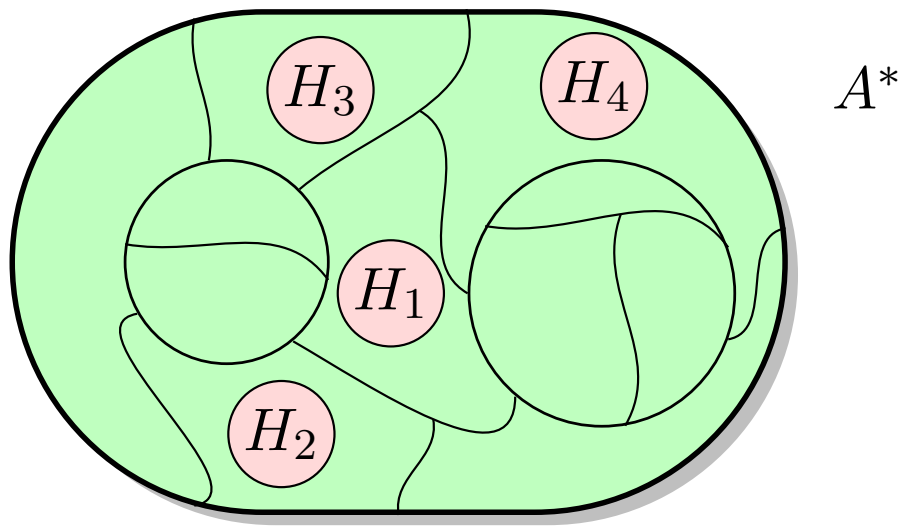


Generalized $Pol(\mathcal{C})$-separation: Approach (3)

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not $Pol(\mathcal{C})$-separable**.

The algorithm needs two hypotheses on $\mathbf{L}$:

- It has an algebraic structure (given by the fact that it is the set of languages recognized by **NFAs**).
- It is **$\mathcal{C}$-compatible**:
  $\mathcal{C}$ is finite $\Rightarrow$ exists **finest partition** of A^* into languages of $\mathcal{C}$:



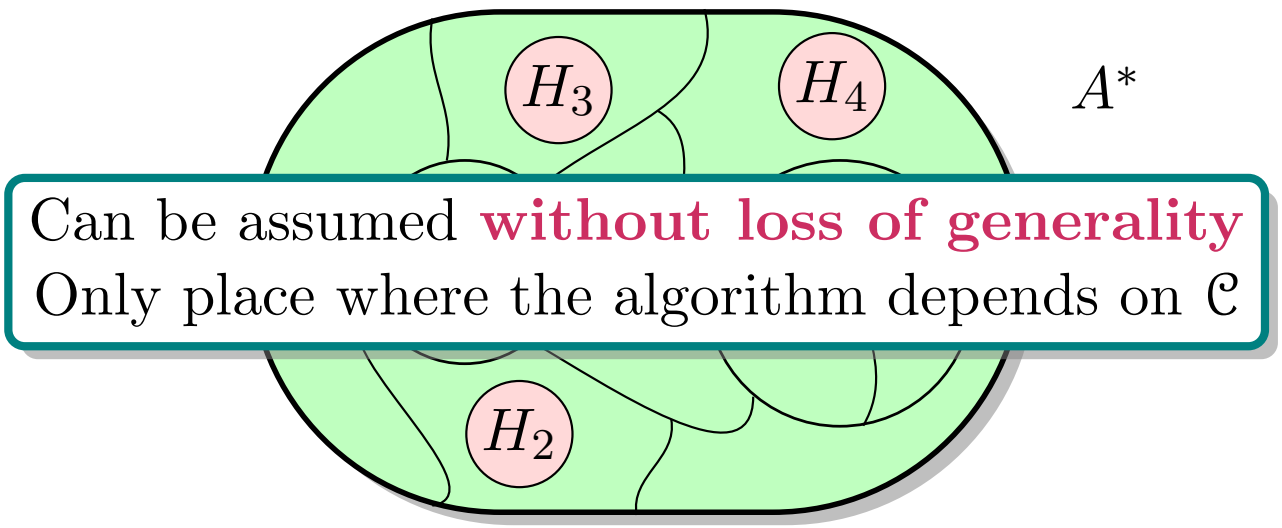
Any $H \in \mathbf{L}$ must be **included in a class of this partition**.

Generalized $Pol(\mathcal{C})$-separation: Approach (3)

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not $Pol(\mathcal{C})$-separable**.

The algorithm needs two hypotheses on $\mathbf{L}$:

- It has an algebraic structure (given by the fact that it is the set of languages recognized by **NFAs**).
- It is **$\mathcal{C}$-compatible**:
  $\mathcal{C}$ is finite $\Rightarrow$ exists **finest partition** of A^* into languages of $\mathcal{C}$:



Any $H \in \mathbf{L}$ must be **included in a class of this partition**.

