Concatenation hierarchies and separation

Marc Zeitoun

LaBRI, Bordeaux University

Caalm '19, CMI Chennai

23/1/2019

Based on joint work with Thomas Place

Regular Languages, Concatenation, Separation

Regular Languages, Concatenation, Separation

Regular expressions for describing languages

Regular language = set of words built from:

$\emptyset$, $\{\varepsilon\}$, $\{a\}$	$\emptyset$, ε, a
Union	$L_1 + L_2$
Concatenation	$L_1 L_2$
Iteration (star)	L^*

$$L_1 L_2 = \{u_1 \cdot u_2 \mid u_1 \in L_1 \text{ and } u_2 \in L_2\}$$
$$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cdots$$

Regular expressions for describing languages

Regular language = set of words built from:

$\emptyset$, $\{\varepsilon\}$, $\{a\}$	$\emptyset$, ε, a
Union	$L_1 + L_2$
Concatenation	$L_1 L_2$
Iteration (star)	L^*

$$L_1 L_2 = \{u_1 \cdot u_2 \mid u_1 \in L_1 \text{ and } u_2 \in L_2\}$$

$$L^* = \{\varepsilon\} \cup L \cup L^2 \cup L^3 \cdots$$

Example

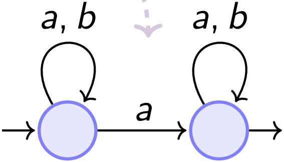
$\left((aa)^* + b\right)^*$ = Words over $\{a, b\}$ with even blocks of a's

Regularity is robust (1): words containing an 'a'

$$(a + b)^* a (a + b)^*$$

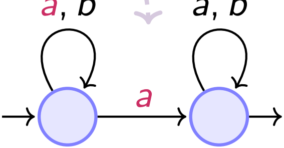
Regularity is robust (1): words containing an 'a'

$$(a+b)^*a(a+b)^*$$



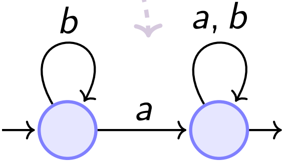
Regularity is robust (1): words containing an ‘a’

$$(a+b)^*a(a+b)^*$$



Regularity is robust (1): words containing an 'a'

$$(a + b)^* a (a + b)^*$$



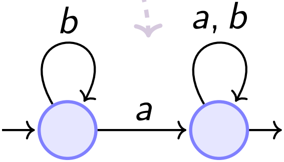
Regularity is robust (1): words containing an '*a*'

$$(a+b)^*a(a+b)^*$$

$$f : (a+b)^* \to (\{0,1\}, \times)$$

$$a \mapsto 0 \quad b \mapsto 1$$

$$f(bab) = f(b)f(a)f(b)$$



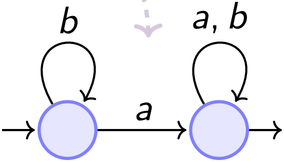
Regularity is robust (1): words containing an '*a*'

$$(a+b)^* a (a+b)^*$$

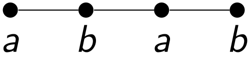
$$f : (a+b)^* \to (\{0,1\}, \times)$$

$$a \mapsto 0 \quad b \mapsto 1$$

$$f(bab) = f(b)f(a)f(b)$$



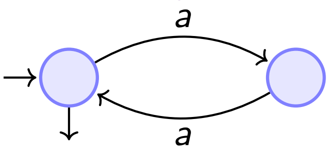
$$\exists x \; a(x)$$



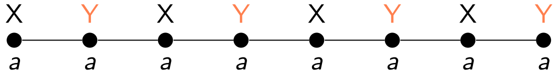
Regularity, robustness (2): words of even length

$$(aa)^*$$

$$f : a^* \to (\mathbb{Z}/2\mathbb{Z}, +)$$
$$a \mapsto 1$$



$$\forall X\, \forall Y$$
$$\big(\ \min \in \mathrm{X} \wedge \mathrm{Alternate}(\mathrm{X}, \mathrm{Y})\ \big) \Rightarrow \max \in Y$$



Robustness Theorem for Regular Word Languages

Kleene, Büchi, Elgot, Trakhtenbrot (60s)

For a language of finite words L, **TFAE**:

1. L is described by a regular expression ($\cup$, $\bullet$, $\star$).
2. L is recognized by an NFA.
3. L is recognized by a DFA.
4. L is described by an MSO sentence.
5. L is recognized by a morphism into a finite monoid.

.

Robustness Theorem for Regular Word Languages

Kleene, Büchi, Elgot, Trakhtenbrot (60s)

For a language of finite words L, **TFAE**:

1. L is described by a regular expression ($\cup$, $\bullet$, $\star$).
2. L is recognized by an NFA.

3. L is recognized by a DFA.
4. L is described by an MSO sentence.
5. L is recognized by a morphism into a finite monoid.

.

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ **Built-in complement**

Robustness Theorem for Regular Word Languages

Kleene, Büchi, Elgot, Trakhtenbrot (60s)

For a language of finite words L, **TFAE**:

1. L is described by a regular expression ($\cup$, $\bullet$, $\star$).
2. L is recognized by an NFA.

3. L is recognized by a DFA.
4. L is described by an MSO sentence.
5. L is recognized by a morphism into a finite monoid.
6. L is described by a generalized regular expression.

Built-in complement

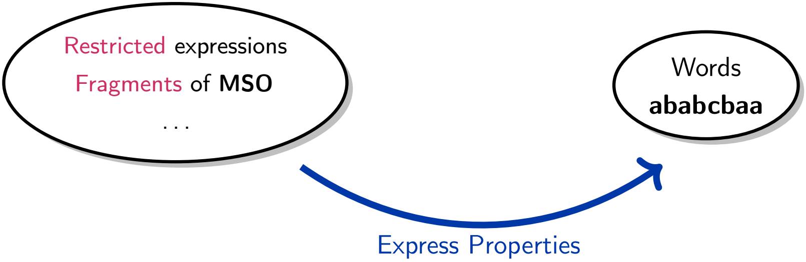
Generalized regular expression:

- Built from singletons, using $\cup$, $\bullet$, $\star$ and complement.

Goal: Understanding expressiveness of fragments

Descriptive Formalisms
Syntax

Structures
Semantics



We want to understand what a formalism can express

What does "understand" mean?

Meaningful fragments of regular languages

What languages can be expressed by a simple expression/formula?

What does simple mean? Several possible choices, e.g.:

- For (generalized) expressions: number of nested stars.
- For formulas: number of alternations between $\exists$ and $\forall$.

Meaningful fragments of regular languages

What languages can be expressed by a simple expression/formula?

What does simple mean? Several possible choices, e.g.:

- For (generalized) expressions: number of nested stars.
- For formulas: number of alternations between $\exists$ and $\forall$.

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

Meaningful fragments of regular languages

What languages can be expressed by a simple expression/formula?

What does simple mean? Several possible choices, e.g.:

- For (generalized) expressions: number of nested stars.
- For formulas: number of alternations between $\exists$ and $\forall$.

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

E.g., over alphabet $\{a, b\}$: $(a^* b^*)^* = (a + b)^*$

Meaningful fragments of regular languages

What languages can be expressed by a simple expression/formula?

What does simple mean? Several possible choices, e.g.:

- For (generalized) expressions: number of nested stars.
- For formulas: number of alternations between $\exists$ and $\forall$.

