Schützenberger's Star-Free Theorem and what Followed

Thomas Place

LaBRI, Bordeaux University

March 22, 2016

Schützenberger's Theorem

Regular Languages

Important Reminder: Regular Languages

Finite words over a finite alphabet A:

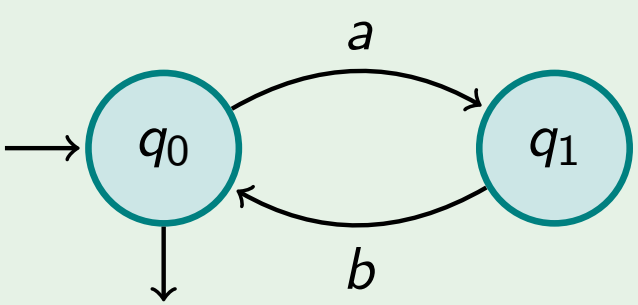
- Word = finite concatenations of letters in A.
  (Example: "*abba*" and "*bbaabba*" for $A = \{a, b\}$).
- Language = set of words. The language of all words is A^*.
- Regular language = language than can be defined by
  a finite automaton or a regular expression.

Regular Languages

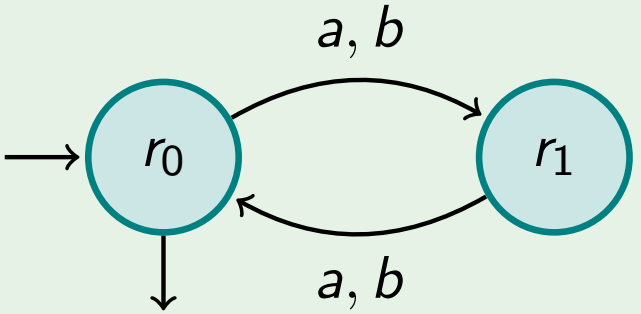
Important Reminder: Regular Languages

Finite words over a finite alphabet A:

- Word = finite concatenations of letters in A.
  (Example: "*abba*" and "*bbaabba*" for $A = \{a, b\}$).
- Language = set of words. The language of all words is A^*.
- Regular language = language than can be defined by
  a finite automaton or a regular expression.



Language $(ab)^*$



Language of words of even length

Star-free Languages: What are they ?

Definition by Induction

- all **finite and co-finite languages** are star-free
  ($\{a\}$ is star-free, $\{b, bab, aa\}$ is star-free, A^* is star-free,...)

Star-free Languages: What are they ?

Definition by Induction

- all **finite and co-finite languages** are star-free
  ($\{a\}$ is star-free, $\{b, bab, aa\}$ is star-free, A^* is star-free,...)
- if L_1, L_2 are star-free then,
  - the **union** $L_1 \cup L_2$ is star-free,
  - the **intersection** $L_1 \cap L_2$ is star-free,
  - the **complement** $\overline{L_1}$ is star-free.

Star-free Languages: What are they ?

Definition by Induction

- all **finite and co-finite languages** are star-free
  ($\{a\}$ is star-free, $\{b, bab, aa\}$ is star-free, A^* is star-free,...)
- if L_1, L_2 are star-free then,
  - the **union** $L_1 \cup L_2$ is star-free,
  - the **intersection** $L_1 \cap L_2$ is star-free,
  - the **complement** $\overline{L_1}$ is star-free.
- if L_1, L_2 are star-free then the **concatenation**, $L_1 \cdot L_2$ is star-free as well.

Star-free Languages: What are they ?

Definition by Induction

- all **finite and co-finite languages** are star-free
  ($\{a\}$ is star-free, $\{b, bab, aa\}$ is star-free, A^* is star-free,...)
- if L_1, L_2 are star-free then,
  - the **union** $L_1 \cup L_2$ is star-free,
  - the **intersection** $L_1 \cap L_2$ is star-free,
  - the **complement** $\overline{L_1}$ is star-free.
- if L_1, L_2 are star-free then the **concatenation**, $L_1 \cdot L_2$ is star-free as well.

The language of words that do **not** contain a "a" is star-free:

$$\overline{A^* \cdot a \cdot A^*}$$

Star-free Languages: What are they ?

Definition by Induction

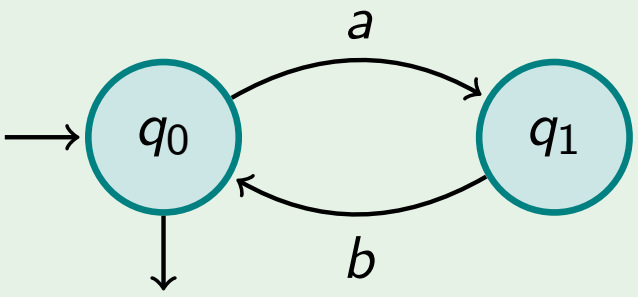
- all **finite and co-finite languages** are star-free ($\{a\}$ is star-free, $\{b, bab, aa\}$ is star-free, A^* is star-free,...)
- if L_1, L_2 are star-free then,
  - the **union** $L_1 \cup L_2$ is star-free,
  - the **intersection** $L_1 \cap L_2$ is star-free,
  - the **complement** $\overline{L_1}$ is star-free.
- if L_1, L_2 are star-free then the **concatenation**, $L_1 \cdot L_2$ is star-free as well.

The language of words that do **not** contain a "a" is star-free:

$$\overline{A^* \cdot a \cdot A^*}$$

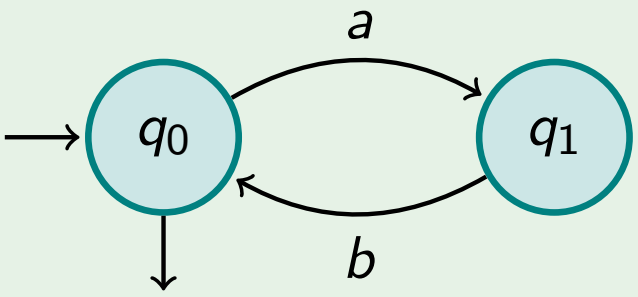
The star-free languages form a **strict** subclass of the regular languages.

Star-free Languages: Examples



Language $(ab)^*$

Star-free Languages: Examples



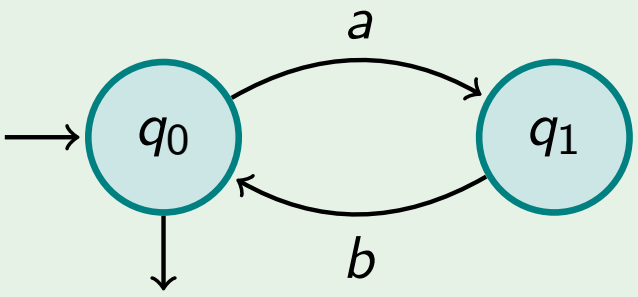
Language $(ab)^*$

This language is star-free

A word is in the language iff

- It does not begin with a "b"
- It does not contain two consecutive "a"
- It does not contain two consecutive "b"
- It does not end with a "a"

Star-free Languages: Examples



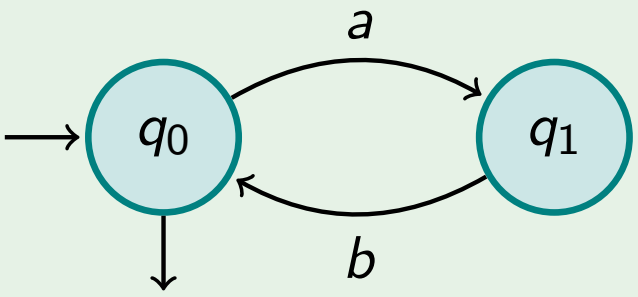
Language $(ab)^*$

This language is star-free

A word is in the language iff

- It does not belong to $b \cdot A^*$
- It does not belong to $A^* \cdot aa \cdot A^*$.
- It does not belong to $A^* \cdot bb \cdot A^*$.
- It does not belong to $A^* \cdot a$

Star-free Languages: Examples



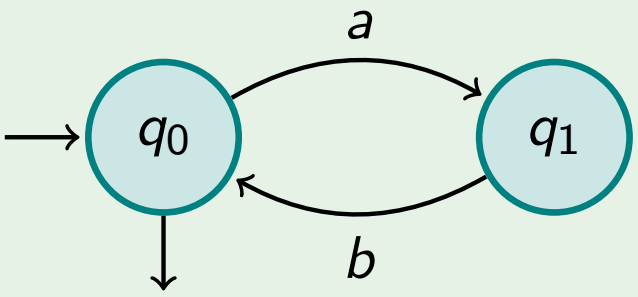
Language $(ab)^*$

This language is star-free

A word is in the language iff

- It belongs to $\overline{b \cdot A^*}$
- It belongs to $\overline{A^* \cdot aa \cdot A^*}$
- It belongs to $\overline{A^* \cdot bb \cdot A^*}$
- It belongs to $\overline{A^* \cdot a}$

Star-free Languages: Examples

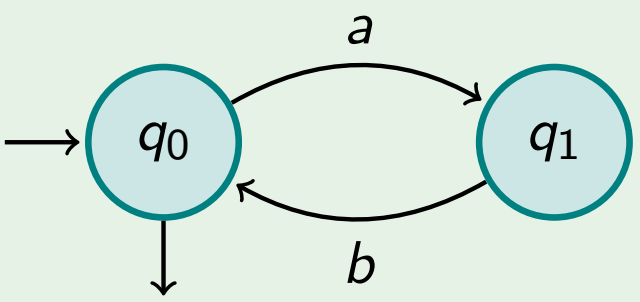


Language $(ab)^*$

This language is star-free

$$\overline{\overline{b \cdot A^*} \cap \overline{A^* \cdot a}} \cap \overline{\overline{A^* \cdot aa \cdot A^*} \cap \overline{A^* \cdot bb \cdot A^*}}$$

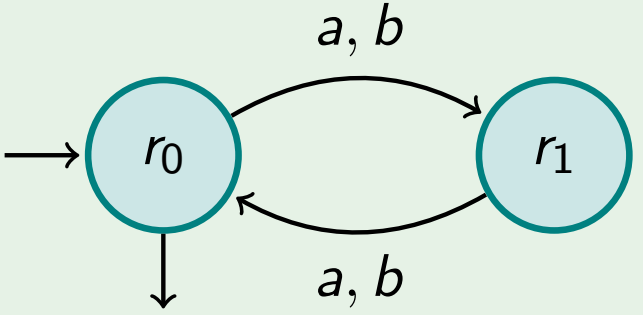
Star-free Languages: Examples



Language $(ab)^*$

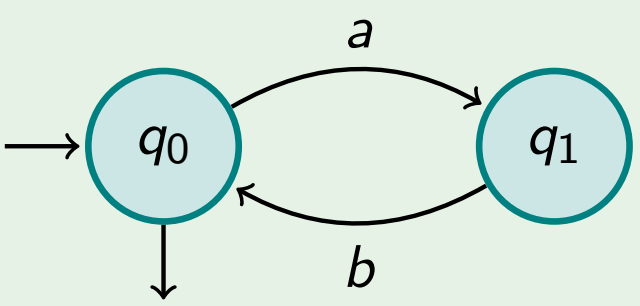
This language is star-free

$$\overline{\overline{b \cdot A^*} \cap \overline{A^* \cdot a}} \cap \overline{\overline{A^* \cdot aa \cdot A^*} \cap \overline{A^* \cdot bb \cdot A^*}}$$



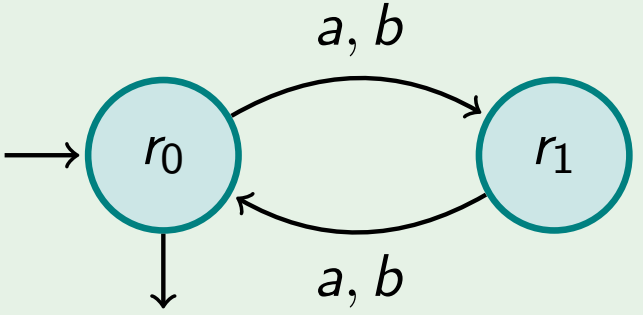
Words of even length

Star-free Languages: Examples



Language $(ab)^*$

This language is star-free

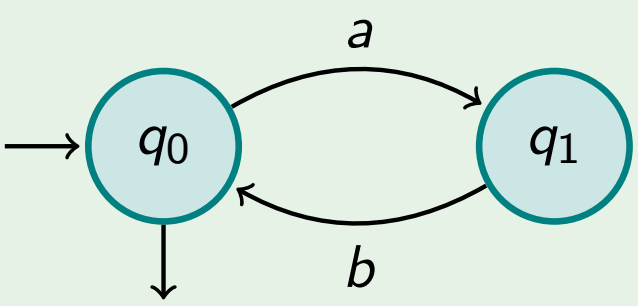


Words of even length

This language is **not** star-free

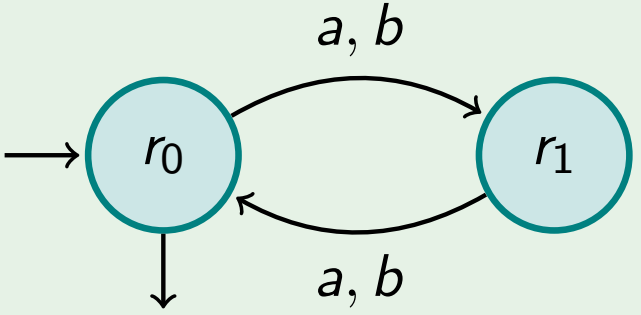
$$\overline{b \cdot A^*} \cap \overline{A^* \cdot aa \cdot A^*} \cap \overline{A^* \cdot a} \cap \overline{A^* \cdot bb \cdot A^*}$$

Star-free Languages: Examples



Language $(ab)^*$

This language is star-free



Words of even length

This language is **not** star-free

$$\overline{b \cdot A^*} \cap \overline{A^* \cdot aa \cdot A^*} \cap \overline{A^* \cdot a} \cap \overline{A^* \cdot bb \cdot A^*}$$

Being star-free is a **semantic property** of a language.
Not obvious from the syntax that defines it.