Least fixpoint computation of $\mathcal{T}^n[\mathbf{L}]$

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not** $Pol(\mathcal{C})$**-separable**.

$\mathcal{T}^n[\mathbf{L}]$ is computed by **induction on** n:

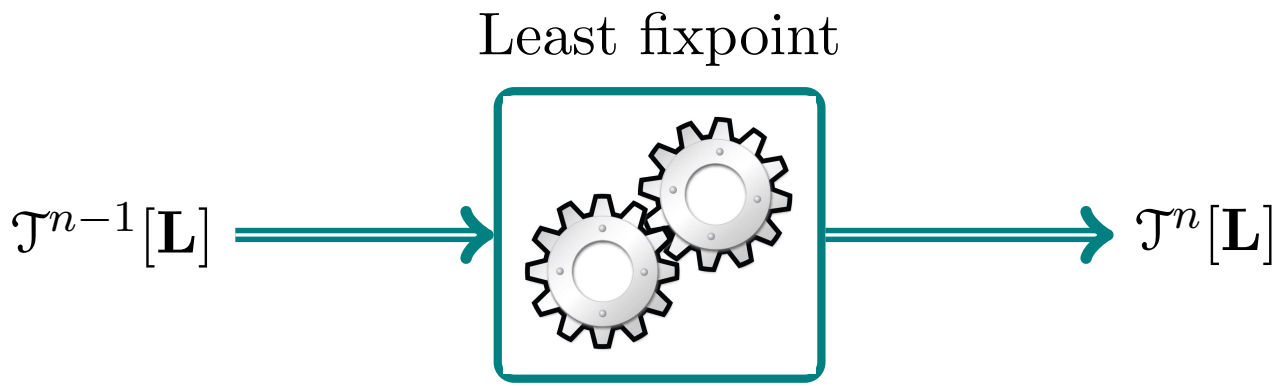
- $\mathcal{T}^1[\mathbf{L}] \subseteq \mathbf{L}$ is the set of **nonempty** languages in $\mathbf{L}$.

Least fixpoint computation of $\mathcal{T}^n[\mathbf{L}]$

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not** $Pol(\mathcal{C})$**-separable**.

$\mathcal{T}^n[\mathbf{L}]$ is computed by **induction on** n:

- $\mathcal{T}^1[\mathbf{L}] \subseteq \mathbf{L}$ is the set of **nonempty** languages in $\mathbf{L}$.
- For $n \geq 2$, one first computes $\mathcal{T}^{n-1}[\mathbf{L}]$ and inject it into a **least fixpoint procedure** which outputs $\mathcal{T}^n[\mathbf{L}]$:

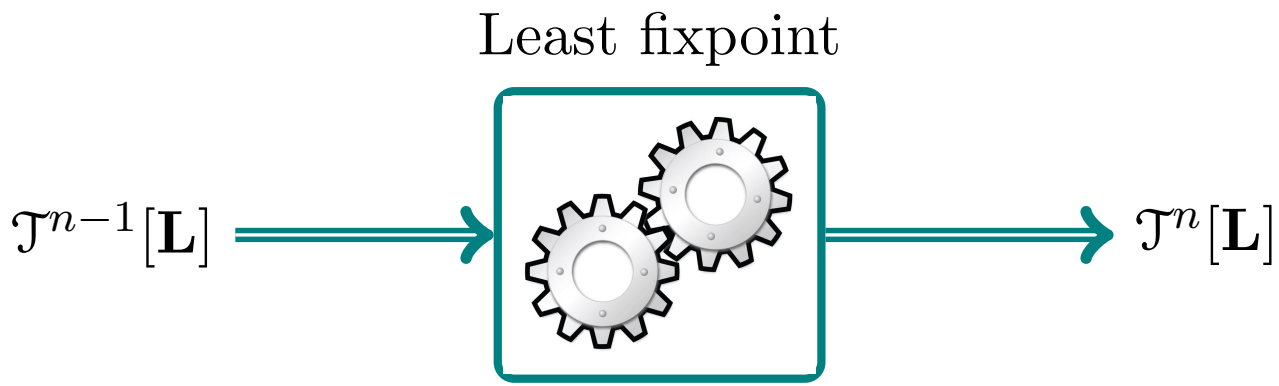


Least fixpoint computation of $\mathcal{T}^n[\mathbf{L}]$

Given $n \geq 1$, we compute the set $\mathcal{T}^n[\mathbf{L}] \subseteq \mathbf{L}^n$ of all tuples $\overline{L} \in \mathbf{L}^n$ which are **not $Pol(\mathcal{C})$-separable**.