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

E.g., over alphabet $\{a, b\}$: $(a^* b^*)^* = (a + b)^* = \overline{\emptyset}$.

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

What is the GSH of b^* over $A = \{a, b\}$?

- GSH at most 1: $b^* = \overline{(a+b)^* a (a+b)^*}$.
- Can we do better?

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

What is the GSH of b^* over $A = \{a, b\}$?

- GSH at most 1: $b^* = \overline{(a+b)^* a (a+b)^*}$.
- Can we do better? Yes: $\overline{\overline{\emptyset}\ a\ \overline{\emptyset}}$.

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

What is the GSH of $(a(bb)^*a)^*$ over $A = \{a, b\}$?

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

What is the GSH of $(a(bb)^*a)^*$ over $A = \{a, b\}$?

- GSH at most 1:

$$\varepsilon \; + \; aA^* \cap A^*a \cap \overline{A^*ab(bb)^*aA^*}$$

- Can we do better?

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

What is the GSH of $(a(bb)^*a)^*$ over $A = \{a, b\}$?

- GSH at most 1:

$$\varepsilon \;+\; aA^* \cap A^*a \cap \overline{A^*ab(bb)^*aA^*}$$

- Can we do better?

$$\varepsilon \;+\; a\overline{\emptyset} \cap \overline{\emptyset}a \cap \overline{\overline{\emptyset}ab(bb)^*a\overline{\emptyset}}$$

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

Natural problems, but turned out to be **difficult**:

- Problem 1 **solved** in **1988** by Hashiguchi and **2005** by Kirsten.
- Problem 2 **open**: no language of GSH 2 is known!

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

Natural problems, but turned out to be **difficult**:

- Problem 1 **solved** in **1988** by Hashiguchi and **2005** by Kirsten.
- Problem 2 **open**: no language of GSH 2 is known!

  $\Longrightarrow$ "restrict the generalization": what about languages of GSH 0?

Star height problems

Star height problems (Eggan, 1963)

For a regular language L, compute for it:

1. A regular expression with the **minimum** number of nested stars
2. A **generalized** expression

Natural problems, but turned out to be **difficult**:

- Problem 1 **solved** in **1988** by Hashiguchi and **2005** by Kirsten.
- Problem 2 **open**: no language of GSH 2 is known!

  $\Longrightarrow$ "restrict the generalization": what about languages of GSH 0?

Notation. GSH0 = Star-free = SF

Temporary conclusion

- Regular languages are easy, but complement is hard.
- Understanding a class = designing algorithms testing membership

Outline

1. The Membership Problem

2. Concatenation Hierarchies

3. Beyond Membership: Separation

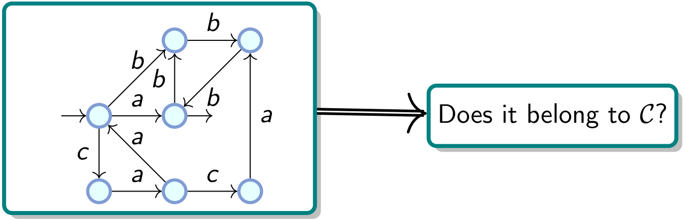
4. Generalizing Separation

The Membership Problem

Capturing expressiveness: seminal result

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

- **INPUT** A (regular) language L.
- **QUESTION** Does L belong to $\mathcal{C}$?



Capturing expressiveness: seminal result

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

- **INPUT** A (regular) language L.
- **QUESTION** Does L belong to $\mathcal{C}$?

Examples of classes $\mathcal{C}$:

- Languages definable in **FO**.
- Languages of SH k.
- Languages of GSH $k \geq 1$.
- Languages of GSH 0 (called star-free, denoted SF).

Capturing expressiveness: seminal result

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

- **INPUT** A (regular) language L.
- **QUESTION** Does L belong to $\mathcal{C}$?

Examples of classes $\mathcal{C}$:

- Languages definable in **FO**.
- Languages of SH k.
- Languages of GSH $k \geq 1$.
- Languages of GSH 0 (called star-free, denoted SF).

Schützenberger '65

For L a regular language, the following are equivalent:

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

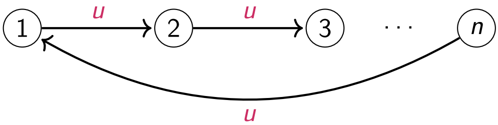
Counter-free automata

Schützenberger '65

For L a regular language, the following are equivalent:

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

An automaton is counter-free if it has no pattern:



$n \geq 2$

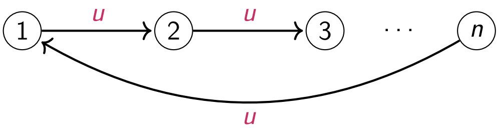
Counter-free automata

Schützenberger '65

For L a regular language, the following are equivalent:

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

An automaton is counter-free if it has no pattern:



$n \geq 2$

Example

Minimal DFA of b^* has no counter	$\Rightarrow$	Star-free
Minimal DFA of $(a(bb)^*a)^*$ has a counter	$\Rightarrow$	Not star-free

Star-free expressions vs. first-order logic

First-order logic, with only the linear order '$<$'.

$$\begin{array}{ccccccccc} a & b & b & c & a & a & a & c & a \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

Word = sequence of labeled positions.

Star-free expressions vs. first-order logic

First-order logic, with only the linear order '$<$'.

$$\begin{array}{ccccccccc} a & b & b & c & a & a & a & c & a \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

Word = sequence of labeled positions.

- Positions can be quantified: $\exists x \varphi$, $\forall x \varphi$.
- One binary predicate: the linear-order $x < y$.
- Unary predicates $a(x)$, $b(x)$, $c(x)$ testing the label of position x.

Star-free expressions vs. first-order logic

First-order logic, with only the linear order '$<$'.

$$\begin{array}{ccccccccc} a & b & b & c & a & a & a & c & a \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

Word = sequence of labeled positions.

- Positions can be quantified: $\exists x \varphi$, $\forall x \varphi$.
- One binary predicate: the linear-order $x < y$.
- Unary predicates $a(x), b(x), c(x)$ testing the label of position x.
- No quantification over sets of positions.

Star-free expressions vs. first-order logic

First-order logic, with only the linear order '$<$'.

$$\begin{array}{ccccccccc} a & b & b & c & a & a & a & c & a \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

Word = sequence of labeled positions.

- Positions can be quantified: $\exists x \varphi$, $\forall x \varphi$.
- One binary predicate: the linear-order $x < y$.
- Unary predicates $a(x), b(x), c(x)$ testing the label of position x.
- No quantification over sets of positions.

Example: in the future of every 'a', there is a 'b'

$$\forall x \left(a(x) \Rightarrow \exists y \left((y > x) \wedge b(y) \right) \right)$$

Why is Schützenberger's theorem interesting?

1. Link with first-order logic **FO**.

Schützenberger '65, McNaughton, Papert '71

For L a regular language, the following are equivalent:

1. L is **FO**-definable. **semantic**
2. L is star-free. **semantic**
3. The minimal automaton of L is counter-free. **syntactic**

Why is Schützenberger's theorem interesting?

1. Link with first-order logic **FO**.

Schützenberger '65, McNaughton, Papert '71

For L a regular language, the following are equivalent:

1. L is **FO**-definable. **semantic**
2. L is star-free. **semantic**
3. The minimal automaton of L is counter-free. **syntactic**

SF	FO
$A^*, \emptyset$	True, False
$\cup,\ A^*\setminus$	$\vee,\ \neg$
KaL	$\exists x\ a(x) \wedge \varphi_K^{<x}(x) \wedge \varphi_L^{>x}(x)$

Why is Schützenberger's theorem interesting?