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

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

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

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

Why is it interesting ? - A First Argument

- The syntactic monoid is a finite computable canonical representation of L
- Aperiodicity is a syntactic and easy to decide property of this algebraic object

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**

Why is it interesting ? - A First Argument

- The syntactic monoid is a finite computable canonical representation of L
- Aperiodicity is a syntactic and easy to decide property of this algebraic object

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** — **semantic**
2. the syntactic monoid of L is **aperiodic** — **syntactic**

Why is it interesting ? - A First Argument

- The syntactic monoid is a finite computable canonical representation of L
- Aperiodicity is a syntactic and easy to decide property of this algebraic object

$\Rightarrow$ **effective** characterization of the star-free languages.

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**
3. the minimal automaton of L is **counter-free** } **syntactic**

(McNaughton-Papert'71).

Why is it interesting ? - A First Argument

- The syntactic monoid is a finite computable canonical representation of L
- Aperiodicity is a syntactic and easy to decide property of this algebraic object

$\Rightarrow$ **effective** characterization of the star-free languages.

The counter-free condition

McNaughton-Papert-Schützenberger

Given L regular, the following are equivalent:

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

Counters

A **counter** is a **non-trivial** sequence of states $q_0, \ldots, q_n$ $(n \geq 1)$ along with a word w such that,

$$q_0 \xrightarrow{w} q_1 \xrightarrow{w} q_2 \xrightarrow{w} \cdots \xrightarrow{w} q_n \xrightarrow{w} q_0$$

The counter-free condition

McNaughton-Papert-Schützenberger

Given L regular, the following are equivalent:

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

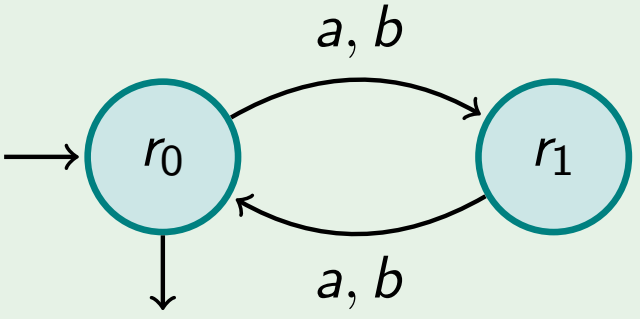
Counters

A **counter** is a **non-trivial** sequence of states $q_0, \ldots, q_n$ $(n \geq 1)$ along with a word w such that,

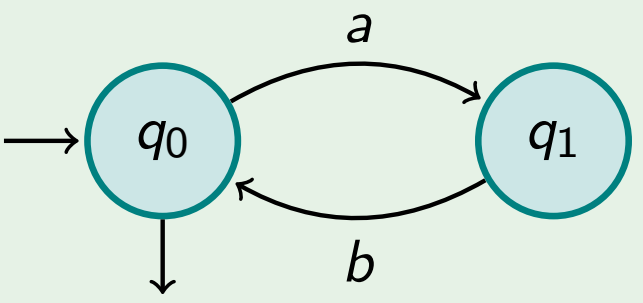
$$q_0 \xrightarrow{w} q_1 \xrightarrow{w} q_2 \xrightarrow{w} \cdots \xrightarrow{w} q_n \xrightarrow{w} q_0$$

Easy to bound the length of the sequences and words we have to look for, which **yields decidability**.

Examples (1)

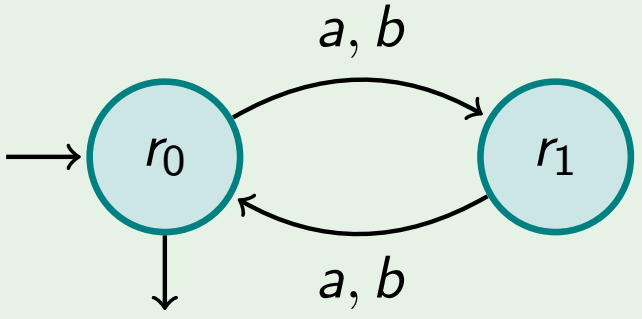


Words of even length

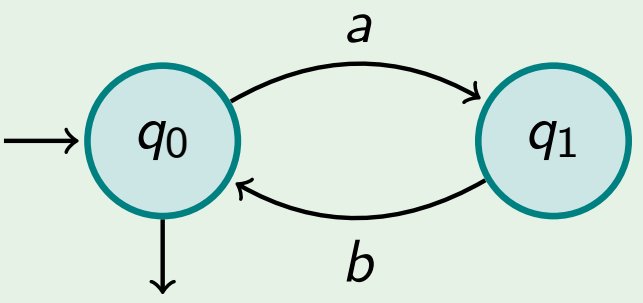


Language $(ab)^*$

Examples (1)



Words of even length

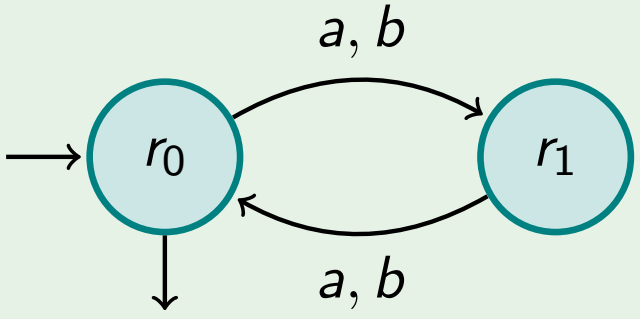


Language $(ab)^*$

Has a counter: not star-free

$$r_0 \xrightarrow{a} r_1 \xrightarrow{a} r_0$$

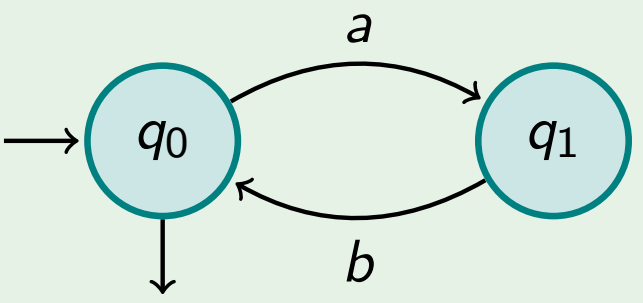
Examples (1)



Words of even length

Has a counter: not star-free

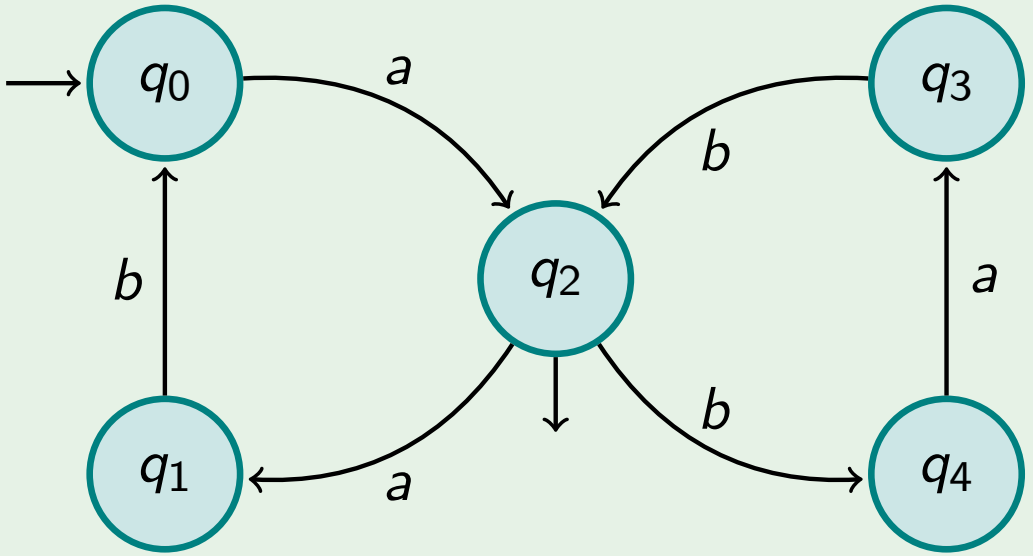
$r_0 \xrightarrow{a} r_1 \xrightarrow{a} r_0$



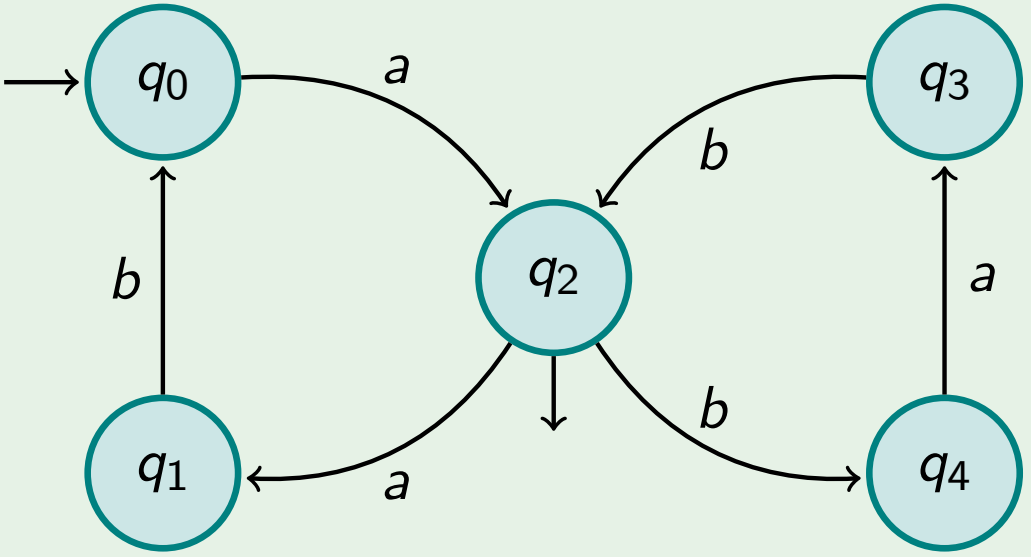
Language $(ab)^*$

Has no counter: star-free

Examples (2)



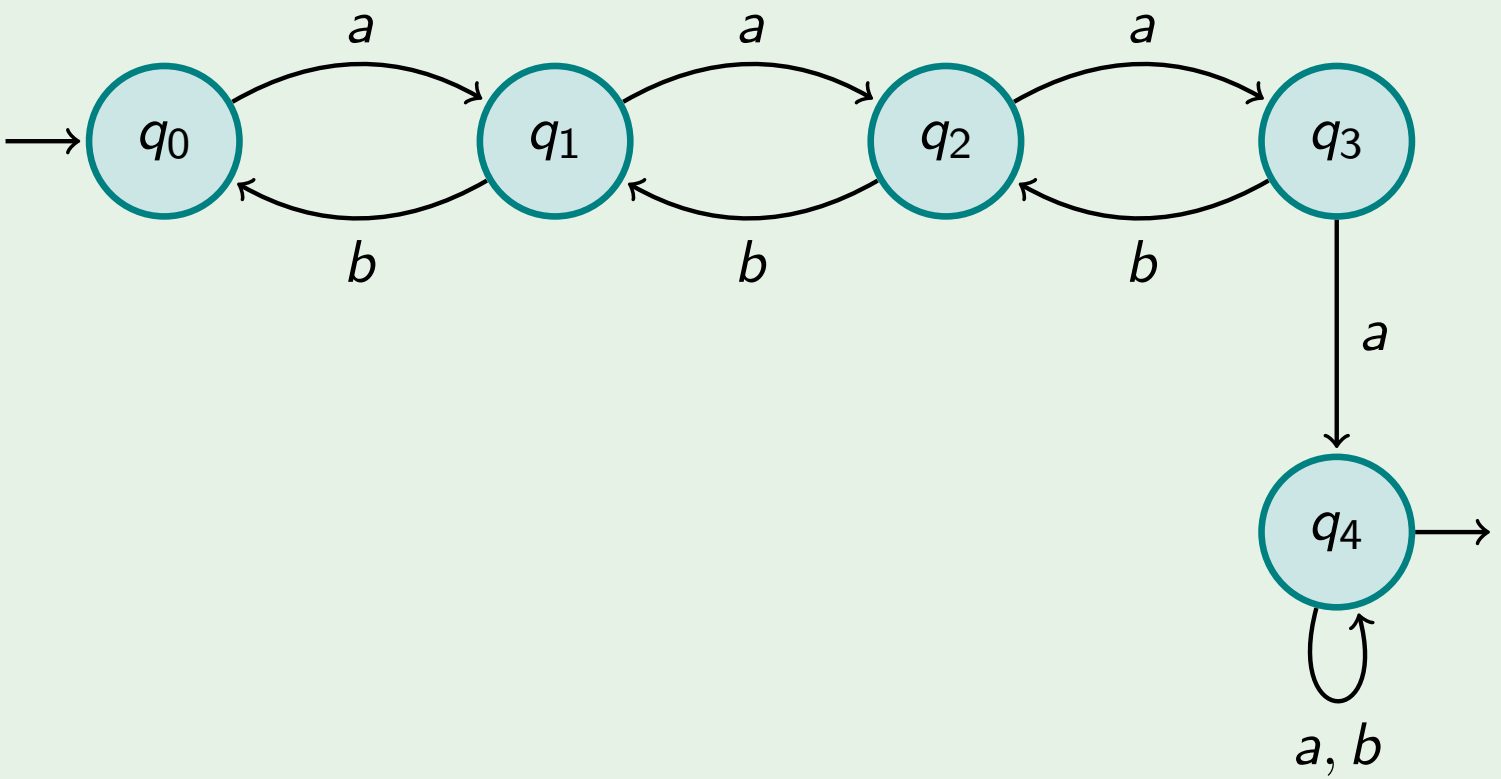
Examples (2)