$\mathcal{T}^n[\mathbf{L}]$ is computed by **induction on n**:

- $\mathcal{T}^1[\mathbf{L}] \subseteq \mathbf{L}$ is the set of **nonempty** languages in $\mathbf{L}$.
- For $n \geq 2$, one first computes $\mathcal{T}^{n-1}[\mathbf{L}]$ and inject it into a **least fixpoint procedure** which outputs $\mathcal{T}^n[\mathbf{L}]$:



The procedure computes $\mathcal{T}^n[\mathbf{L}]$ from a subset of trivial elements and **adds more with operations until a fixpoint is reached**. Implementing these operations requires having $\mathcal{T}^{n-1}[\mathbf{L}]$ in hand.

$BPol(\mathcal{C})$-separation: Three main steps (3)

Step 1

Step 2

Step 3

Getting rid of Boolean closure

(K, L) $BPol(\mathcal{C})$-separable
iff
$\exists k$ $(K, L)^k$ $Pol(\mathcal{C})$-separable

Solving generalized $Pol(\mathcal{C})$-separation:
Input: $(L_1, \ldots, L_n)$ (regular)
Output: Is $(L_1, \ldots, L_n)$ $Pol(\mathcal{C})$-separable ?

$BPol(\mathcal{C})$-separation: Three main steps (3)

Step 1

Step 2

Step 3

Getting rid of Boolean closure

(K, L) $BPol(\mathcal{C})$-separable
iff
$\exists k$ $(K, L)^k$ $Pol(\mathcal{C})$-separable

Solving generalized $Pol(\mathcal{C})$-separation:
Input: $(L_1, \ldots, L_n)$ (regular)
Output: Is $(L_1, \ldots, L_n)$ $Pol(\mathcal{C})$-separable ?

$BPol(\mathcal{C})$-separation: Three main steps (3)

Step 1

Step 2

Step 3

Getting rid of Boolean closure

(K, L) $BPol(\mathcal{C})$-separable
iff
$\exists k$ $(K, L)^k$ $Pol(\mathcal{C})$-separable

Solving generalized $Pol(\mathcal{C})$-separation:
Input: $(L_1, \ldots, L_n)$ (regular)
Output: Is $(L_1, \ldots, L_n)$ $Pol(\mathcal{C})$-separable ?

Reuse Step 2 as a subprocedure
in our $BPol(\mathcal{C})$ algorithm

$BPol(\mathcal{C})$-separation - General approach

We continue to work with a set of languages $\mathbf{L}$ with appropriate properties.

We compute the set $\mathcal{A}[\mathbf{L}] \subseteq \mathbf{L}^2$ of pairs $(K, L) \in \mathbf{L}^2$ which are **not $BPol(\mathcal{C})$-separable**.

$BPol(\mathcal{C})$-separation - General approach

We continue to work with a set of languages $\mathbf{L}$ with appropriate properties.

We compute the set $\mathcal{A}[\mathbf{L}] \subseteq \mathbf{L}^2$ of pairs $(K, L) \in \mathbf{L}^2$ which are **not $BPol(\mathcal{C})$-separable**.

By the two previous steps

Given two languages L_1, L_2, t.f.a.e.,

1. L_1 $BPol(\mathcal{C})$-separable from L_2.
2. There exists $k \geq 1$ s.t. $(L_1, L_2)^k$ is $Pol(\mathcal{C})$-separable.

$BPol(\mathcal{C})$-separation - General approach

We continue to work with a set of languages $\mathbf{L}$ with appropriate properties.

We compute the set $\mathcal{A}[\mathbf{L}] \subseteq \mathbf{L}^2$ of pairs $(K, L) \in \mathbf{L}^2$ which are **not** $BPol(\mathcal{C})$**-separable**.

By the two previous steps

Given two languages L_1, L_2, t.f.a.e.,

1. L_1 is **not** $BPol(\mathcal{C})$-separable from L_2.
2. For all $k \geq 1$, $(L_1, L_2)^k$ is **not** $Pol(\mathcal{C})$-separable.

$BPol(\mathcal{C})$-separation - General approach

We continue to work with a set of languages $\mathbf{L}$ with appropriate properties.

We compute the set $\mathcal{A}[\mathbf{L}] \subseteq \mathbf{L}^2$ of pairs $(K, L) \in \mathbf{L}^2$ which are **not $BPol(\mathcal{C})$-separable**.

By the two previous steps

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k$ **not** $Pol(\mathcal{C})$-separable (i.e. for all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$).