1. Link with first-order logic **FO**.

Schützenberger '65, McNaughton, Papert '71

For L a regular language, the following are equivalent:

1. L is **FO**-definable. **semantic**
2. L is star-free. **semantic**
3. The minimal automaton of L is counter-free. **syntactic**

SF	FO
$A^*, \emptyset$	True, False
$\cup, \ A^* \setminus$	$\vee, \ \neg$
KaL	$\exists x \ a(x) \wedge \varphi_K^{<x}(x) \wedge \varphi_L^{>x}(x)$

2. Provides an effective characterization of **SF** and **FO**.

Why is Schützenberger's theorem interesting?

1. Link with first-order logic **FO**.

Schützenberger '65, McNaughton, Papert '71

For L a regular language, the following are equivalent:

1. L is **FO**-definable. **semantic**
2. L is star-free. **semantic**
3. The minimal automaton of L is counter-free. **syntactic**

SF	FO
$A^*, \emptyset$	True, False
$\cup,\ A^*\setminus$	$\vee,\ \neg$
KaL	$\exists x\ a(x) \wedge \varphi_K^{<x}(x) \wedge \varphi_L^{>x}(x)$

2. Provides an effective characterization of **SF** and **FO**.
3. Constructive proof $\Rightarrow$ normal forms for **SF**-expressions/**FO**.

Recap

- Understanding fragment $\mathcal{C}$ = solving $\mathcal{C}$-**membership**

- **Successful methodology for SF = FO**, reproduced
  - For other logical classes on words (eg, several restrictions of FO).
  - For other structures: infinite words, trees.

- Proof provides a **canonical representation** of languages in $\mathcal{C}$.

Recap

- Understanding fragment $\mathcal{C}$ = solving $\mathcal{C}$-**membership**
- **Successful methodology for SF = FO**, reproduced
  - For other logical classes on words (eg, several restrictions of FO).
  - For other structures: infinite words, trees.
- Proof provides a **canonical representation** of languages in $\mathcal{C}$.
- Still, the methodology seems to **fail for some major classes**.

Concatenation Hierarchies

Concatenation hierarchies: Motivation

Definition of SF

- **SF** = smallest class such that:
  - $\emptyset \in \mathrm{SF}$ and $A^* \in \mathrm{SF}$.
  - SF is closed under Boolean operations over A^*.
  - SF is closed under marked concatenation $K, L \mapsto KaL$.

Concatenation hierarchies: Motivation

Definition of SF

- **SF** = smallest class such that:
  - $\emptyset \in \mathrm{SF}$ and $A^* \in \mathrm{SF}$.
  - SF is closed under Boolean operations over A^*.
  - SF is closed under marked concatenation $K, L \mapsto KaL$.

Goal

Classify SF languages according to some complexity measure.

Complexity measures of SF/FO languages

What languages can be expressed by a simple expression/formula?

What does simple mean? Several possible choices, e.g.:

- For **SF**: number of alternations complement/concatenation.
- For **FO**: number of alternations between $\exists$ and $\forall$.

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Straubing-Thérien hierarchy '81

- $\mathrm{ST}[0] = \{\emptyset, A^*\}$.

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Straubing-Thérien hierarchy '81

- $\text{ST}[0] = \{\emptyset, A^*\}$.
- $\text{ST}\left[n + \frac{1}{2}\right] = Pol(\text{ST}[n])$.

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Straubing-Thérien hierarchy '81

- $\text{ST}[0] = \{\emptyset, A^*\}$.
- $\text{ST}\left[n + \frac{1}{2}\right] = Pol(\text{ST}[n])$.
- $\text{ST}[n + 1] = Bool(\text{ST}\left[n + \frac{1}{2}\right])$.

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Straubing-Thérien hierarchy '81

- $\text{ST}[0] = \{\emptyset, A^*\}$.
- $\text{ST}\left[n+\frac{1}{2}\right] = Pol(\text{ST}[n])$.
- $\text{ST}[n+1] = Bool(\text{ST}\left[n+\frac{1}{2}\right])$.

Brzozowski-Cohen hierarchy '71

- $\text{BC}[0] = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$.
- $\text{BC}\left[n+\frac{1}{2}\right] = Pol(\text{BC}[n])$.
- $\text{BC}[n+1] = Bool(\text{BC}\left[n+\frac{1}{2}\right])$.

Two standard complexity hierarchies inside SF

Two classes built on top of $\mathcal{C}$

- Boolean closure $Bool(\mathcal{C})$.
- Polynomial closure $Pol(\mathcal{C})$ = closure under **marked concatenation** + **union**

How many needed alternations between Boolean operations and concatenations?

Straubing-Thérien hierarchy '81

- $\mathrm{ST}[0] = \{\emptyset, A^*\}$.
- $\mathrm{ST}\left[n+\frac{1}{2}\right] = Pol(\mathrm{ST}[n])$.
- $\mathrm{ST}[n+1] = Bool\left(\mathrm{ST}\left[n+\frac{1}{2}\right]\right)$.

Brzozowski-Cohen hierarchy '71

- $\mathrm{BC}[0] = \{\emptyset, \{\varepsilon\}, A^+, A^*\}$.
- $\mathrm{BC}\left[n+\frac{1}{2}\right] = Pol(\mathrm{BC}[n])$.
- $\mathrm{BC}[n+1] = Bool\left(\mathrm{BC}\left[n+\frac{1}{2}\right]\right)$.

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

Brzozowski-Cohen and Straubing-Thérien hierarchies

Natural questions

- Are the hierarchies strict?
- Logical description of each level?
- What is known about membership?

Brzozowski-Cohen and Straubing-Thérien hierarchies

Natural questions

- Are the hierarchies strict?
- Logical description of each level?
- What is known about membership?

What is known

1. Both are strict (Brzozowski-Knast 1978 + interleaving),

$$(a \cdots (a(ab)^* b)^* \cdots b)^*$$

2. Natural logical description wihin FO.
3. Membership for BC reduces to membership for ST.
4. Membership solved for only few levels.

Logical counterpart: quantifier alternation hierarchies

Intuition: marked concatenation corresponds to $\exists$.

- $\Sigma_i = \underbrace{\exists^*\forall^*\exists^*\forall^*\exists^*\cdots}_{\text{at most } i \text{ blocks } \exists^* \text{ or } \forall^*} \quad \varphi,$ (φ quantifier free).
- $\mathcal{B}\Sigma_i$ = Finite Boolean combinations of Σ_i.

Logical counterpart: quantifier alternation hierarchies

Intuition: marked concatenation corresponds to $\exists$.

- $\Sigma_i = \underbrace{\exists^*\forall^*\exists^*\forall^*\exists^* \cdots}_{\text{at most } i \text{ blocks } \exists^* \text{ or } \forall^*} \quad \varphi, \qquad (\varphi \text{ quantifier free}).$
- $\mathcal{B}\Sigma_i$ = Finite Boolean combinations of Σ_i.

Quantifier Alternation Hierarchies

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

Logical counterpart: quantifier alternation hierarchies

Intuition: marked concatenation corresponds to $\exists$.