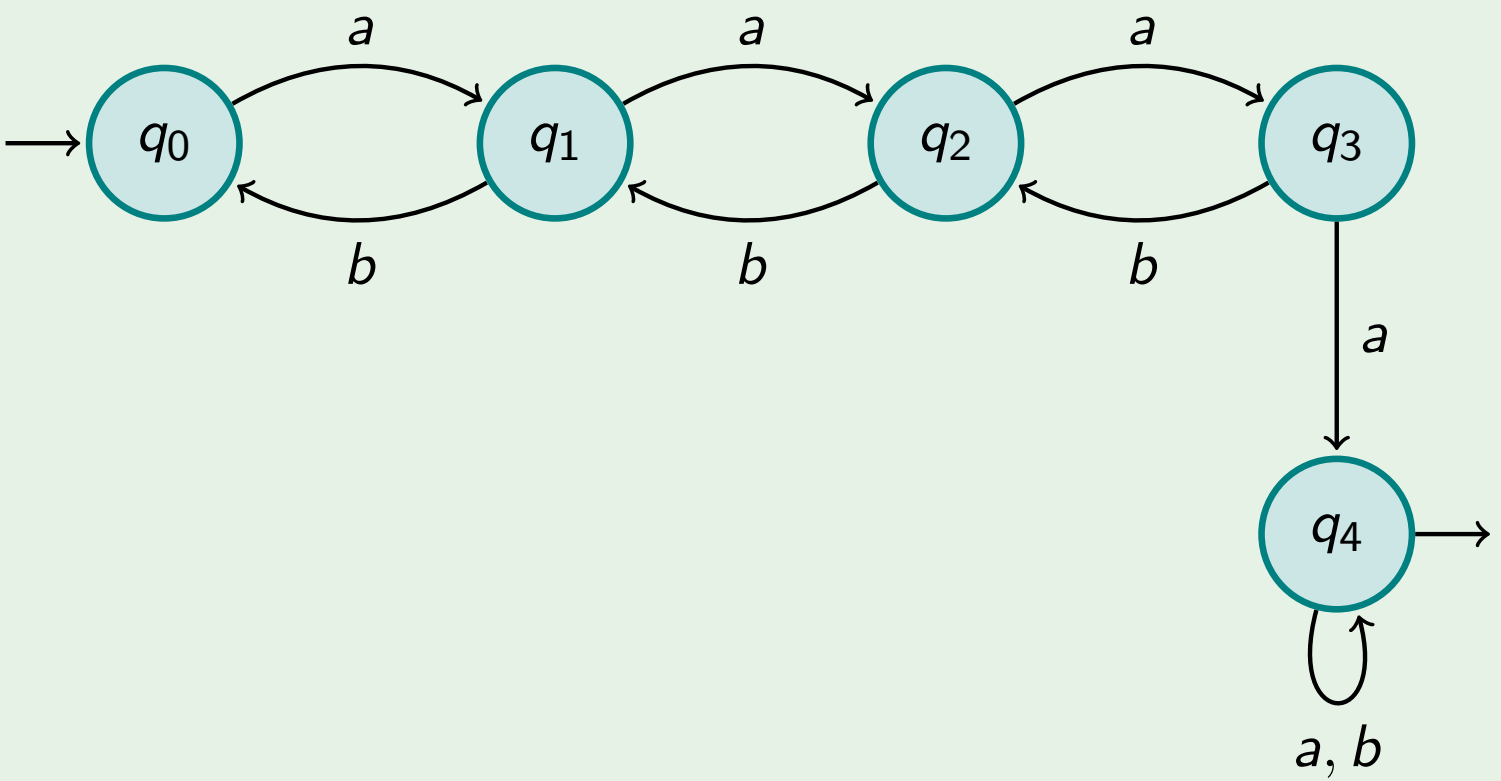
Has a counter: not star-free

$$q_0 \xrightarrow{ab} q_4 \xrightarrow{ab} q_2 \xrightarrow{ab} q_0$$

Examples (3)



Examples (3)



No counter: star-free

Good luck finding a star-free description

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**
3. the minimal automaton of L is **counter-free** (McNaughton-Papert'71). } **syntactic**

Why is it interesting ? - Second Argument

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**
3. the minimal automaton of L is **counter-free** } **syntactic**
   (McNaughton-Papert'71).

Why is it interesting ? - Second Argument

- The proofs of the hard direction $(2, 3 \Rightarrow 1)$ is **constructive**.
- Assuming aperiodicity or counter-free, we have a generic way to **construct a star-free description** of L by induction

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**
3. the minimal automaton of L is **counter-free** } **syntactic**
   (McNaughton-Papert'71).

Why is it interesting ? - Second Argument

- The proofs of the hard direction $(2, 3 \Rightarrow 1)$ is **constructive**.
- Assuming aperiodicity or counter-free, we have a generic way to **construct a star-free description** of L by induction

  $\Rightarrow$ We get **normal forms** for star-free descriptions

Schützenberger's Theorem (Schützenberger'65)

Given a regular language L, the following are equivalent:

1. L is **star-free** } **semantic**
2. the syntactic monoid of L is **aperiodic** } **syntactic**
3. the minimal automaton of L is **counter-free** (McNaughton-Papert'71). } **syntactic**

Why is it interesting ?

- A **Effective** characterization of the star-free languages.
- B **Generic construction** of star-free descriptions.

$\Rightarrow$ Altogether, we **learn a lot** about the star-free languages.

Why should we care about star-free languages ?

Complements to Schützenberger's Theorem

The class of star-free languages is important: it admits other definitions.

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

1. L is **star-free**.

semantic

4. the syntactic monoid of L is **aperiodic**.
5. the minimal automaton of L is **counter-free** (McNaughton-Papert'71).

syntactic

Complements to Schützenberger's Theorem

The class of star-free languages is important: it admits other definitions.

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

1. L is **star-free**.
2. L can be defined in **first-order logic** (FO) (McNaughton-Papert'71).

} **semantic**

4. the syntactic monoid of L is **aperiodic**.
5. the minimal automaton of L is **counter-free** (McNaughton-Papert'71).

} **syntactic**

Complements to Schützenberger's Theorem

The class of star-free languages is important: it admits other definitions.

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

1. L is **star-free**.
2. L can be defined in **first-order logic** (FO) (McNaughton-Papert'71).
3. L can be defined in **linear temporal logic** (LTL) (Kamp'68).

} **semantic**

4. the syntactic monoid of L is **aperiodic**.
5. the minimal automaton of L is **counter-free** (McNaughton-Papert'71).

} **syntactic**

Complements to Schützenberger's Theorem

The class of star-free languages is important: it admits other definitions.

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

1. L is **star-free**.
2. L can be defined in **first-order logic** (FO) (McNaughton-Papert'71).
3. L can be defined in **linear temporal logic** (LTL) (Kamp'68).

} **semantic**

4. the syntactic monoid of L is **aperiodic**.
5. the minimal automaton of L is **counter-free** (McNaughton-Papert'71).

} **syntactic**

The benefits of the theorem apply to both FO and LTL .

More about First-Order Logic (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 that can be quantified.

More about First-Order Logic (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 that can be quantified.
- Given $a \in A$, one can test if x is labeled by a with $a(x)$.
- One can test the linear order: $x < y$.

More about First-Order Logic (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 that can be quantified.
- Given $a \in A$, one can test if x is labeled by a with $a(x)$.
- One can test the linear 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.

This class is the class of star-free languages.

Defining Languages with First-Order Logic (FO)

Why are all star-free languages FO definable ?

- Finite and co-finite languages are FO-definable.
- Union, Intersection and Complement correspond to boolean connectives: $\vee, \wedge, \neg$.

Defining Languages with First-Order Logic (FO)

Why are all star-free languages FO definable ?

- Finite and co-finite languages are FO-definable.
- Union, Intersection and Complement correspond to boolean connectives: $\vee, \wedge, \neg$.
- Concatenation corresponds to existential quantification:

$w \in L_1 \cdot L_2$

There exists a position x in w such that:

- the prefix made up of all positions $y \leq x$ belongs to L_1.
- the suffix made up of all positions $y > x$ belongs to L_2.

Defining Languages with First-Order Logic (FO)

Why are all star-free languages FO definable ?

- Finite and co-finite languages are FO-definable.
- Union, Intersection and Complement correspond to boolean connectives: $\vee, \wedge, \neg$.
- Concatenation corresponds to existential quantification:

$w \in L_1 \cdot L_2$

There exists a position x in w such that:
- the prefix made up of all positions $y \leq x$ belongs to L_1.
- the suffix made up of all positions $y > x$ belongs to L_2.

Proving the converse inclusion is more technical.

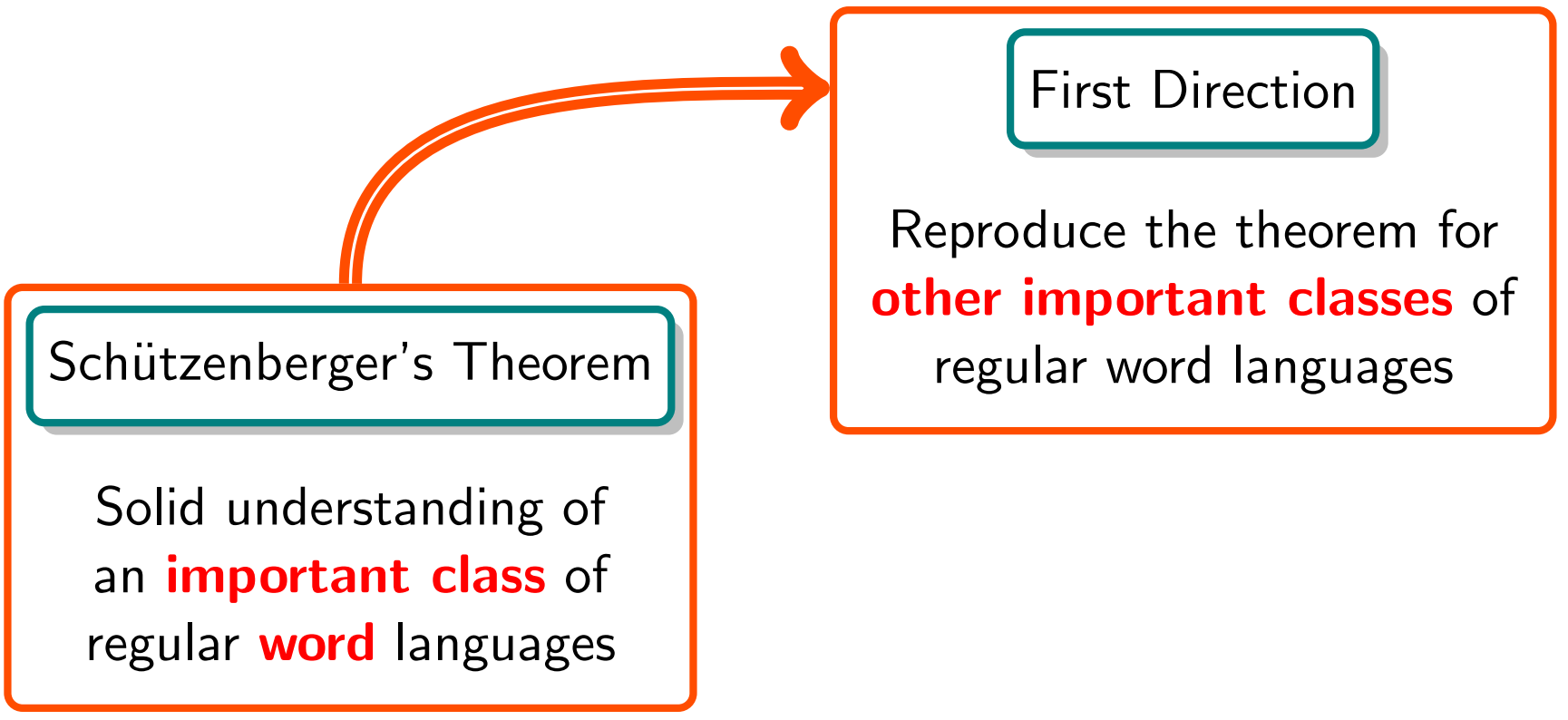
What is Next ?