$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

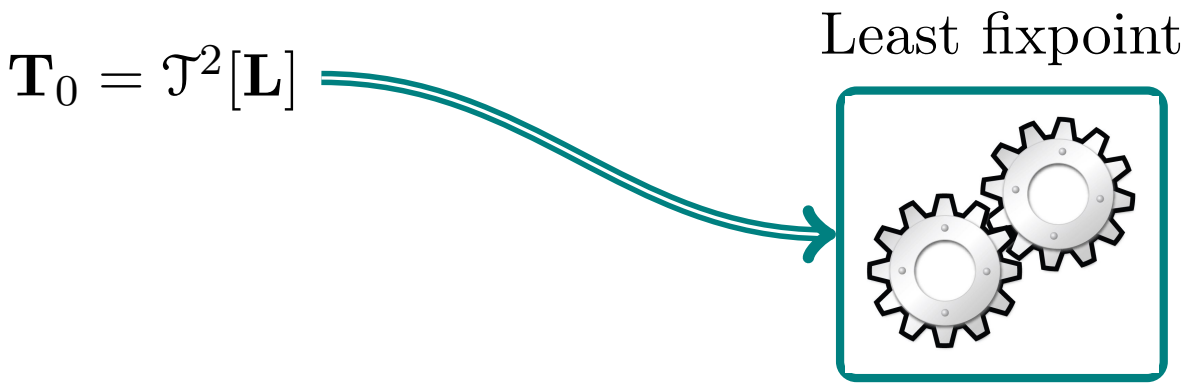
$\mathbf{T}_0 = \mathcal{T}^2[\mathbf{L}]$