- $\Sigma_i = \underbrace{\exists^*\forall^*\exists^*\forall^*\exists^* \cdots}_{\text{at most } i \text{ blocks } \exists^* \text{ or } \forall^*} \quad \varphi,$ $\qquad$ (φ quantifier free).
- $\mathcal{B}\Sigma_i$ = Finite Boolean combinations of Σ_i.

Quantifier Alternation Hierarchies

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

Two versions

- Order signature: $<$ and $a()$.
- Enriched signature: $<$, $a()$, $+1$, $\min()$, $\max()$ and ε.

Logical counterpart: quantifier alternation hierarchies

Intuition: marked concatenation corresponds to $\exists$.

- $\Sigma_i = \underbrace{\exists^*\forall^*\exists^*\forall^*\exists^*\cdots}_{\text{at most } i \text{ blocks } \exists^* \text{ or } \forall^*} \quad \varphi, \qquad (\varphi \text{ quantifier free}).$
- $\mathcal{B}\Sigma_i$ = Finite Boolean combinations of Σ_i.

Quantifier Alternation Hierarchies

$$\begin{array}{ccccccccccccccc} \Sigma_1 & \subsetneq & \mathcal{B}\Sigma_1 & \subsetneq & \Sigma_2 & \subsetneq & \mathcal{B}\Sigma_2 & \subsetneq & \Sigma_3 & \subsetneq & \mathcal{B}\Sigma_3 & \subsetneq & \Sigma_4 & \subsetneq & \mathrm{FO} \\ \frac{1}{2} & & 1 & & \frac{3}{2} & & 2 & & \frac{5}{2} & & 3 & & \frac{7}{2} & & \end{array}$$

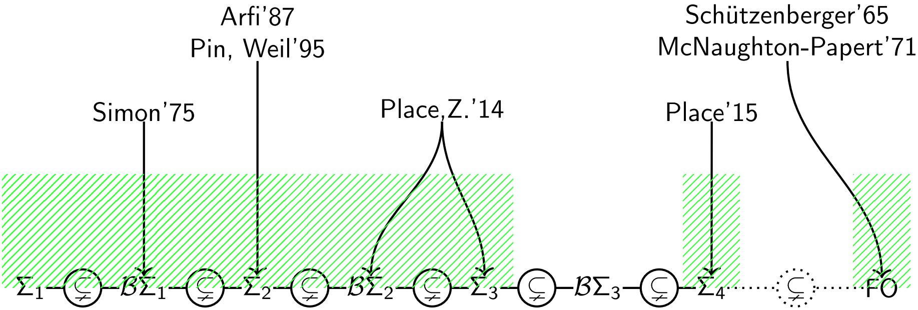
Two versions

- Order signature: $<$ and $a()$.
- Enriched signature: $<$, $a()$, $+1$, $\min()$, $\max()$ and ε.

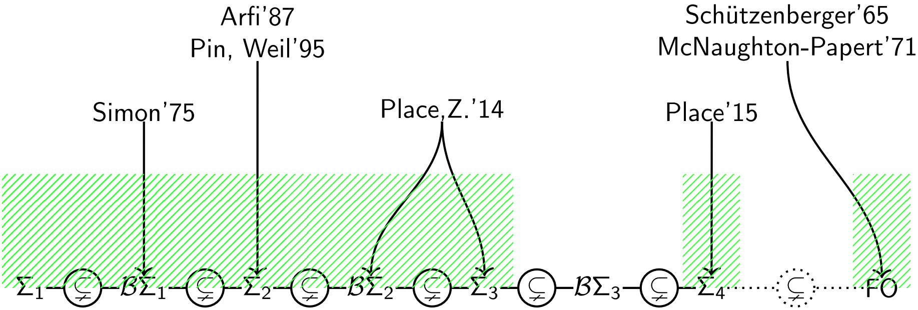
Logical Correspondence Theorem (Thomas '82, Perrin-Pin '86)

- Straubing-Thérien hierarchy = order quantifier alternation hierarchy.
- Brzozowski-Cohen hierarchy = enriched quantifier alternation hierarchy.

The membership problem for BC and ST hierarchies



The membership problem for BC and ST hierarchies



Enrichment Theorem for membership (Straubing, 1985 – Pin, Weil 1997)

Membership for a level in the enriched hierarchy (ie, BC)
reduces to
Membership for the same level in the order hierarchy (ie, ST).

Generalizations in two directions

1. Proofs are ad hoc for BC and ST: obtain generic theorems.
   For given $\mathcal{C}$, what about $Pol(\mathcal{C})$, $Bool(Pol(\mathcal{C}))$,...

2. Recent results via generalizations of membership:
   - separation,
   - covering.

Generic concatenation hierarchies

Generic pattern parametrized by the basis

- $\mathcal{C}[0]$ (basis) Boolean algebra in REG closed under left/right quotients.

Generic concatenation hierarchies

Generic pattern parametrized by the basis

- $\mathcal{C}[0]$ (basis) Boolean algebra in REG closed under left/right quotients.
- $\mathcal{C}[n+\frac{1}{2}]$: close $\mathcal{C}[n]$ under $K, L \mapsto KaL$ and $\cup$.
- $\mathcal{C}[n+1]$: close $\mathcal{C}[n+\frac{1}{2}]$ under Boolean operations.

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

Generic concatenation hierarchies

Generic pattern parametrized by the basis

- $\mathcal{C}[0]$ (basis) Boolean algebra in REG closed under left/right quotients.
- $\mathcal{C}[n+\frac{1}{2}]$: close $\mathcal{C}[n]$ under $K, L \mapsto KaL$ and $\cup$.
- $\mathcal{C}[n+1]$: close $\mathcal{C}[n+\frac{1}{2}]$ under Boolean operations.

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

Examples

- Straubing-Thérien: $\mathcal{C}[0] = \{\emptyset, A^*\}$.
- Brzozowski-Cohen: $\mathcal{C}[0] = \{\emptyset, \{\varepsilon\}, A^*, A^+\}$.
- Pin-Margolis: $\mathcal{C}[0] =$ group languages.

Generic Hierarchies

Natural questions

- Are the hierarchies strict?
- Logical description of each level?
- What is known about membership?

Strictness of generic hierarchies

Strictness Theorem (Place, Z. '17)

Any hierarchy whose basis is finite is **strict**.

Generic logical correspondence

Logical Correspondence Theorem (Place, Z. '17)

For any basis $\mathcal{C}$, there is a natural set $\mathcal{S}$ of first order predicates, st.

Concatenation hierarchy of basis $\mathcal{C}$

=

Quantifier alternation hierarchy over signature $\mathcal{S}$

Generalizes the correspondences discovered for BC and ST hierarchies.

Generic logical correspondence

Logical Correspondence Theorem (Place, Z. '17)

For any basis $\mathcal{C}$, there is a natural set $\mathcal{S}$ of first order predicates, st.

Concatenation hierarchy of basis $\mathcal{C}$

$=$

Quantifier alternation hierarchy over signature $\mathcal{S}$

Generalizes the correspondences discovered for BC and ST hierarchies.

Intuition

For each $L \in \mathcal{C}$, add 4 predicates in addition to $<$ and $a(), b(), \dots$

- $w \models I_L(x, y)$ when $x < y$ and $w]x, y[\in L$ (*Infix*).