After Schützenberger's Theorem

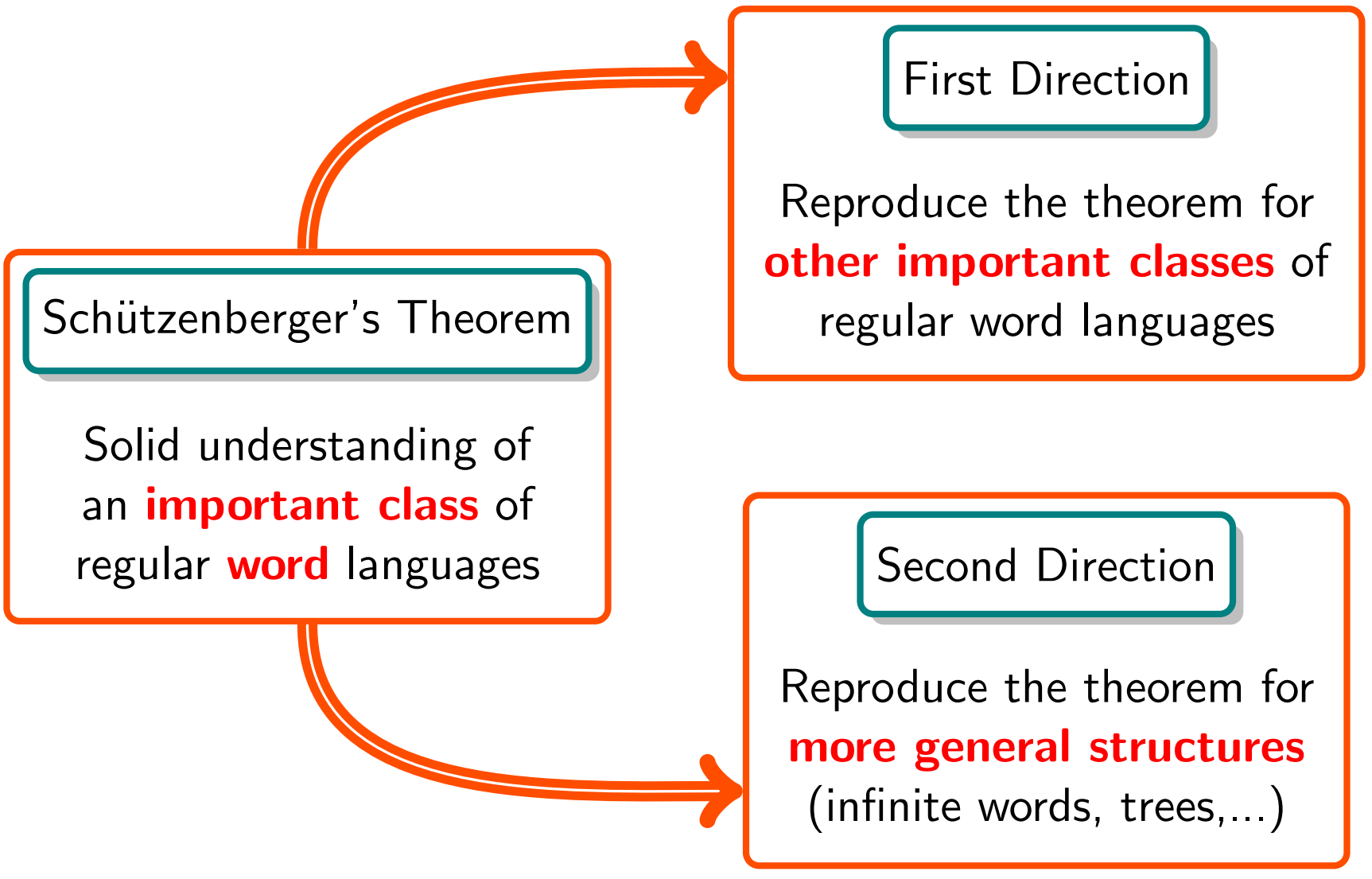
Schützenberger's Theorem

Solid understanding of
an **important class** of
regular **word** languages

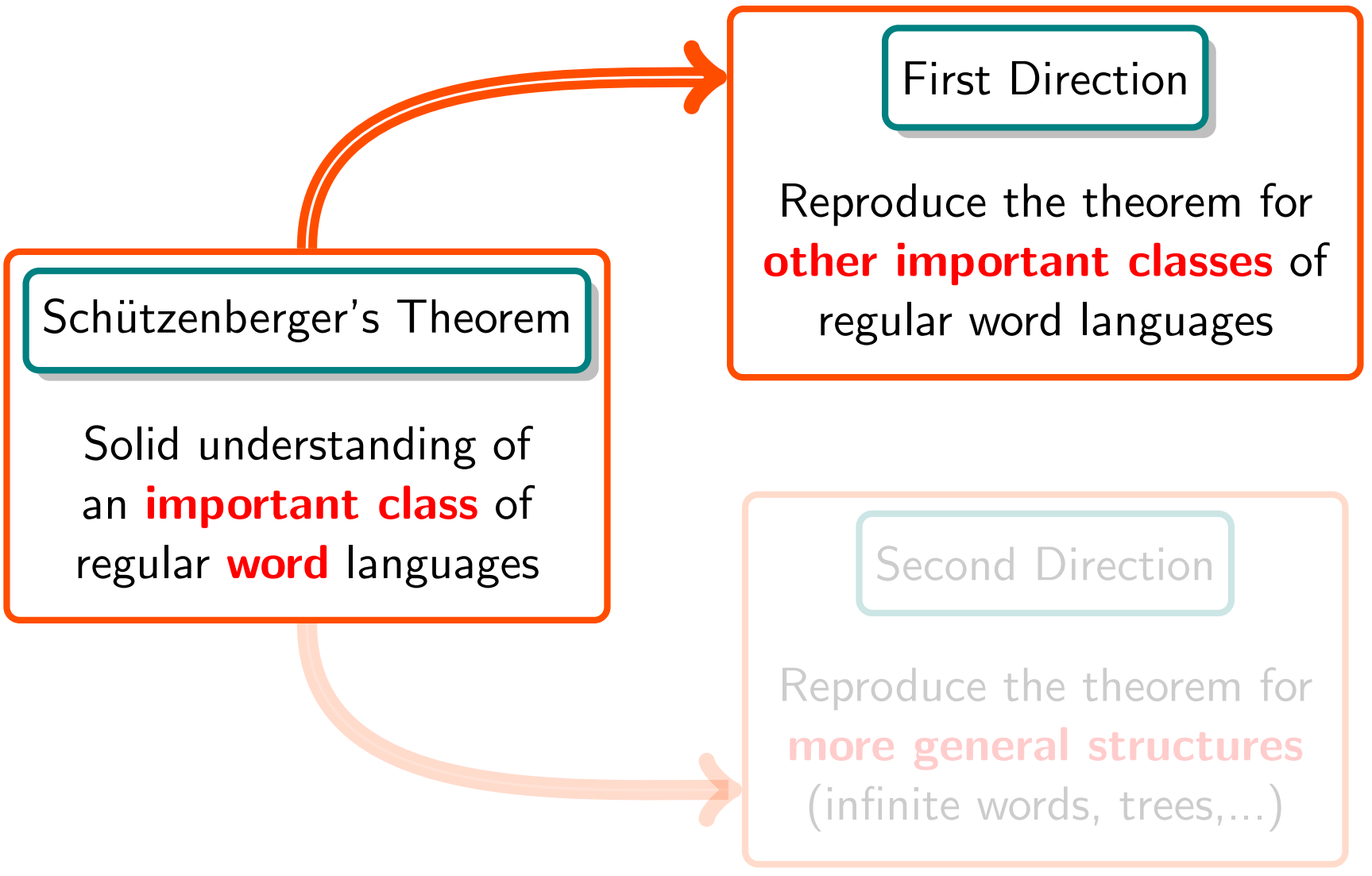
After Schützenberger's Theorem



After Schützenberger's Theorem



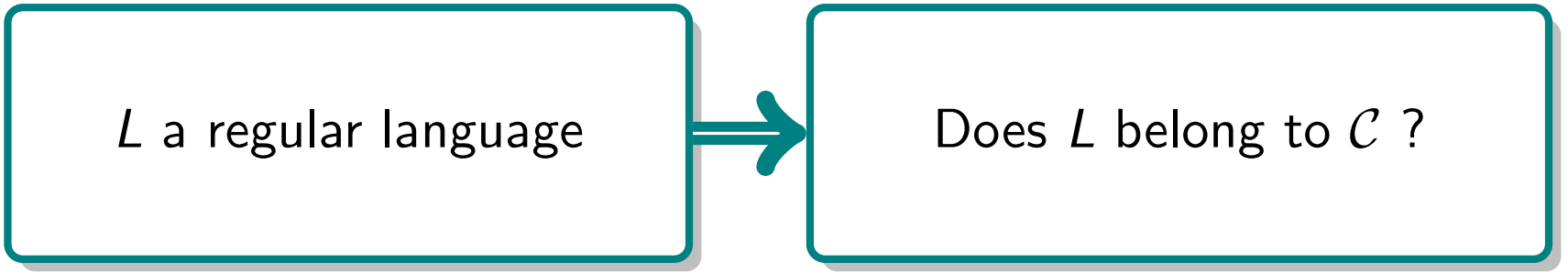
After Schützenberger's Theorem



General Objective: Membership Problem

Reproducing Schützenberger's Theorem for other classes of languages amounts to considering the following problem.

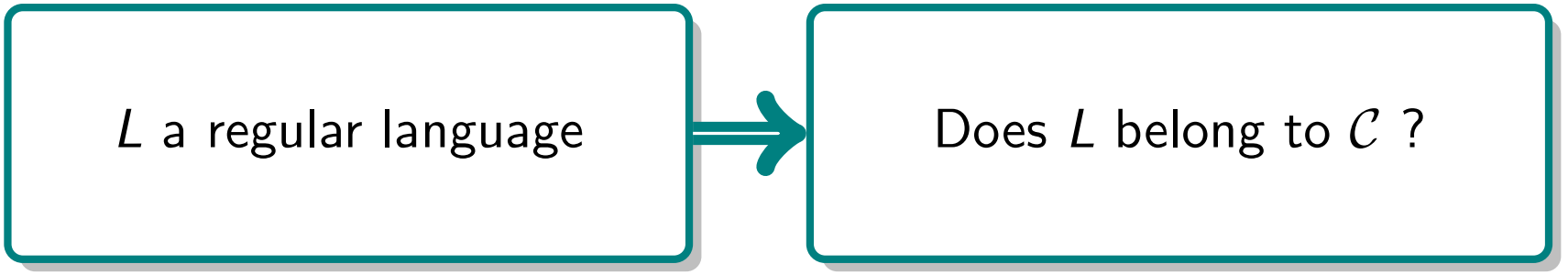
Given such a class $\mathcal{C}$, the goal is to solve the associated **membership problem**:



General Objective: Membership Problem

Reproducing Schützenberger's Theorem for other classes of languages amounts to considering the following problem.

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



Given a particular class, there are two incremental goals to the problem:

- Objective 1: get an **algorithm** that decides it.
- Objective 2: find a generic way to **construct a sentence** that defines L when it exists.

Concatenation Hierarchies and Quantifier Alternation

What makes a star-free description complicated ?

Simple Language

$A^* \cdot aba \cdot A^*$

What makes a star-free description complicated ?

Simple Language

$A^* \cdot aba \cdot A^*$

More Complicated

$A^* \cdot a \cdot (\overline{A^* \cdot aba \cdot A^*} \cap \overline{A^* \cdot bab \cdot A^*}) \cdot a \cdot A^*$

What makes a star-free description complicated ?

Simple Language

$A^* \cdot aba \cdot A^*$

More Complicated

$A^* \cdot a \cdot (\overline{A^* \cdot aba \cdot A^*} \cap \overline{A^* \cdot bab \cdot A^*}) \cdot a \cdot A^*$

Even More Complicated

$A^* \cdot a \cdot \overline{A^* \cdot a \cdot (\overline{A^* \cdot aba \cdot A^*} \cap \overline{A^* \cdot bab \cdot A^*}) \cdot a \cdot A^*} \cdot b \cdot A^*$

What makes a star-free description complicated ?

Simple Language

$A^* \cdot aba \cdot A^*$

More Complicated

$A^* \cdot a \cdot (\overline{A^* \cdot aba \cdot A^*} \cap \overline{A^* \cdot bab \cdot A^*}) \cdot a \cdot A^*$

Even More Complicated

$A^* \cdot a \cdot \overline{A^* \cdot a \cdot (\overline{A^* \cdot aba \cdot A^*} \cap \overline{A^* \cdot bab \cdot A^*}) \cdot a \cdot A^*} \cdot b \cdot A^*$

Complicated = alternation **concatenation** – **complementation**

Formalized by concatenation hierarchies.

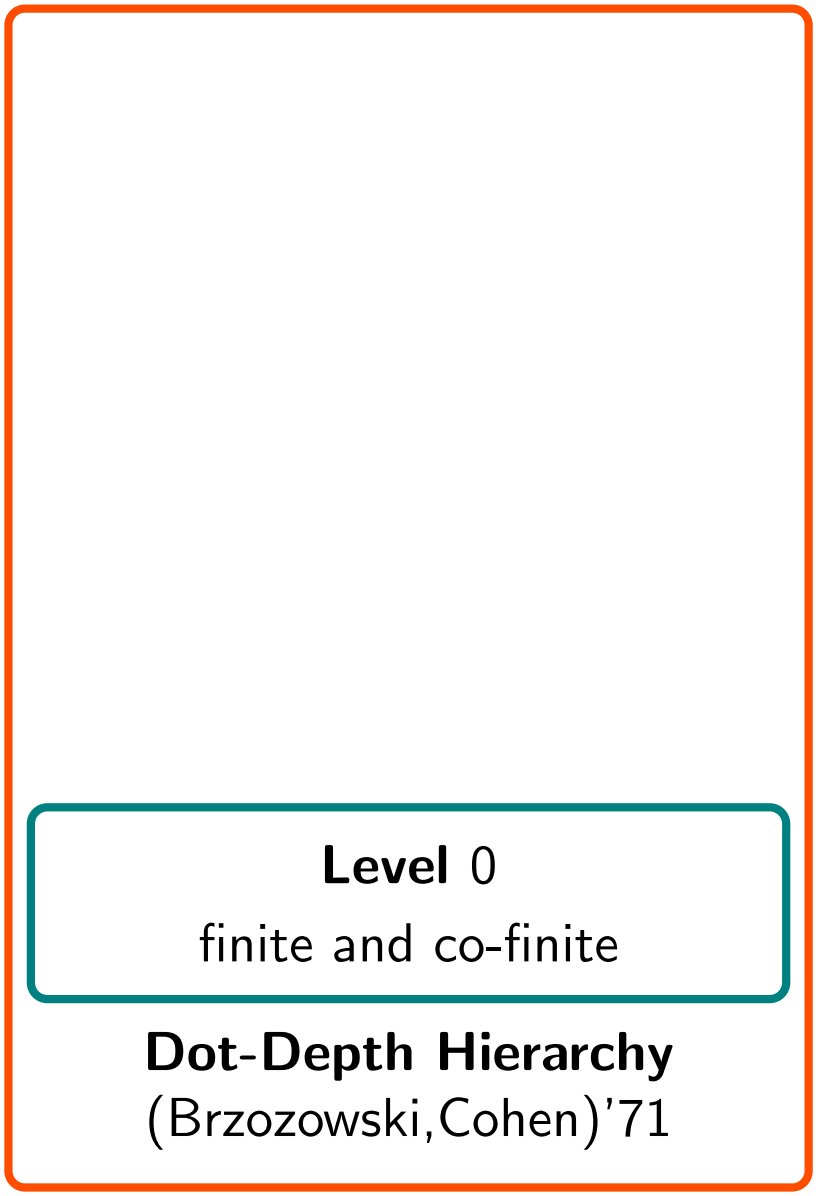
Classifying Star-free Languages (1)