$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

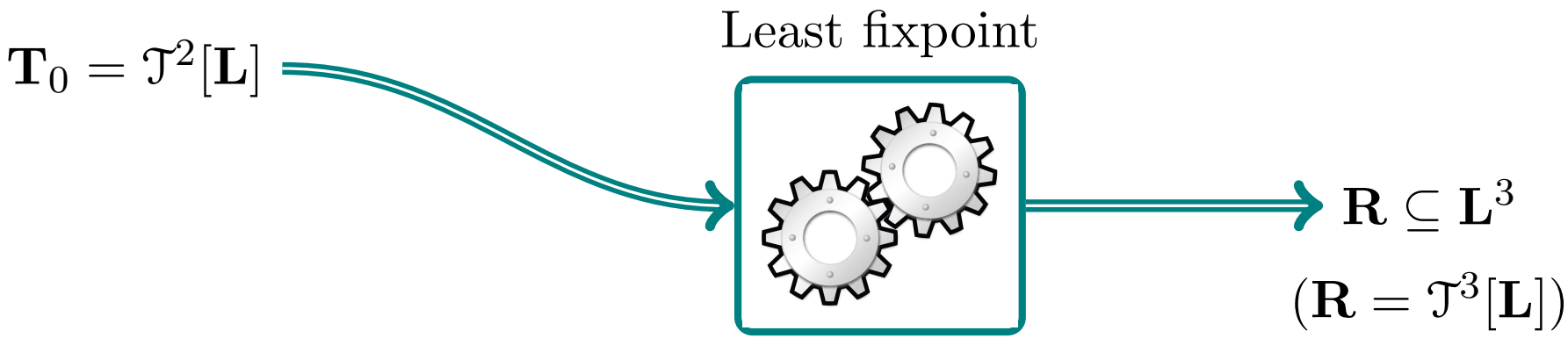


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

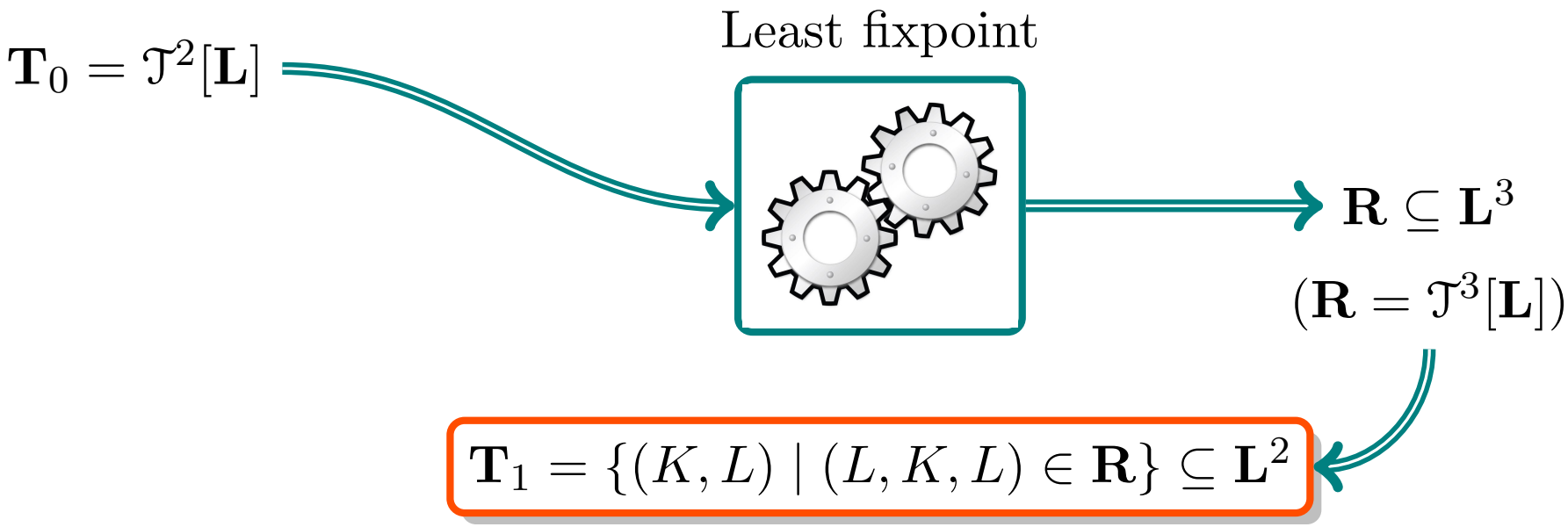


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

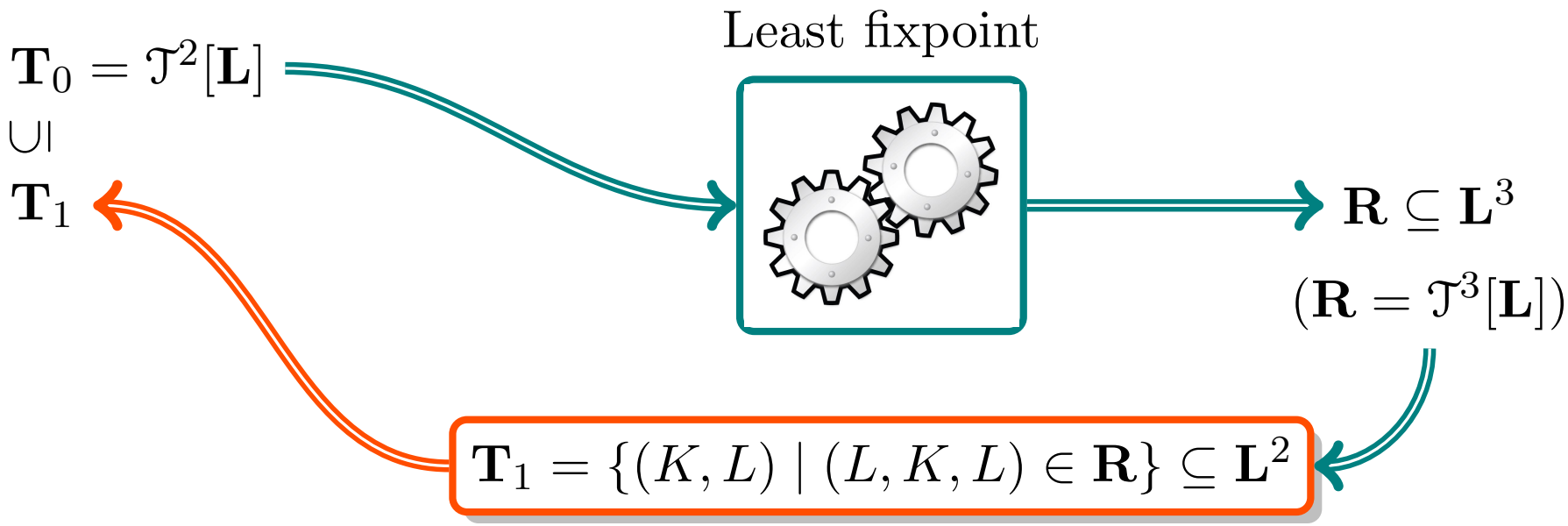


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.



$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

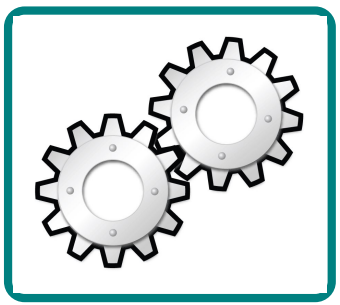
In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