Generic logical correspondence

Logical Correspondence Theorem (Place, Z. '17)

For any basis $\mathcal{C}$, there is a natural set $\mathcal{S}$ of first order predicates, st.

$$\text{Concatenation hierarchy of basis } \mathcal{C} = \text{Quantifier alternation hierarchy over signature } \mathcal{S}$$

Generalizes the correspondences discovered for BC and ST hierarchies.

Intuition

For each $L \in \mathcal{C}$, add 4 predicates in addition to $<$ and $a(), b(), \dots$

- $w \models I_L(x, y)$ when $x < y$ and $w]x, y[\in L$ (*I*nfix).
- $w \models P_L(y)$ when $w[1, y[\in L$ (*P*refix).
- $w \models S_L(x)$ when $w]x, n] \in L$ (*S*uffix).

Generic logical correspondence

Logical Correspondence Theorem (Place, Z. '17)

For any basis $\mathcal{C}$, there is a natural set $\mathcal{S}$ of first order predicates, st.

Concatenation hierarchy of basis $\mathcal{C}$
$=$
Quantifier alternation hierarchy over signature $\mathcal{S}$

Generalizes the correspondences discovered for BC and ST hierarchies.

Intuition

For each $L \in \mathcal{C}$, add 4 predicates in addition to $<$ and $a(), b(), \dots$

- $w \models I_L(x, y)$ when $x < y$ and $w]x, y[\in L$ (*Infix*).
- $w \models P_L(y)$ when $w[1, y[\in L$ (*Prefix*).
- $w \models S_L(x)$ when $w]x, n] \in L$ (*Suffix*).
- $w \models W_L$ when $w \in L$ (*Whole word*).

Generic membership theorem

Generic membership Theorem (Place, Z. '17, Place '15)

For any finite basis $\mathcal{C}$, levels $\frac{1}{2}, 1, \frac{3}{2}, \quad \frac{5}{2}$ have decidable membership.

Generic membership theorem

Generic membership Theorem (Place, Z. '17, Place '15)

For any finite basis $\mathcal{C}$, levels $\frac{1}{2}, 1, \frac{3}{2}, \quad \frac{5}{2}$ have decidable membership.

Remember: state of the art went **up to level $\frac{7}{2}$** for ST and BC hierarchies.

Generic membership theorem

Generic membership Theorem (Place, Z. '17, Place '15)

For any finite basis $\mathcal{C}$, levels $\frac{1}{2}, 1, \frac{3}{2}, \quad \frac{5}{2}$ have decidable membership.

Remember: state of the art went **up to level $\frac{7}{2}$** for ST and BC hierarchies.

The alphabet trick...

Languages in ST $\left[\frac{3}{2}\right]$ (Pin and Straubing '85)

Languages of level ST $\left[\frac{3}{2}\right]$ are unions of languages of the form $B_0^* a_1 B_1^* \cdots a_n B_n^*$

$$\mathrm{ST}\left[\frac{3}{2}\right] = \text{level } \frac{1}{2} \text{ with basis } \{B^* \mid B \subseteq A\}.$$

ST[q] is also level $(q-1)$ in another hierarchy with finite basis.

Generic membership theorem

Generic membership Theorem (Place, Z. '17, Place '15)

For any finite basis $\mathcal{C}$, levels $\frac{1}{2}, 1, \frac{3}{2},\quad \frac{5}{2}$ have decidable membership.

Remember: state of the art went **up to level $\frac{7}{2}$** for ST and BC hierarchies.

The alphabet trick...

Languages in ST $\left[\frac{3}{2}\right]$ (Pin and Straubing '85)

Languages of level ST $\left[\frac{3}{2}\right]$ are unions of languages of the form $B_0^* a_1 B_1^* \cdots a_n B_n^*$

$$\mathrm{ST}\left[\frac{3}{2}\right] = \text{level } \frac{1}{2} \text{ with basis } \{B^* \mid B \subseteq A\}.$$

ST[q] is also level $(q-1)$ in another hierarchy with finite basis.

Corollary (by **Alphabet trick**)

In ST hierarchy, levels $\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2},\quad \frac{7}{2}$ have decidable membership

Generic membership theorem

Generic membership Theorem (Place, Z. '17, Place '15)

For any finite basis $\mathcal{C}$, levels $\frac{1}{2}, 1, \frac{3}{2}, \quad \frac{5}{2}$ have decidable membership.

Remember: state of the art went **up to level $\frac{7}{2}$** for ST and BC hierarchies.

The alphabet trick...

Languages in ST $\left[\frac{3}{2}\right]$ (Pin and Straubing '85)

Languages of level ST $\left[\frac{3}{2}\right]$ are unions of languages of the form $B_0^* a_1 B_1^* \cdots a_n B_n^*$

$$\text{ST}\left[\tfrac{3}{2}\right] = \text{level } \tfrac{1}{2} \text{ with basis } \{B^* \mid B \subseteq A\}.$$

ST[q] is also level $(q-1)$ in another hierarchy with finite basis.

Corollary (by **Alphabet trick**)

In ST and BC hierarchy, levels $\frac{1}{2}, 1, \frac{3}{2}, 2, \frac{5}{2}, \quad \frac{7}{2}$ have decidable membership

Recap

- Generic construction process for concatenation hierarchies.
- Generic logical correspondence.
- Generic strictness theorem.
- Generic membership theorem.

Recap

- Generic construction process for concatenation hierarchies.
- Generic logical correspondence.
- Generic strictness theorem.
- Generic membership theorem.

Recent results required solving **harder problems** than membership.

Beyond Membership: Separation

Beyond membership: Separation

Recent results required solving **harder problems** than membership.

Motivation:

- Classe $\mathcal{C}$ with decidable membership.
- Class $\mathrm{Op}(\mathcal{C})$ built on top of $\mathcal{C}$ with undecidable membership.

Beyond membership: Separation

Recent results required solving **harder problems** than membership.

Motivation:

- Classe $\mathcal{C}$ with decidable membership.
- Class $\mathrm{Op}(\mathcal{C})$ built on top of $\mathcal{C}$ with undecidable membership.

Nice idea, Henckell and Rhodes '88

Prove more on $\mathcal{C}$ to recover membership decidability for $\mathrm{Op}(\mathcal{C})$.

Nice statement, Almeida '96

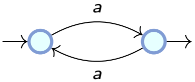
Almeida'96: a problem introduced by Henckell can be formulated as **separation**.

Beyond membership: Separation

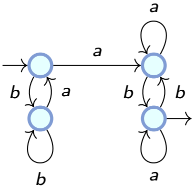
Decide the following problem:

Take **2** regular languages L_1, L_2

L_1



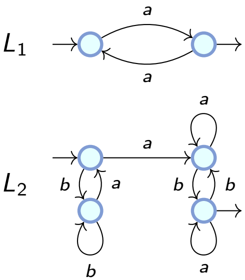
L_2



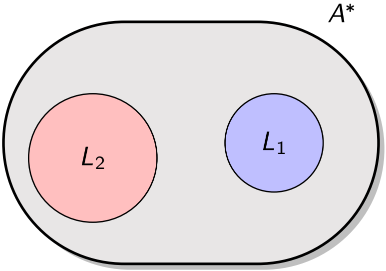
Beyond membership: Separation

Decide the following problem:

Take **2** regular languages L_1, L_2



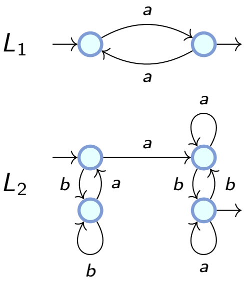
Can L_1 be separated from L_2 with a language from $\mathcal{C}$?