The idea was formalized by two (closely related) infinite hierarchies which classify star-free languages into levels.

$$0 \text{ — } \tfrac{1}{2} \text{ — } 1 \text{ — } \tfrac{3}{2} \text{ — } 2 \text{ — } \tfrac{5}{2} \text{ — } 3 \text{ — } \tfrac{7}{2} \text{ — } 4 \cdots\cdots\cdots\cdots$$

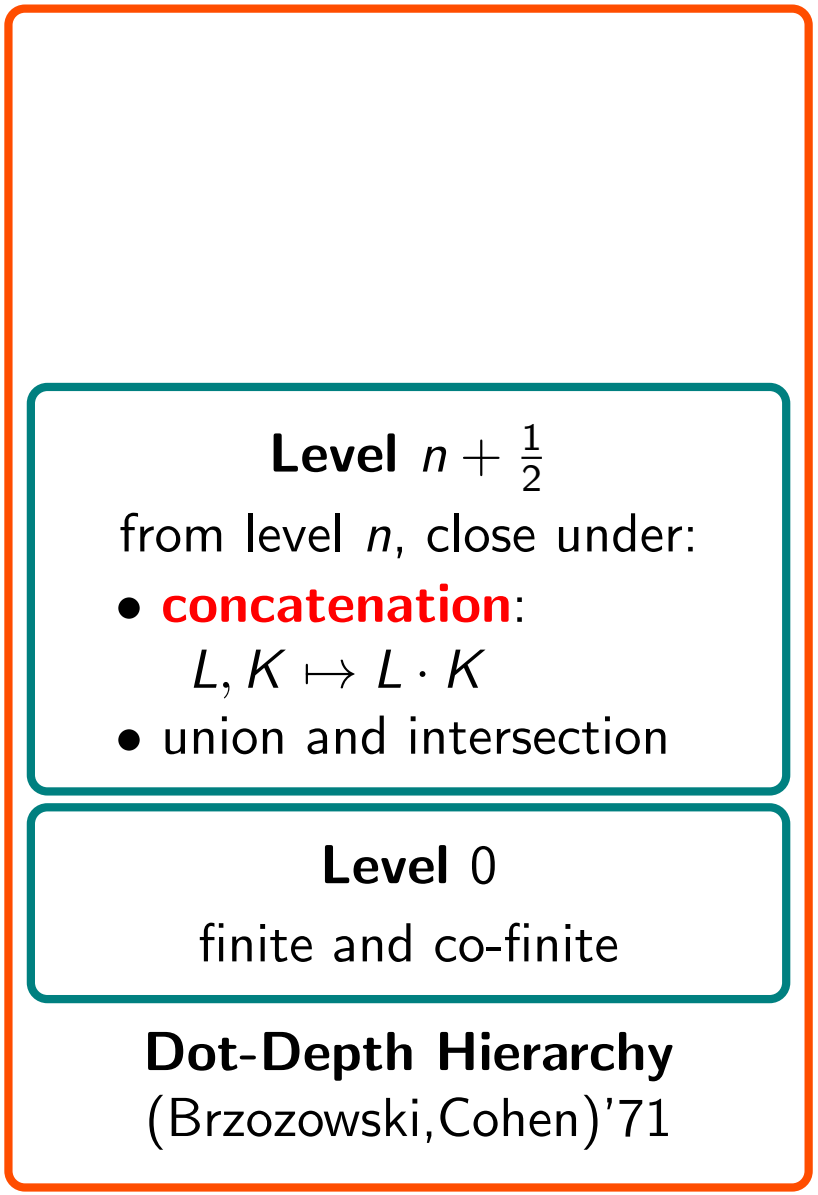
1. one goes from level i to level $i + \frac{1}{2}$ using concatenation.
2. one goes from level $i + \frac{1}{2}$ to level $i + 1$ using complementation.

Classifying Star-free Languages (2)



Examples:
A^*, $\{aa\}$, $\{b\}$, $\{aa, bb\}$,...

Classifying Star-free Languages (2)



Examples for level $\frac{1}{2}$:
$A^* \cdot aa \cdot A^*, A^* \cdot ab \cdot A^* \cup b \cdot A^* \cdot a, \ldots$

Examples:
A^*, $\{aa\}$, $\{b\}$, $\{aa, bb\}, \ldots$

Classifying Star-free Languages (2)

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

from level n, close under:

- **concatenation**:
  $L, K \mapsto L \cdot K$
- union and intersection

Level 0

finite and co-finite

Dot-Depth Hierarchy
(Brzozowski,Cohen)'71

Example for level 1:

$$\overline{b \cdot A^*} \cap \overline{A^* \cdot aa \cdot A^*} \cap \overline{A^* \cdot bb \cdot A^*} \cap \overline{A^* \cdot a}$$

Examples for level $\frac{1}{2}$:

$A^* \cdot aa \cdot A^*, A^* \cdot ab \cdot A^* \cup b \cdot A^* \cdot a, \ldots$

Examples:

A^*, $\{aa\}$, $\{b\}$, $\{aa, bb\}, \ldots$

Classifying Star-free Languages (2)

Level $n + 1$

from level $n + \frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n + \frac{1}{2}$

from level n, close under:

- **concatenation**:
  $L, K \mapsto L \cdot K$
- union and intersection

Level 0

finite and co-finite

Dot-Depth Hierarchy
(Brzozowski,Cohen)'71

Level 0

A^* and $\emptyset$

Straubing-Thérien Hierarchy
(Straubing)'81 (Thérien)'81

Classifying Star-free Languages (2)

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

from level n, close under:

- **concatenation**:
  $L, K \mapsto L \cdot K$
- union and intersection

Level 0

finite and co-finite

Dot-Depth Hierarchy
(Brzozowski,Cohen)'71

Level $n+\frac{1}{2}$

from level n, close under:

- **marked concatenation**:
  $L, K, a \mapsto L \cdot a \cdot K$
- union and intersection

Level 0

A^* and $\emptyset$

Straubing-Thérien Hierarchy
(Straubing)'81 (Thérien)'81

Classifying Star-free Languages (2)

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

from level n, close under:

- **concatenation**:
  $L, K \mapsto L \cdot K$
- union and intersection

Level 0

finite and co-finite

Dot-Depth Hierarchy
(Brzozowski,Cohen)'71

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

from level n, close under:

- **marked concatenation**:
  $L, K, a \mapsto L \cdot a \cdot K$
- union and intersection

Level 0

A^* and $\emptyset$

Straubing-Thérien Hierarchy
(Straubing)'81 (Thérien)'81

Classifying Star-free Languages (2)

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

from level n,

- **concatenation**:
  $L, K \mapsto L \cdot K$
- union and intersection

Level 0

finite and co-finite

Dot-Depth Hierarchy
(Brzozowski,Cohen)'71

Level $n+1$

from level $n+\frac{1}{2}$, close under:

- **complementation**,
- union and intersection

Level $n+\frac{1}{2}$

close under:

- **marked concatenation**:
  $L, K, a \mapsto L \cdot a \cdot K$
- union and intersection

Level 0

A^* and $\emptyset$

Straubing-Thérien Hierarchy
(Straubing)'81 (Thérien)'81

Both hierarchies are know to be **strict**,
(Brzozowski,Knast)'78

A natural objective

Objective

Precisely understand what can be defined with "simple" star-free descriptions:

Solve **membership for each level** in both hierarchies.

A natural objective

Objective

Precisely understand what can be defined with "simple" star-free descriptions:

Solve **membership for each level** in both hierarchies.

Before talking about this objective, let us motivate it further:

The hierarchies also admit a **natural logical definition**.

Connection with Logic: Quantifier Alternation

What is a simple FO sentence ?

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

What is a complicated sentence ?

Connection with Logic: Quantifier Alternation

What is a simple FO sentence ?

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

What is a complicated sentence ?

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

Complicated = High Quantifier Alternation

Connection with Logic: Quantifier Alternation

What is a simple FO sentence ?

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

$\Rightarrow \Pi_2$ sentence.

What is a complicated sentence ?

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

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

Complicated = High Quantifier Alternation

FO Quantifier Alternation Hierarchy

Level n: Σ_n

For all n, a Σ_n sentence is (in prenex normal form)

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

FO Quantifier Alternation Hierarchy

Level n: Σ_n

For all n, a Σ_n sentence is (in prenex normal form)

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

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

Level n: Π_n

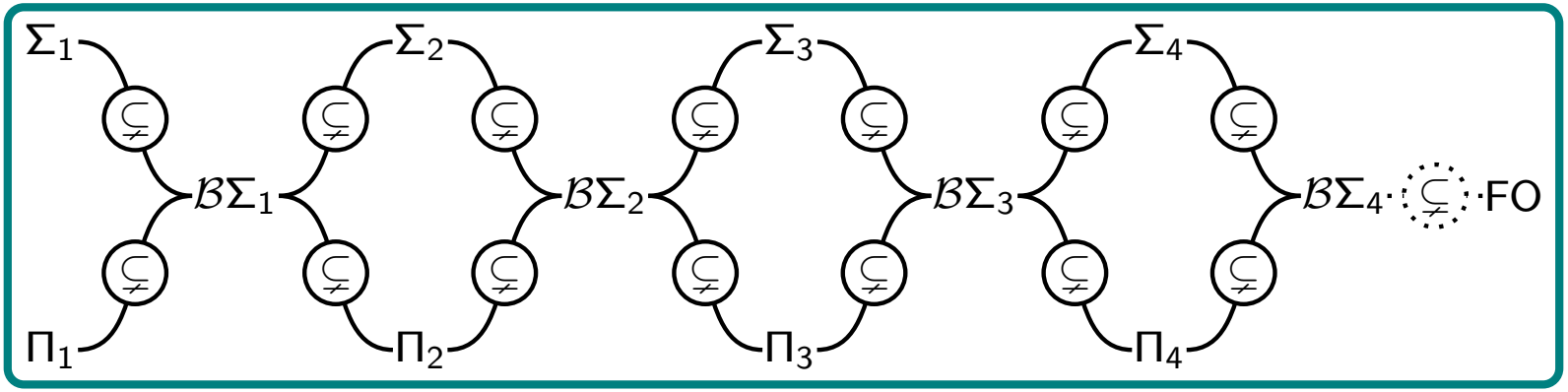
Negation of a Σ_n sentence:

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

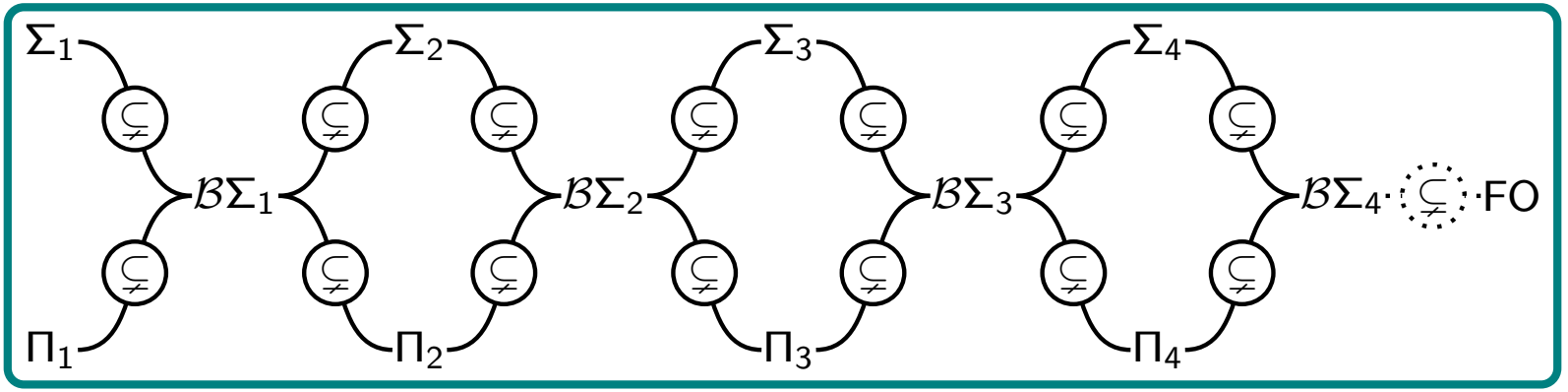
Level n: $\mathcal{B}\Sigma_n$

Boolean combinations of Σ_n (and Π_n) sentences.