$\mathbf{T}_0 = \mathcal{T}^2[\mathbf{L}]$

$\cup|$

$\mathbf{T}_1$

Least fixpoint

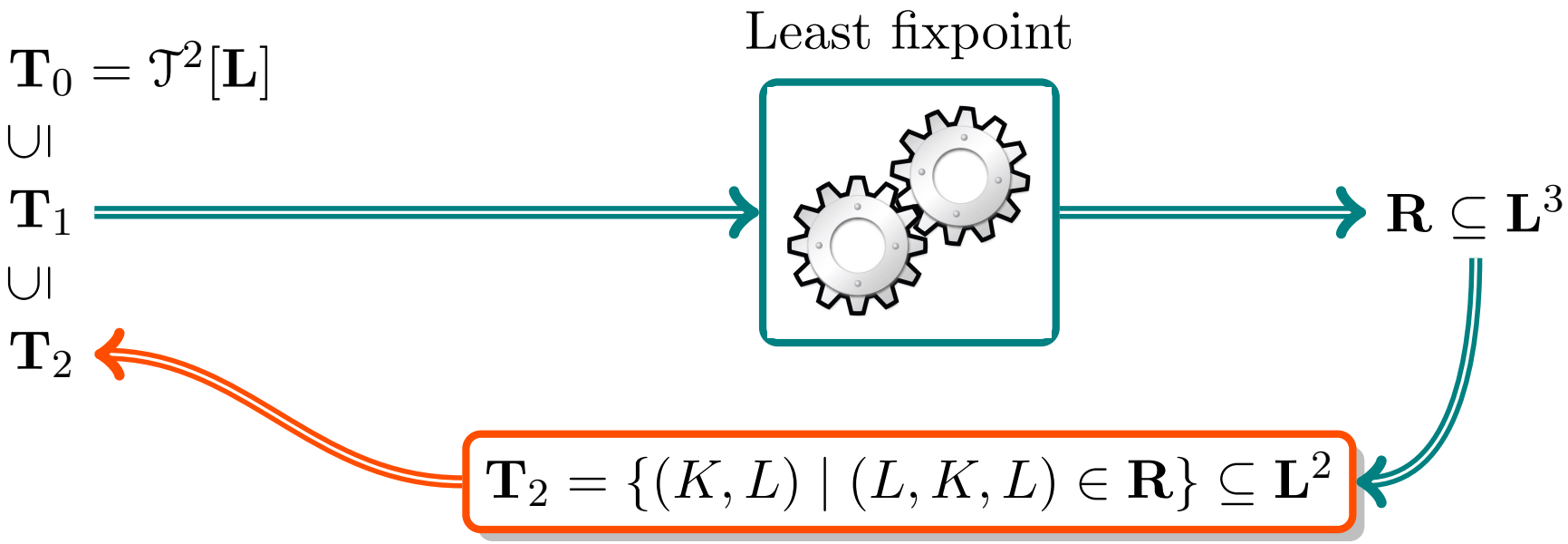


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

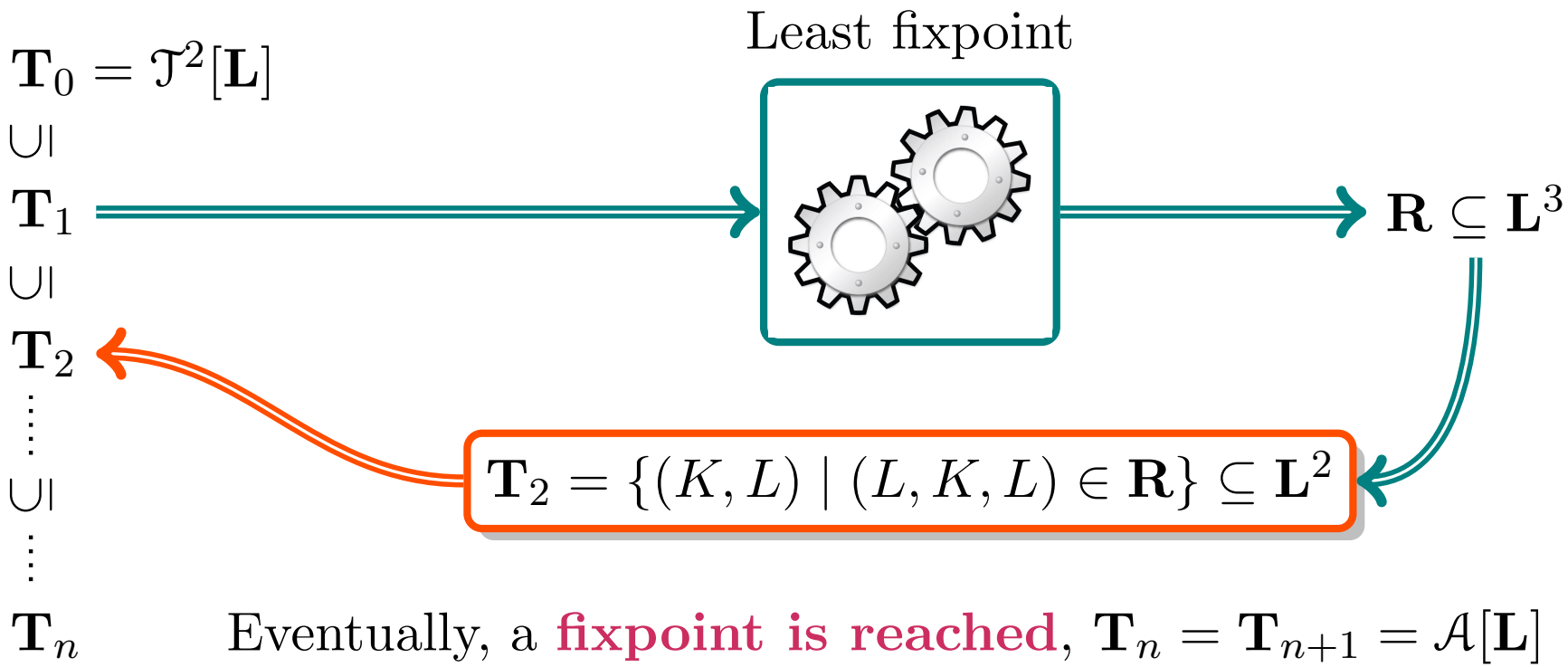


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.