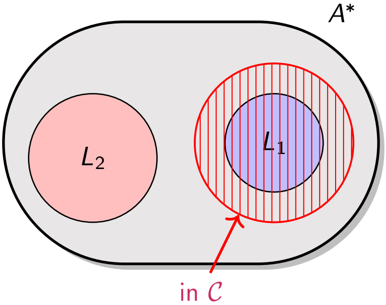
Beyond membership: Separation

Decide the following problem:

Take **2** regular languages L_1, L_2



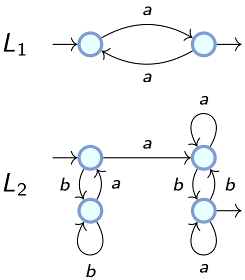
$\Longrightarrow$ Can L_1 be separated from L_2 with a language from $\mathcal{C}$?



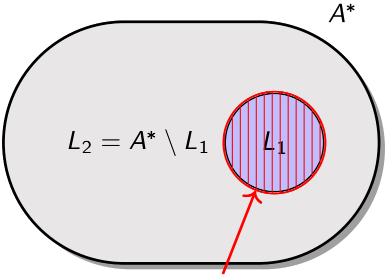
Beyond membership: Separation

Decide the following problem:

Take **2** regular languages L_1, L_2



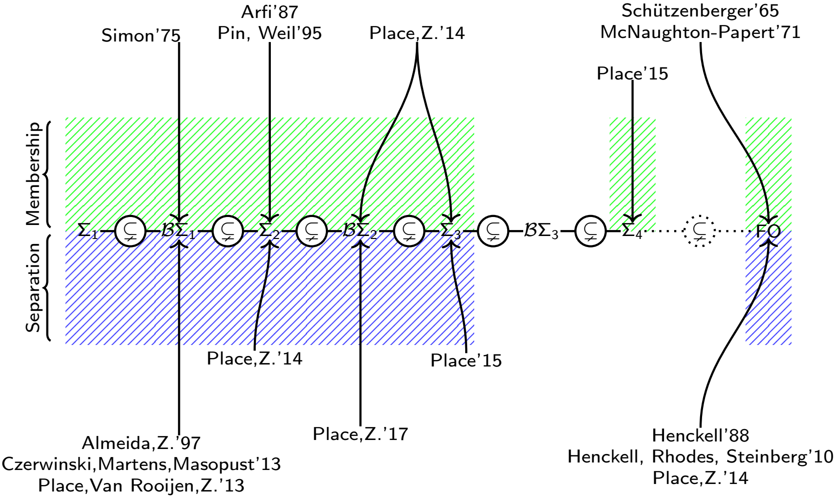
Can L_1 be separated from L_2 with a language from $\mathcal{C}$?



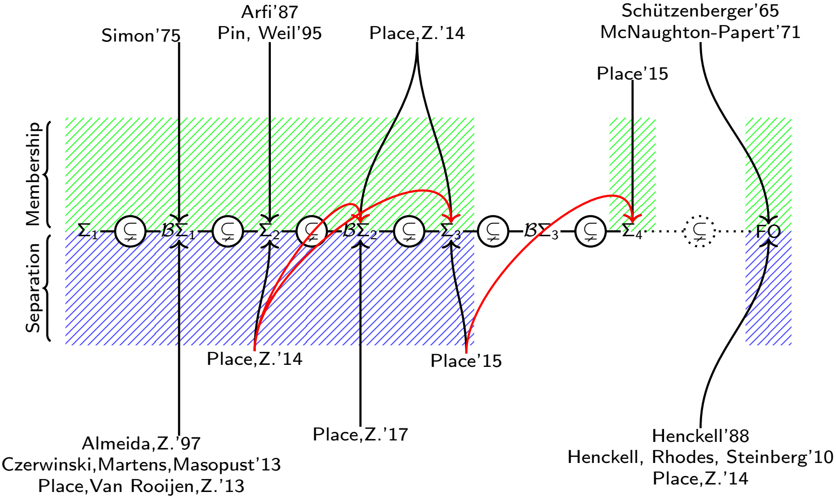
$\mathcal{C}$-separable from complement
$\Leftrightarrow$
in $\mathcal{C}$

Membership can be formally reduced to separation

Separation for classical hierarchies



Separation for classical hierarchies



Some membership algorithms come from separation algorithms for simpler levels

Separation results for generic hierarchies

Generic Separation Theorem (Place, Z. '17, Place '15)

In any hierarchy of **finite basis**, separation is decidable for levels $\frac{1}{2}, 1, \frac{3}{2}$.

Separation results for generic hierarchies

Generic Separation Theorem (Place, Z. '17, Place '15)

In any hierarchy of **finite basis**, separation is decidable for levels $\frac{1}{2}, 1, \frac{3}{2}$.

Jump Theorem (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

Separation results for generic hierarchies

Generic Separation Theorem (Place, Z. '17, Place '15)

In any hierarchy of **finite basis**, separation is decidable for levels $\frac{1}{2}, 1, \frac{3}{2}$.

Jump Theorem (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

Enrichment Theorem (Place, Z. '15)

Separation for a level in the enriched hierarchy (ie, BC)
reduces to
Separation for the same level in the order hierarchy (ie, ST).

Separation results for generic hierarchies

Generic Separation Theorem (Place, Z. '17, Place '15)

In any hierarchy of **finite basis**, separation is decidable for levels $\frac{1}{2}, 1, \frac{3}{2}$.

Jump Theorem (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

Enrichment Theorem (Place, Z. '15)

Separation for a level in the enriched hierarchy (ie, BC)
reduces to
Separation for the same level in the order hierarchy (ie, ST).

Corollary (by Alphabet trick + Enrichment)

Levels $\frac{1}{2}, 1, \frac{3}{2}$, 2 and $\frac{5}{2}$ have decidable separation in ST and BC hierarchies.

The Jump Theorem

Jump Theorem for quotienting lattices (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

The Jump Theorem

Jump Theorem for quotienting lattices (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

The Jump Theorem on Automata

A regular language **is in** $\mathcal{C}\left[\mathbf{n} + \frac{\mathbf{1}}{\mathbf{2}}\right]$ iff its minimal automaton has **no pattern**:

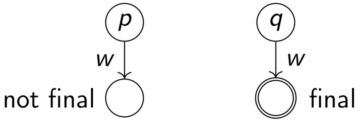
The Jump Theorem

Jump Theorem for quotienting lattices (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

The Jump Theorem on Automata

A regular language **is in $\mathcal{C}\left[\mathbf{n} + \frac{1}{2}\right]$** iff its minimal automaton has **no pattern**:



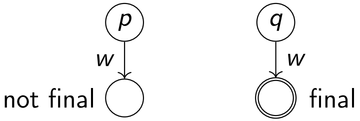
The Jump Theorem

Jump Theorem for quotienting lattices (Place, Z. '15)

Membership for level $n + \frac{1}{2}$ reduces to separation for level $n - \frac{1}{2}$.