FO Quantifier Alternation Hierarchy



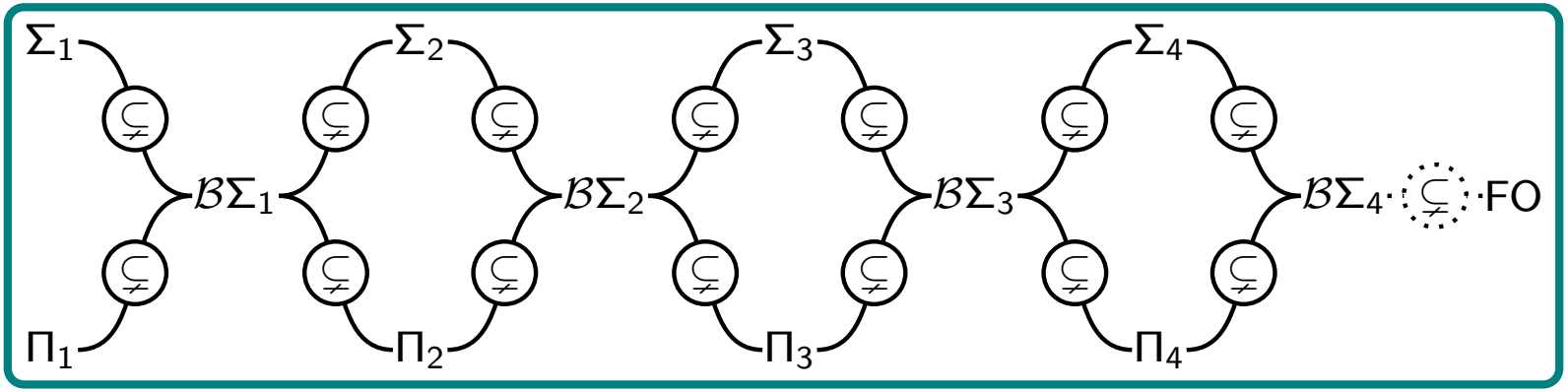
FO Quantifier Alternation Hierarchy



In the Straubing-Thérien Hierarchy (Perrin-Pin'86),

- level n is exactly $\mathcal{B}\Sigma_n$,
- level $n - \frac{1}{2}$ is exactly Σ_n

FO Quantifier Alternation Hierarchy



In the Straubing-Thérien Hierarchy (Perrin-Pin'86),

- level n is exactly $\mathcal{B}\Sigma_n$,
- level $n - \frac{1}{2}$ is exactly Σ_n

Same correspondence between dot-depth and a variant of FO with an enriched signature (Thomas'82)

Membership and the Hierarchies

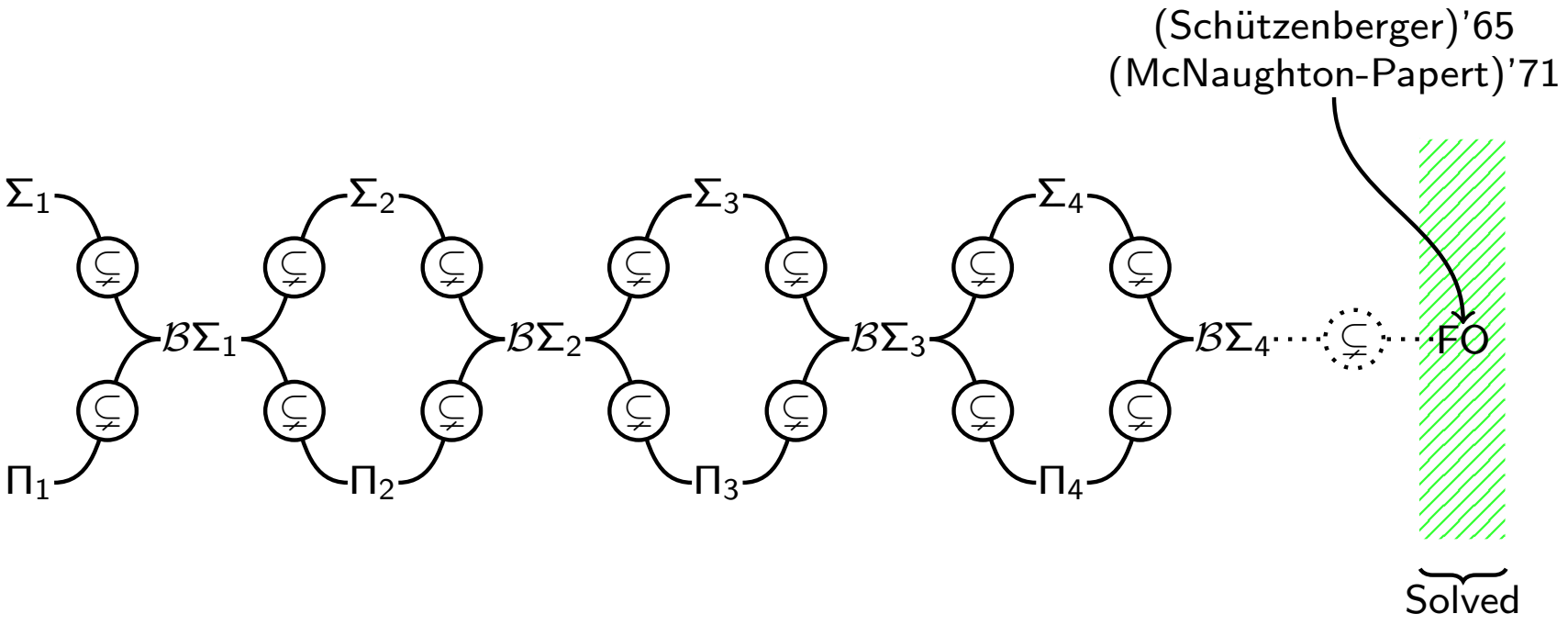
Remark

1. From now on, I will use the logical point of view (Σ_n, $\mathcal{B}\Sigma_n$).
2. Nowadays, people mostly focus on the Straubing-Thérien hierarchy. This is what I will do.

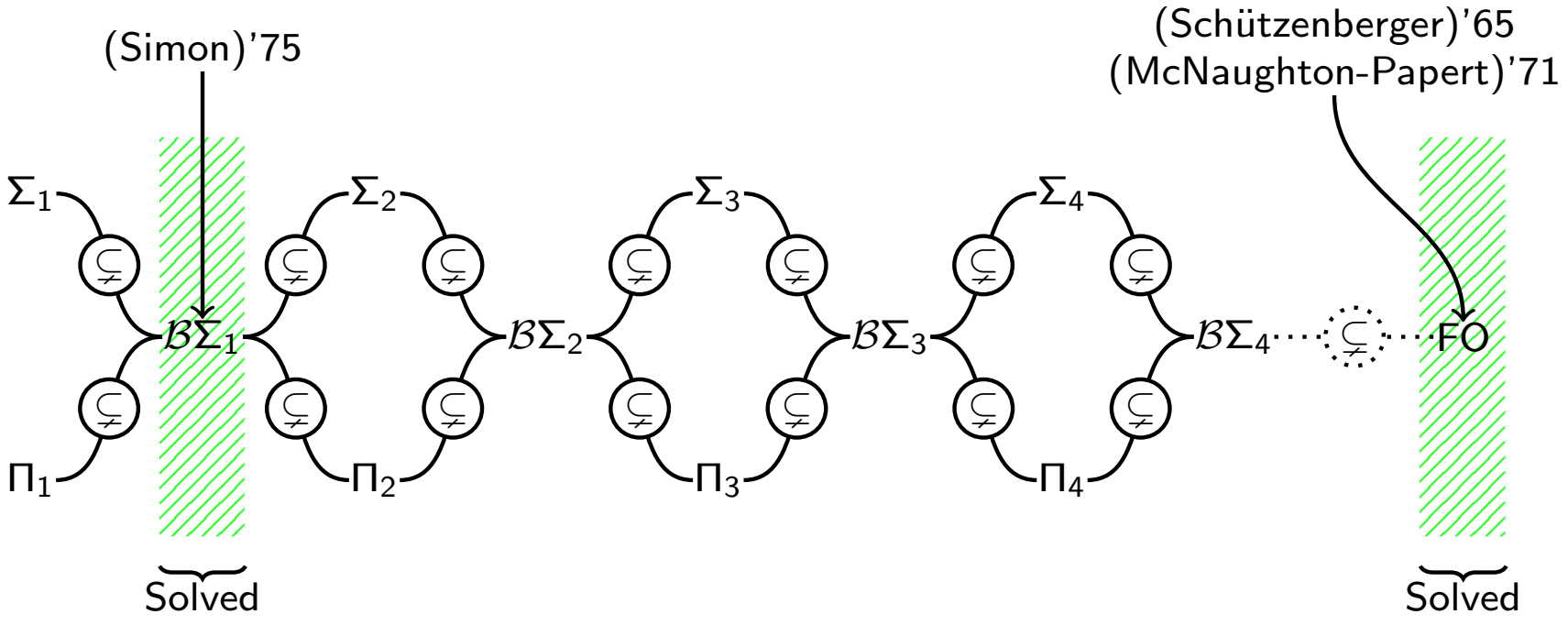
(Straubing'85) For each level:

membership for the dot-depth hierarchy
can be **effectively reduced**
to membership for the Straubing-Thérien Hierarchy

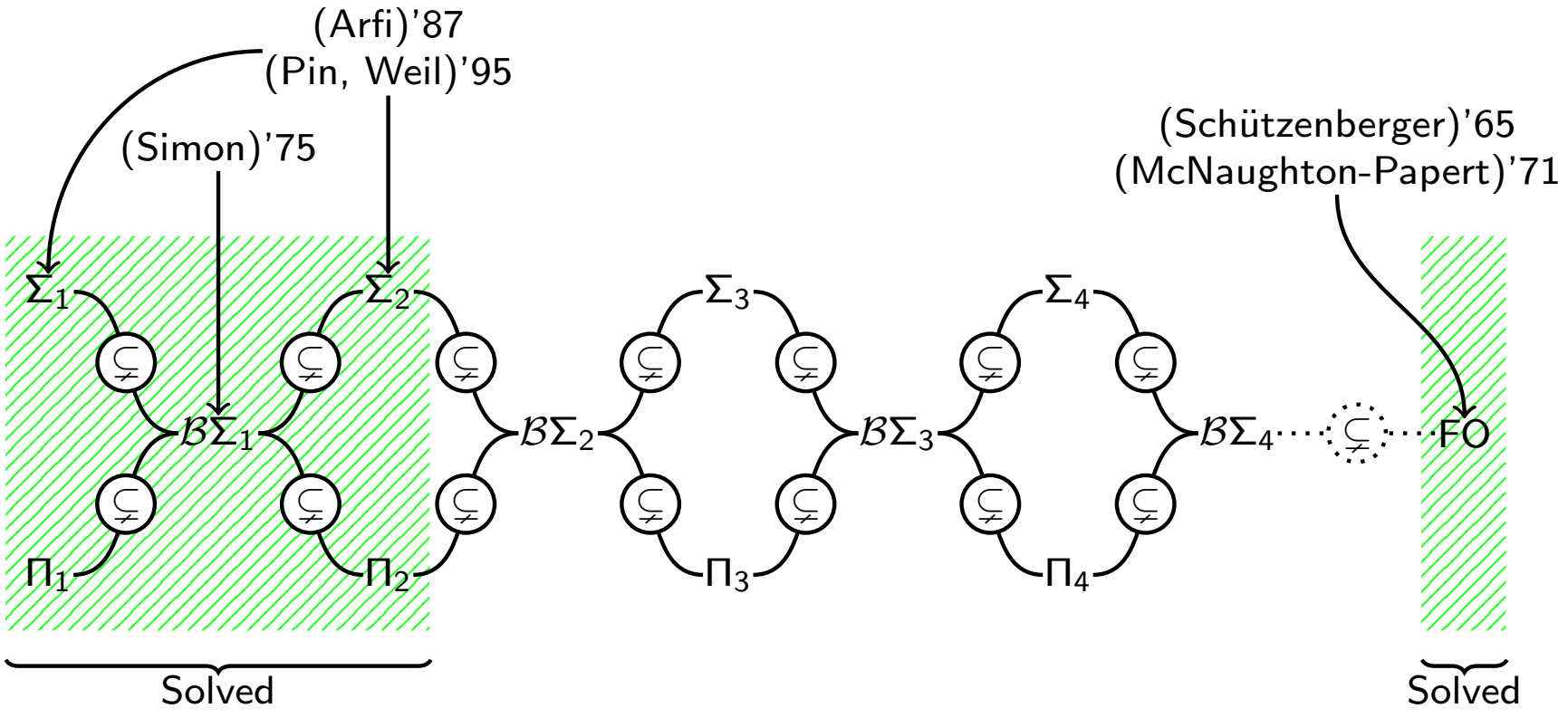
FO Quantifier Alternation: Membership State of the Art