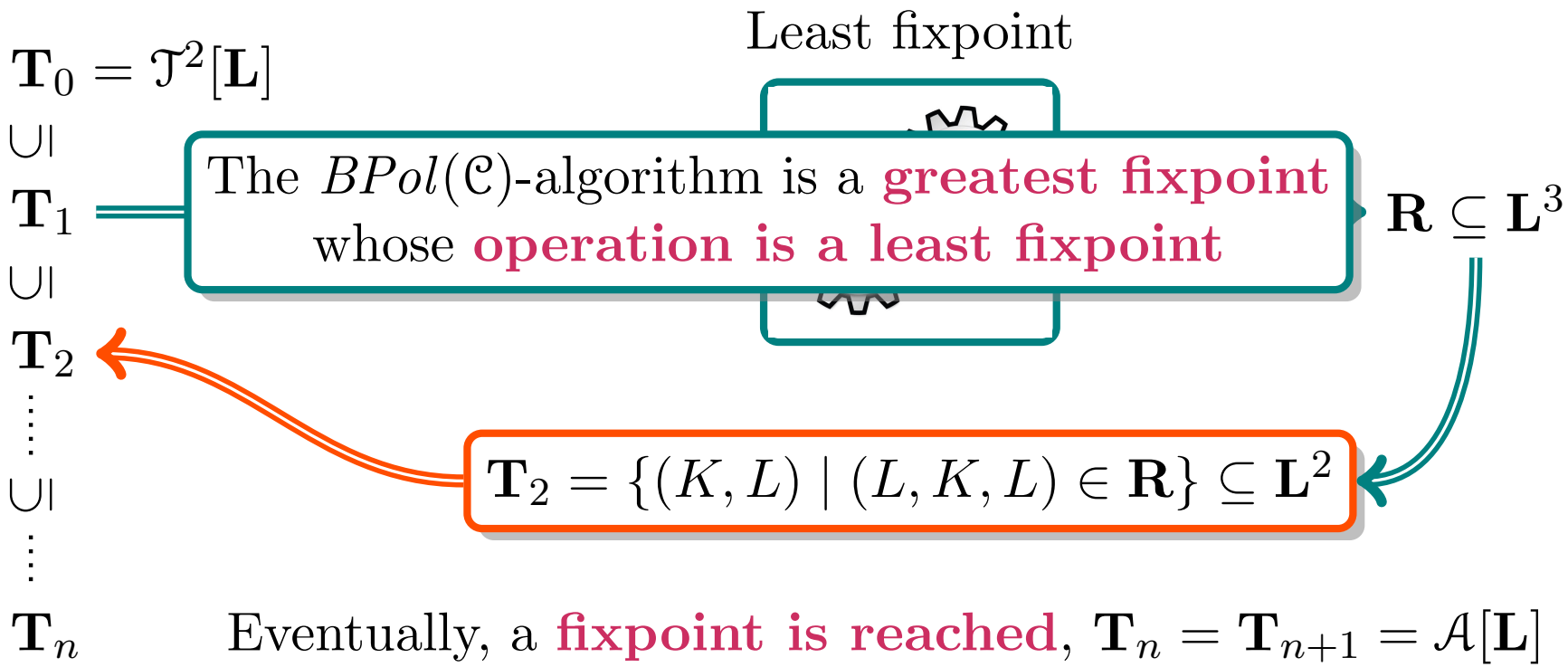


$BPol(\mathcal{C})$-separation - Greatest fixpoint

Given two languages $K, L \in \mathbf{L}$, t.f.a.e.,

1. $(K, L) \in \mathcal{A}[\mathbf{L}]$.
2. For all $k \geq 1$, $(K, L)^k \in \mathcal{T}^{2k}[\mathbf{L}]$.

In particular, $\mathcal{A}[\mathbf{L}] \subseteq \mathcal{T}^2[\mathbf{L}]$. **Greatest fixpoint**:
From $\mathcal{T}^2[\mathbf{L}]$, remove elements with an operation until fixpoint.



Conclusion

Conclusion (1)

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 (1)

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 (1)

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).

Conclusion (1)

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**

Conclusion (2)

Future work

We have the result,

- For any $\mathfrak{C}$,
  $\mathfrak{C}$-separation decidable $\Rightarrow$ $Pol(\mathfrak{C})$-membership decidable.