The Jump Theorem on Automata

A regular language **is in $\mathcal{C}\left[\mathbf{n} + \frac{1}{2}\right]$** iff its minimal automaton has **no pattern**:



where $L_{p,q}$ is **not $\mathcal{C}\left[\mathbf{n} - \frac{1}{2}\right]$-separable** from $L_{p,p} \cap L_{q,q}$

$$L_{p,q} = \{w \mid p \xrightarrow{w} q\}$$

Recap

Current knowledge is captured by these 3 generic results:

1. Separation theorem
   $\mathcal{C}$ finite $\Rightarrow$ separation decidable for $Pol(\mathcal{C})$, $BPol(\mathcal{C})$, and $Pol(BPol(\mathcal{C}))$.
   In particular, unable to deal with 2 levels of complement.

2. Jump theorem
   $\mathcal{C}$-separation decidable $\Rightarrow$ $Pol(\mathcal{C})$-membership decidable.

3. Enrichment theorem.

Generalizing separation

Beyond Separation: Covering

- If $\mathcal{A}$ is the minimal DFA for L

$$L \in \mathcal{C} \qquad \text{iff} \qquad \forall p, q, \qquad L_{p,q} \in \mathcal{C}$$

Comes from reasonable closure properties of usual classes.

Beyond Separation: Covering

- If $\mathcal{A}$ is the minimal DFA for L

$$L \in \mathcal{C} \qquad \text{iff} \qquad \forall p, q, \qquad L_{p,q} \in \mathcal{C}$$

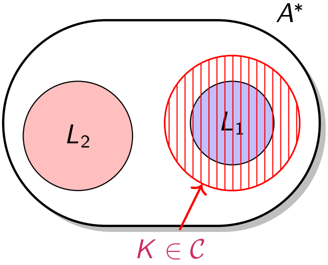
Comes from reasonable closure properties of usual classes.

- We should actually consider a **set** of languages as **input**.
$\Longrightarrow$ natural to extend separation to **several input languages**.

The Covering Problem

- **Recall:** L_1, L_2 $\mathcal{C}$-separable if

$$\exists K \in \mathcal{C}, \qquad L_1 \subseteq K \text{ and } L_2 \cap K = \emptyset$$



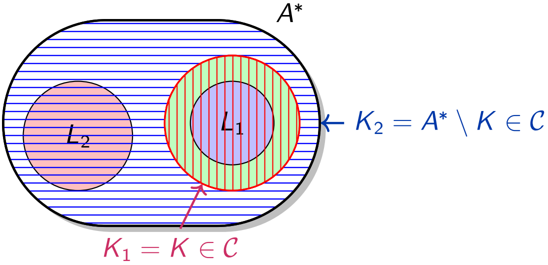
The Covering Problem

- **Recall:** L_1, L_2 $\mathcal{C}$-separable if

$$\exists K \in \mathcal{C}, \qquad L_1 \subseteq K \text{ and } L_2 \cap K = \emptyset$$

- If $\mathcal{C}$ is closed under complement, same as:

$\exists K_1, K_2 \in \mathcal{C}$



The Covering Problem

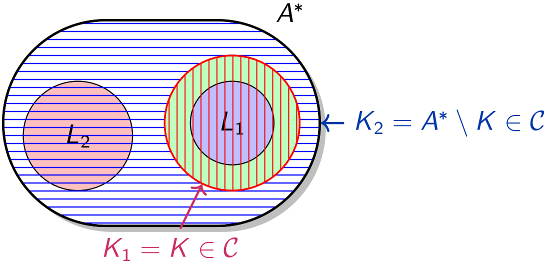
- **Recall:** L_1, L_2 $\mathcal{C}$-separable if

$$\exists K \in \mathcal{C}, \qquad L_1 \subseteq K \text{ and } L_2 \cap K = \emptyset$$

- If $\mathcal{C}$ is closed under complement, same as:

$\exists K_1, K_2 \in \mathcal{C}$

$$\begin{cases} L_1 \cup L_2 \subseteq K_1 \cup K_2 \\ \\ K_1 \text{ does not intersect both } L_1 \text{ and } L_2 \\ K_2 \text{ does not intersect both } L_1 \text{ and } L_2 \end{cases}$$



The Covering Problem

- **Recall:** L_1, L_2 $\mathcal{C}$-separable if

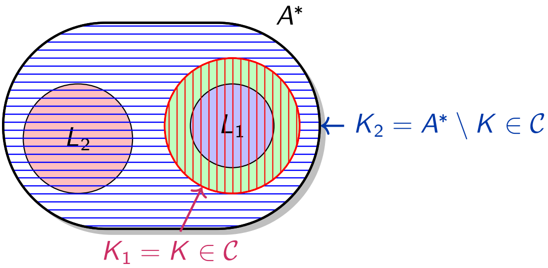
$$\exists K \in \mathcal{C}, \qquad L_1 \subseteq K \text{ and } L_2 \cap K = \emptyset$$

- If $\mathcal{C}$ is closed under complement, same as:

$\exists K_1, K_2 \in \mathcal{C}$

$$\left\{\begin{array}{l} L_1 \cup L_2 \subseteq K_1 \cup K_2 \\ \\ K_1 \text{ does not intersect both } L_1 \text{ and } L_2 \\ K_2 \text{ does not intersect both } L_1 \text{ and } L_2 \end{array}\right.$$

$\}$ $\{K_1, K_2\}$ **covers** $\{L_1, L_2\}$...



The Covering Problem

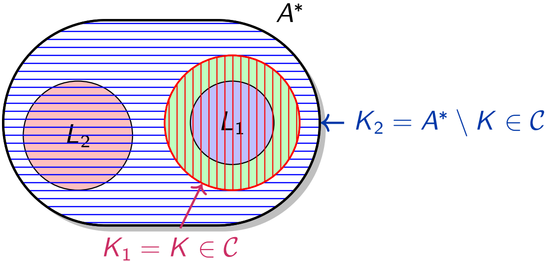
- **Recall:** L_1, L_2 $\mathcal{C}$-separable if

$$\exists K \in \mathcal{C}, \qquad L_1 \subseteq K \text{ and } L_2 \cap K = \emptyset$$

- If $\mathcal{C}$ is closed under complement, same as:

$\exists K_1, K_2 \in \mathcal{C}$

$$\begin{cases} L_1 \cup L_2 \subseteq K_1 \cup K_2 & \big\} \{K_1, K_2\} \textbf{ covers } \{L_1, L_2\}\ldots \\ \left.\begin{array}{l} K_1 \text{ does not intersect both } L_1 \text{ and } L_2 \\ K_2 \text{ does not intersect both } L_1 \text{ and } L_2 \end{array}\right\} & \ldots\textbf{optimally} \text{ wrt separation} \end{cases}$$



$\mathcal{C}$-Covers

- $\mathbf{L} = \{L_1, \ldots, L_n\}$ = set of languages.
- $\mathcal{C}$-**cover** of $\mathbf{L}$ = finite set of languages $\mathbf{K} = \{K_1, \ldots, K_m\}$ from $\mathcal{C}$ st.

$$L_1 \cup \cdots \cup L_n \subseteq K_1 \cup \cdots \cup K_m$$

$\mathcal{C}$-Covers

- $\mathbf{L} = \{L_1, \dots, L_n\}$ = set of languages.
- $\mathcal{C}$-**cover** of $\mathbf{L}$ = finite set of languages $\mathbf{K} = \{K_1, \dots, K_m\}$ from $\mathcal{C}$ st.

$$L_1 \cup \dots \cup L_n \subseteq K_1 \cup \dots \cup K_m$$

- **Note**: When $A^* \in \mathcal{C}$: $\{A^*\}$ is always a $\mathcal{C}$-cover of $\{L_1, \dots, L_n\}$.