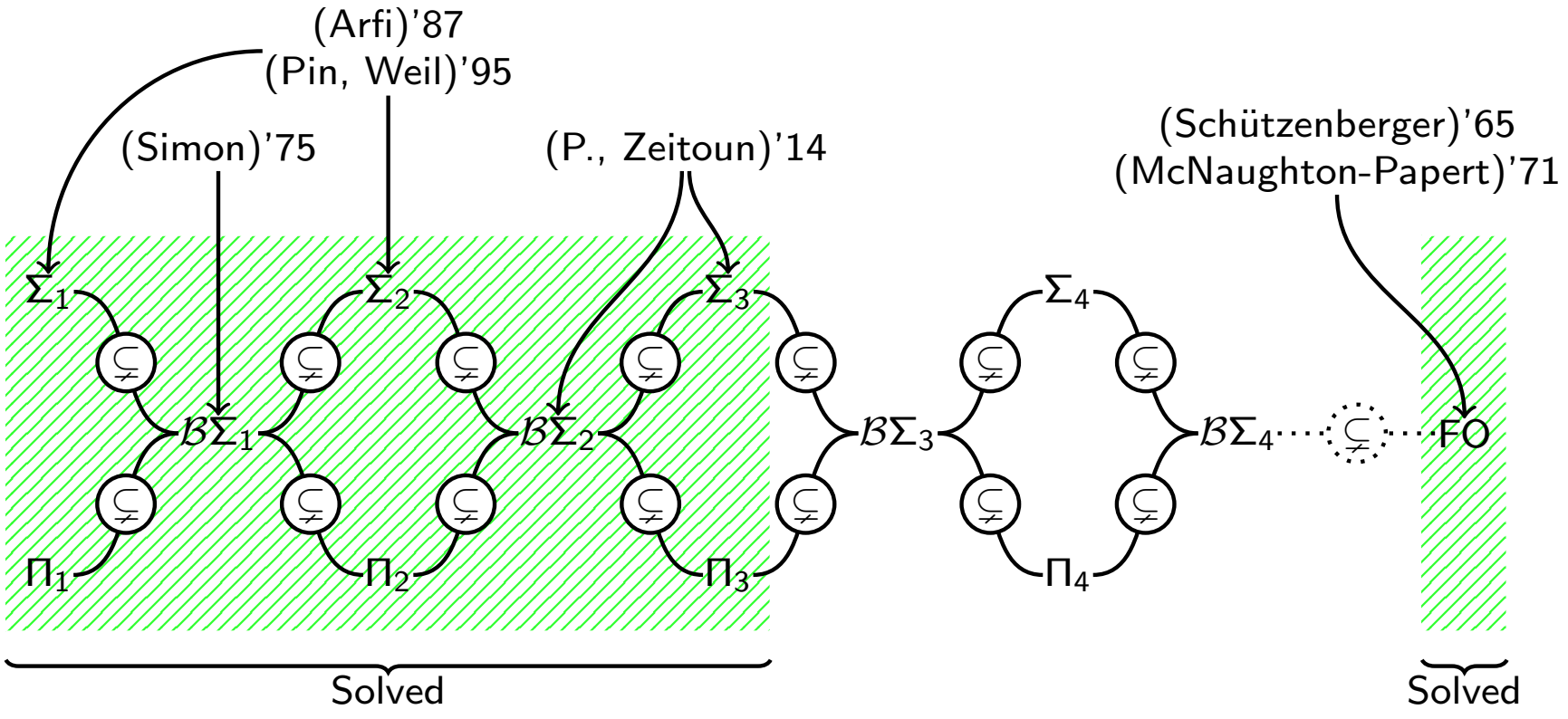
FO Quantifier Alternation: Membership State of the Art



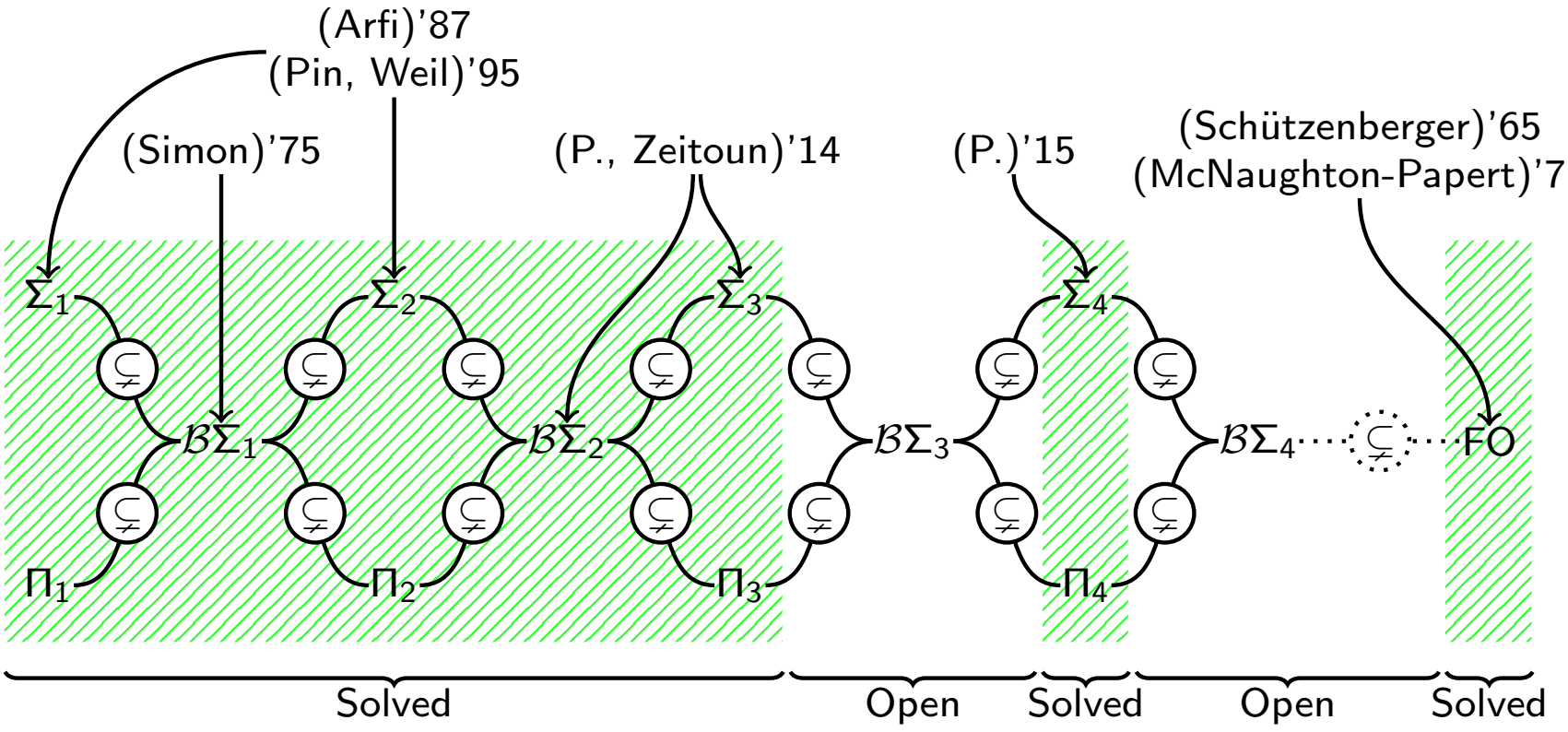
FO Quantifier Alternation: Membership State of the Art



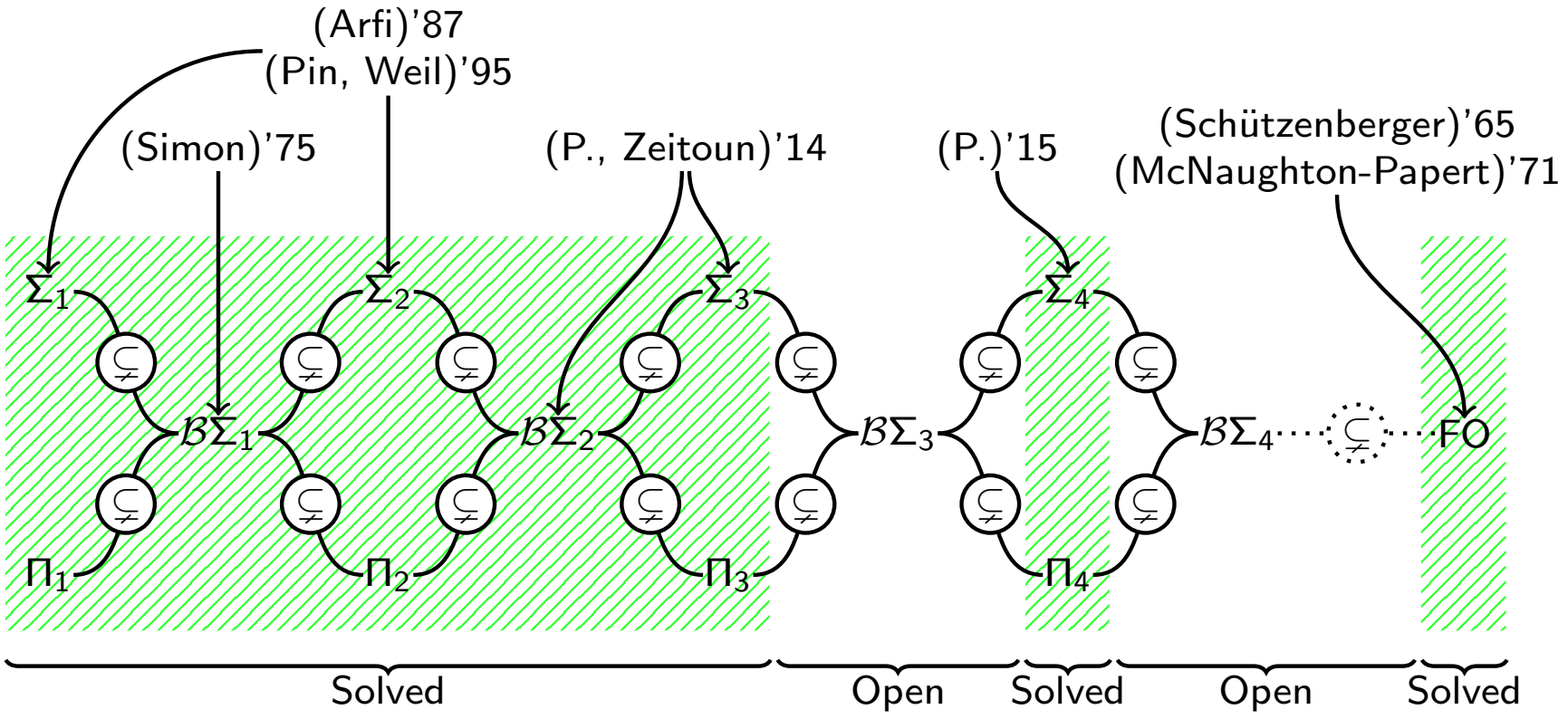
FO Quantifier Alternation: Membership State of the Art



FO Quantifier Alternation: Membership State of the Art



FO Quantifier Alternation: Membership State of the Art



Why is progress on this question so slow ?

Why are we still stuck ?

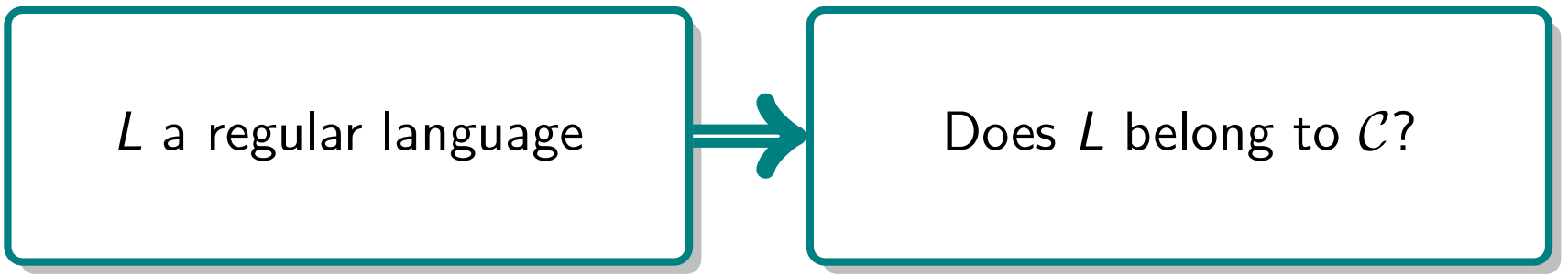
A deeper look at membership questions

Quick Remark

- The original approach uses an algebraic point of view: the central object is the syntactic monoid.
- I will use an automata point of view: the central object is the minimal automaton.

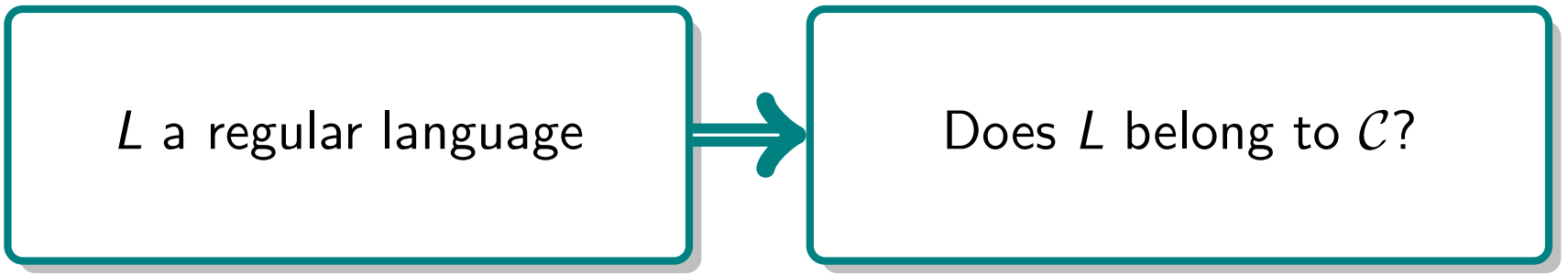
Membership Problem: General approach

For a level $\mathcal{C}$ in the hierarchy, we search for a membership algorithm for $\mathcal{C}$ which decides the following problem.