Conclusion (2)

Future work

We have the result,

- For any $\mathfrak{C}$,
  $\mathfrak{C}$-separation decidable $\Rightarrow$ $Pol(\mathfrak{C})$-membership decidable.

A nontrivial corollary of the $BPol(\mathfrak{C})$-algorithm is as follows:

- For any $\mathfrak{C}$,
  $\mathfrak{C}$-"something" decidable $\Rightarrow$ $BPol(\mathfrak{C})$-membership decidable.

Conclusion (2)

Future work

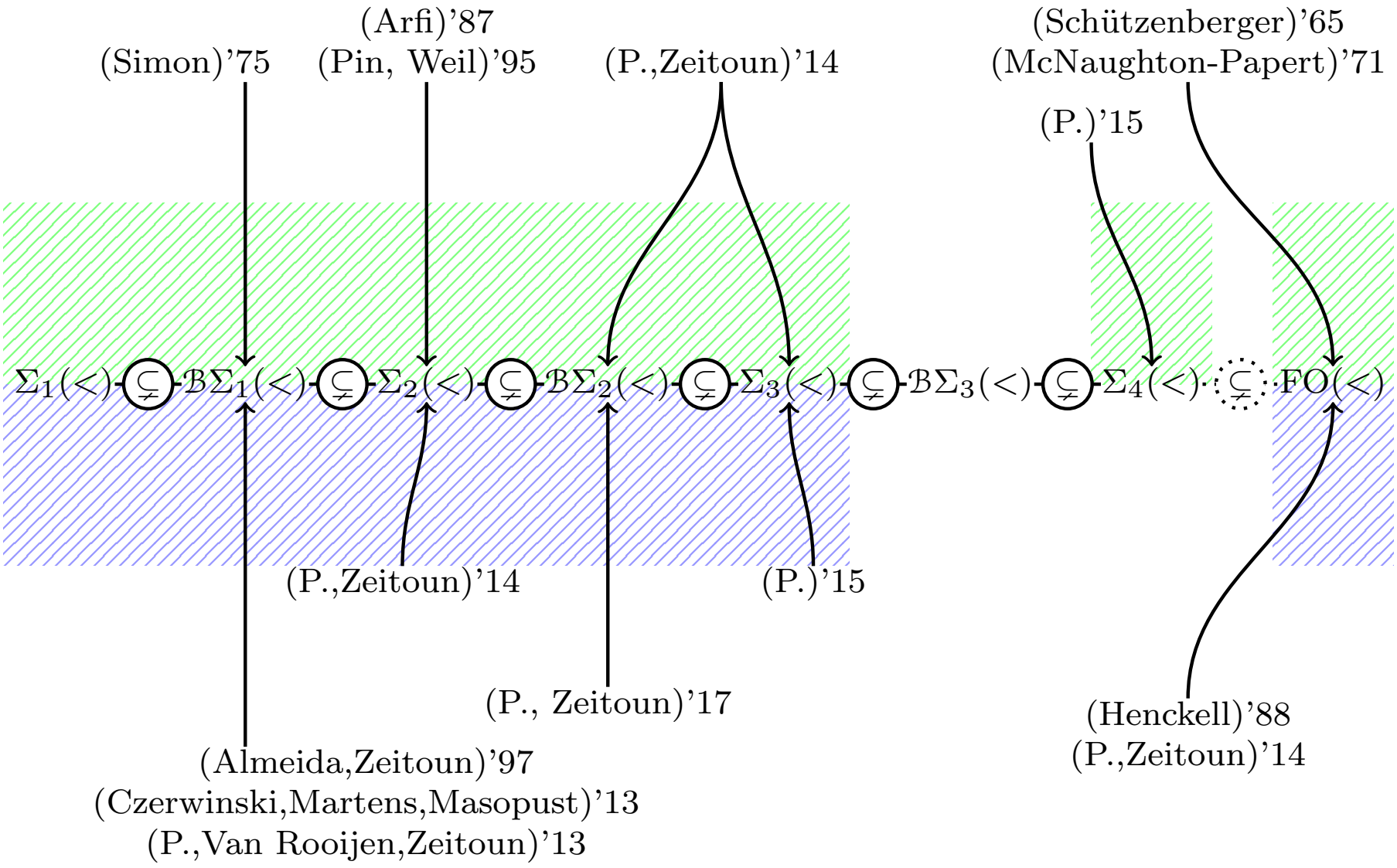
We have the result,

- For any $\mathcal{C}$,
  $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

A nontrivial corollary of the $BPol(\mathcal{C})$-algorithm is as follows:

- For any $\mathcal{C}$,
  $\mathcal{C}$-"something" decidable $\Rightarrow$ $BPol(\mathcal{C})$-membership decidable.

$\mathcal{B}\Sigma_2(<)$-"something" seems to be decidable which would yield a membership algorithm for $\mathcal{B}\Sigma_3(<)$.


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

Thank You