$\mathcal{C}$-Covers

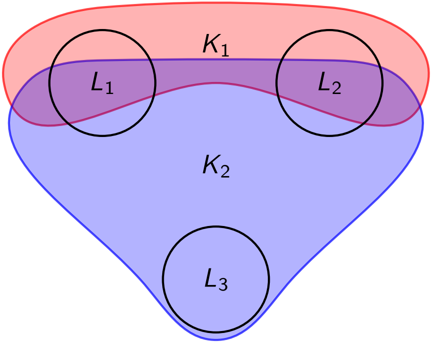
- $\mathbf{L} = \{L_1, \dots, L_n\}$ = set of languages.
- $\mathcal{C}$-**cover** of $\mathbf{L}$ = finite set of languages $\mathbf{K} = \{K_1, \dots, K_m\}$ from $\mathcal{C}$ st.

$$L_1 \cup \cdots \cup L_n \subseteq K_1 \cup \cdots \cup K_m$$

- **Note**: When $A^* \in \mathcal{C}$: $\{A^*\}$ is always a $\mathcal{C}$-cover of $\{L_1, \dots, L_n\}$.

- **Goal**: Measure **how good** a cover is at "separating" input set $\mathbf{L}$.
  We define the **imprint** of $\mathbf{K}$ on $\mathbf{L}$ for this.

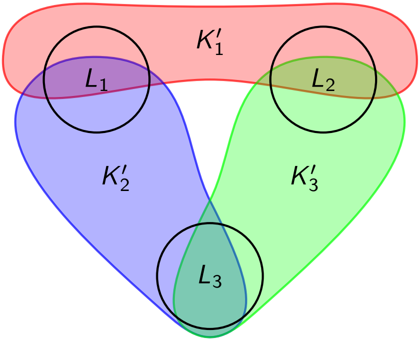
Imprint = Quality of a $\mathcal{C}$-Cover — Example 1



$\mathcal{C}$-Cover $\mathbf{K} = \{K_1, K_2\}$

Imprint $\mathcal{I}[\mathbf{L}](\mathbf{K}) = \Big\{\text{all subsets of } \{L_1, L_2, L_3\}\Big\}$

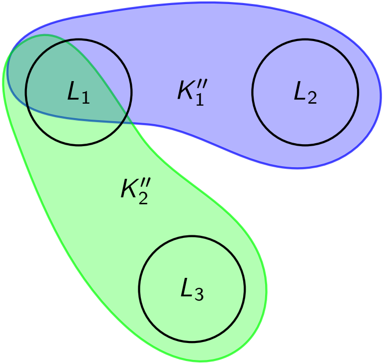
Imprint = Quality of a $\mathcal{C}$-Cover — Example 2 (better)



$\mathcal{C}$-Cover $\mathbf{K}' = \{K_1', K_2', K_3'\}$

Imprint $\mathcal{I}[\mathbf{L}](\mathbf{K}') = \Big\{$all subsets of $\{L_1, L_2, L_3\}$ but $\{L_1, L_2, L_3\}\Big\}$

Imprint = Quality of a $\mathcal{C}$-Cover — Example 3 (even better)



$\mathcal{C}$-Cover $\mathbf{K}'' = \{K_1'', K_2''\}$

Imprint $\mathcal{I}[\mathbf{L}](\mathbf{K}'') = \Big\{$all subsets of $\{L_1, L_2, L_3\}$ but $\{L_1, L_2, L_3\}$ and $\{L_2, L_3\}\Big\}$

Recap: Quality of a $\mathcal{C}$-Cover

- Goal: Measure **how good** a cover is at “separating” an input set.
- Captured by **imprint** of **K** on **L**.
- The smaller the imprint, the better.

Optimal $\mathcal{C}$-covers

- A $\mathcal{C}$-cover $\mathbf{K}$ is optimal if **it has minimal imprint**.

Optimal $\mathcal{C}$-covers

- A $\mathcal{C}$-cover $\mathbf{K}$ is optimal if **it has minimal imprint.**

Example

- $\mathcal{C}$ = Boolean algebra generated by languages A^*aA^* for $a \in A$.
- What is a $\mathcal{C}$-optimal imprint of $\mathbf{L} = \{(ab)^+, (ba)^+, (ac)^+\}$?

Optimal $\mathcal{C}$-covers

- A $\mathcal{C}$-cover **K** is optimal if **it has minimal imprint.**

Example

- $\mathcal{C}$ = Boolean algebra generated by languages A^*aA^* for $a \in A$.
- What is a $\mathcal{C}$-optimal imprint of $\mathbf{L} = \{(ab)^+, (ba)^+, (ac)^+\}$?

Existence Lemma

If $\mathcal{C}$ is closed under finite intersection, there exists an optimal cover.

- Trivial, but non-constructive proof.

Optimal $\mathcal{C}$-covers

- A $\mathcal{C}$-cover **K** is optimal if **it has minimal imprint.**

Example

- $\mathcal{C}$ = Boolean algebra generated by languages A^*aA^* for $a \in A$.
- What is a $\mathcal{C}$-optimal imprint of $\mathbf{L} = \{(ab)^+, (ba)^+, (ac)^+\}$?

Existence Lemma

If $\mathcal{C}$ is closed under finite intersection, there exists an optimal cover.

- Trivial, but non-constructive proof.
- Optimal cover not unique, but optimal imprint wrt. $\mathcal{C}$ is unique.

Optimal $\mathcal{C}$-covers

- A $\mathcal{C}$-cover $\mathbf{K}$ is optimal if **it has minimal imprint.**

Example

- $\mathcal{C}$ = Boolean algebra generated by languages A^*aA^* for $a \in A$.
- What is a $\mathcal{C}$-optimal imprint of $\mathbf{L} = \{(ab)^+, (ba)^+, (ac)^+\}$?

Existence Lemma

If $\mathcal{C}$ is closed under finite intersection, there exists an optimal cover.

- Trivial, but non-constructive proof.
- Optimal cover not unique, but optimal imprint wrt. $\mathcal{C}$ is unique.
- $\mathcal{C}$-Optimal imprints capture more information than $\mathcal{C}$-separation.

The $\mathcal{C}$-Covering Problem

$\mathcal{C}$-**Optimal imprint:** $\mathcal{I}_{\mathcal{C}}[\mathbf{L}] \stackrel{\text{def}}{=} \mathcal{I}[\mathbf{L}](\mathbf{K})$ for any optimal $\mathcal{C}$-cover $\mathbf{K}$ of $\mathbf{L}$.

Theorem (Place Z.'16)

Let $\mathcal{C}$ be a Boolean algebra and $\mathbf{L}$ be a finite set of languages.
Given $L_1, L_2 \in \mathbf{L}$, **TFAE**:

1. L_1 and L_2 are $\mathcal{C}$-separable.
2. $\{L_1, L_2\} \notin \mathcal{I}_{\mathcal{C}}[\mathbf{L}]$.

What did we gain?

Contrary to separation information alone, $\mathcal{I}_{\mathcal{C}}[\mathbf{L}]$,

- has nice, generic properties,
- can be computed for all classes where separation is known decidable.

Conclusion

- Overview of membership and separation for concatenation hierarchies.
- Membership is a natural problem, but too rigid as a setting.
- Separation more flexible, easier to cope with for low levels.
- Often requires solving even more general problems.

Thanks!