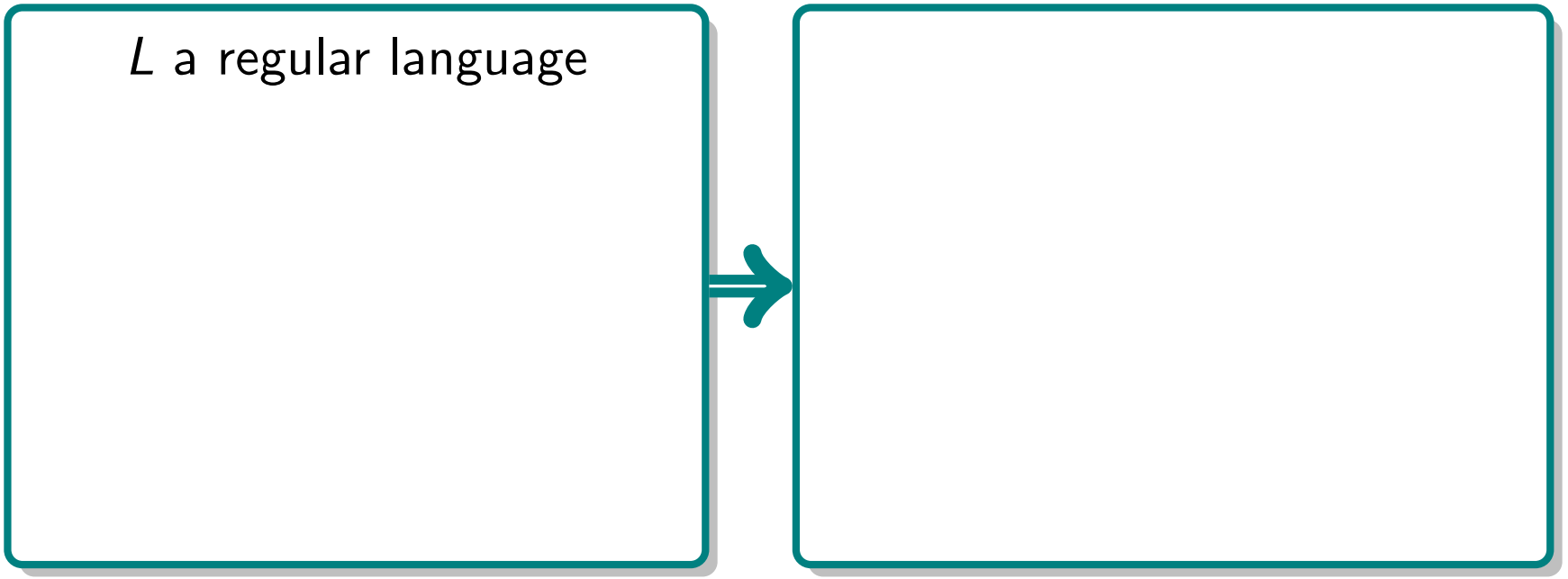
Membership Problem: General approach

For a level $\mathcal{C}$ in the hierarchy, we search for a membership algorithm for $\mathcal{C}$ which decides the following problem.



This is not how the question is approached.

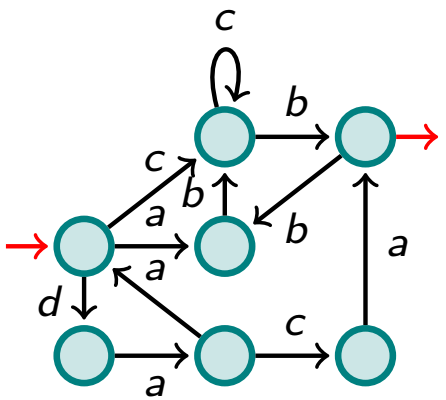
The problem we are really considering



The problem we are really considering

L a regular language

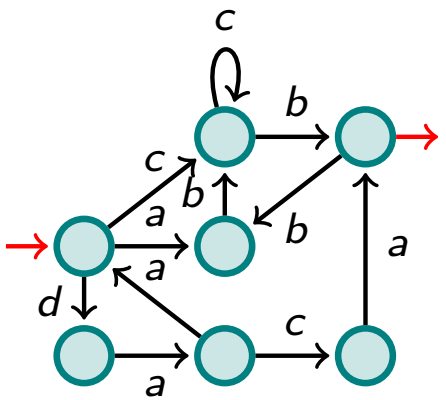
It has a **minimal automaton**



The problem we are really considering

L a regular language

It has a **minimal automaton**



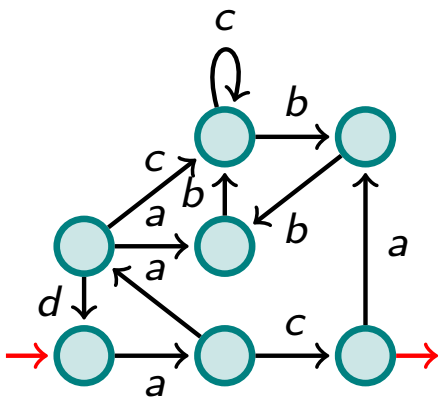
$\Rightarrow$ **set of accepted languages**

The initial and final states can be changed

The problem we are really considering

L a regular language

It has a **minimal automaton**



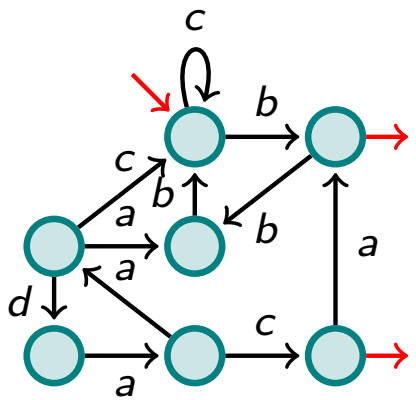
The initial and final states can be changed

$\Rightarrow$ **set of accepted languages**

The problem we are really considering

L a regular language

It has a **minimal automaton**



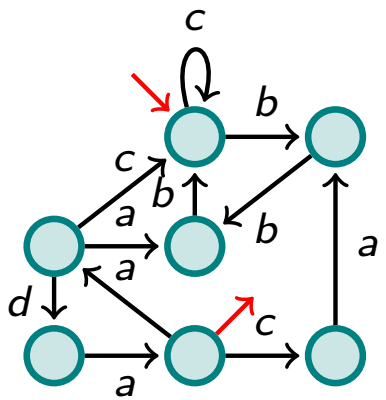
The initial and final states can be changed

$\Rightarrow$ **set of accepted languages**

The problem we are really considering

L a regular language

It has a **minimal automaton**



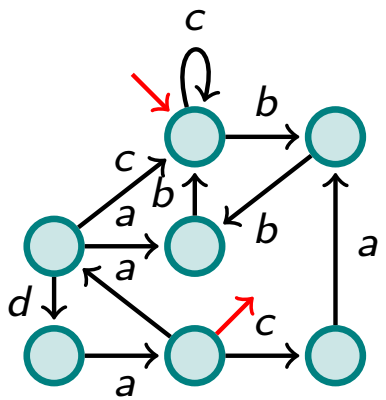
The initial and final states can be changed

$\Rightarrow$ **set of accepted languages**

The problem we are really considering

L a regular language

It has a **minimal automaton**



$\Rightarrow$ **set of accepted languages**

L in $\mathcal{C}$

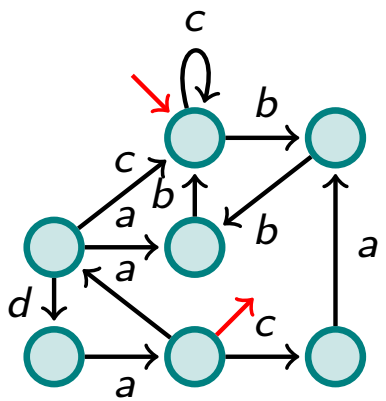
$\Updownarrow$

All accepted languages in $\mathcal{C}$

The problem we are really considering

L a regular language

It has a **minimal automaton**



The initial and final states can be changed

$\Rightarrow$ **set of accepted languages**

L in $\mathcal{C}$

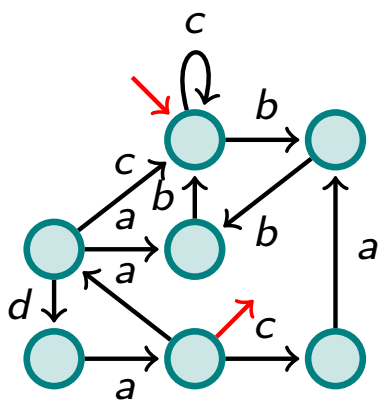
All accepted languages in $\mathcal{C}$

- The set is finite and has a **structure**.
- This structure is connected to our building operations: boolean operations and **concatenation**.
- L is built-up from the languages in the set.

The problem we are really considering

L a regular language

It has a **minimal automaton**



The initial and final states can be changed

$\Rightarrow$ **set of accepted languages**

L in $\mathcal{C}$

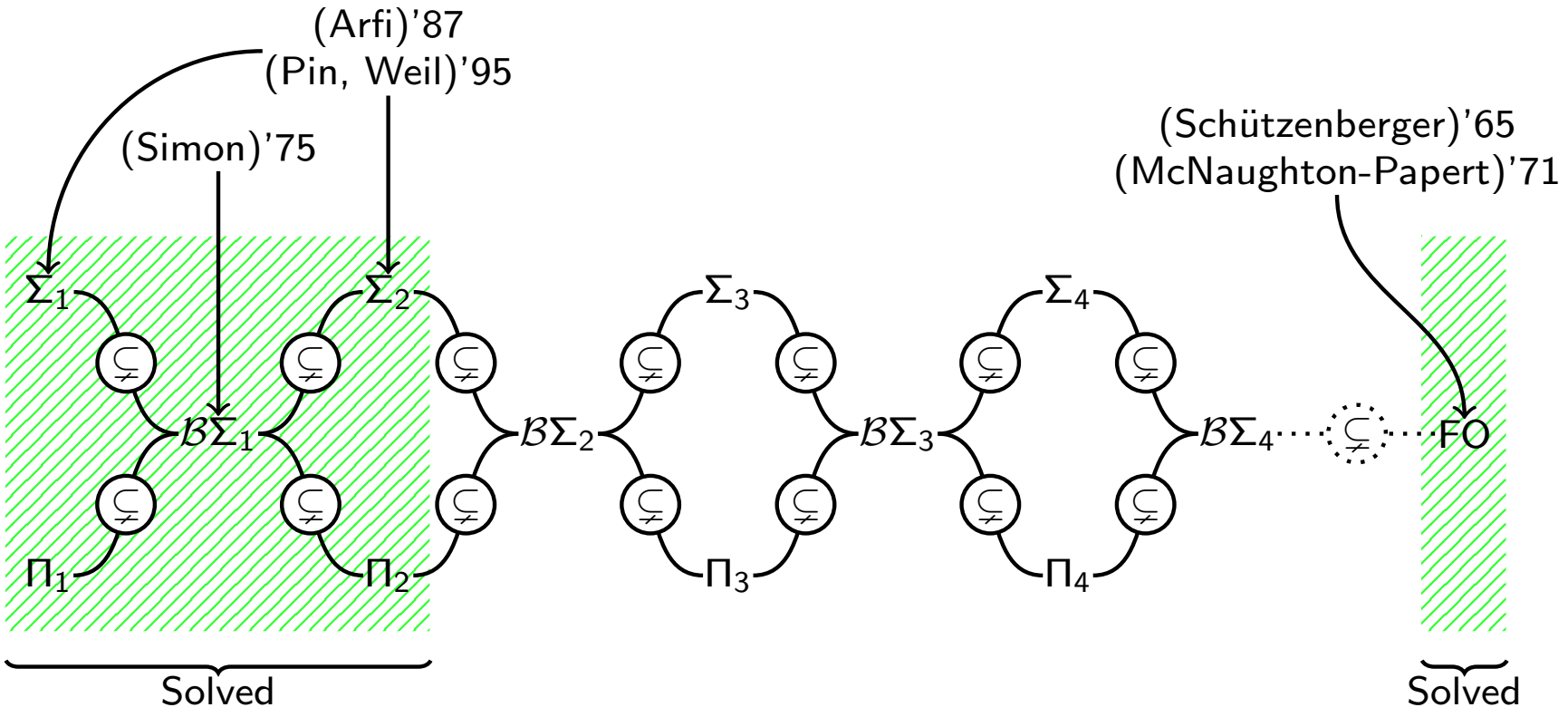
$\Updownarrow$

All accepted languages in $\mathcal{C}$

- The set is finite and has a **structure**.
- This structure is connected to our building operations: boolean operations and **concatenation**.
- L is built-up from the languages in the set.

Algorithms (and their proofs) are **based on this structure**.

With this approach alone, we get the following results



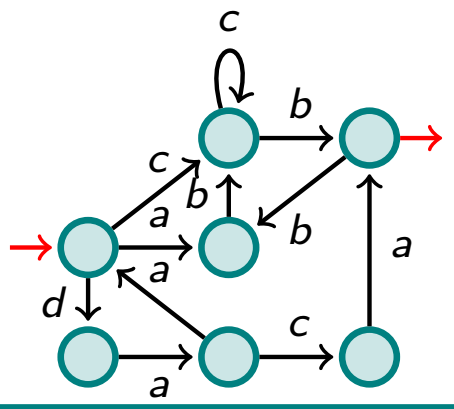
What is the missing ingredient to go higher ?

Issues with the general approach

Reminder: the problem we are really considering

L a regular language

It has a **minimal automaton**



We ask whether **all** accepted languages belong to $\mathcal{C}$

L in $\mathcal{C}$

$\Updownarrow$

All accepted languages in $\mathcal{C}$

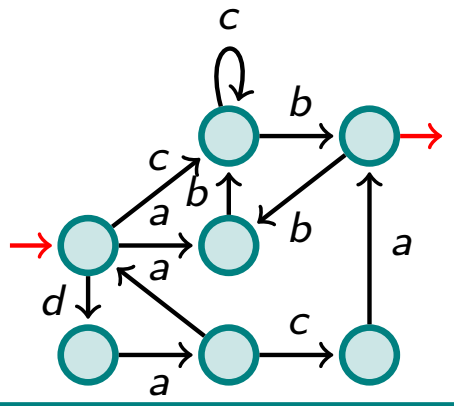
Benefits

Algorithms (and their proofs) are **based on the structure of the set of accepted languages**.

Reminder: the problem we are really considering

L a regular language

It has a **minimal automaton**



We ask whether **all** accepted languages belong to $\mathcal{C}$

L in $\mathcal{C}$ ⇕ All accepted languages in $\mathcal{C}$

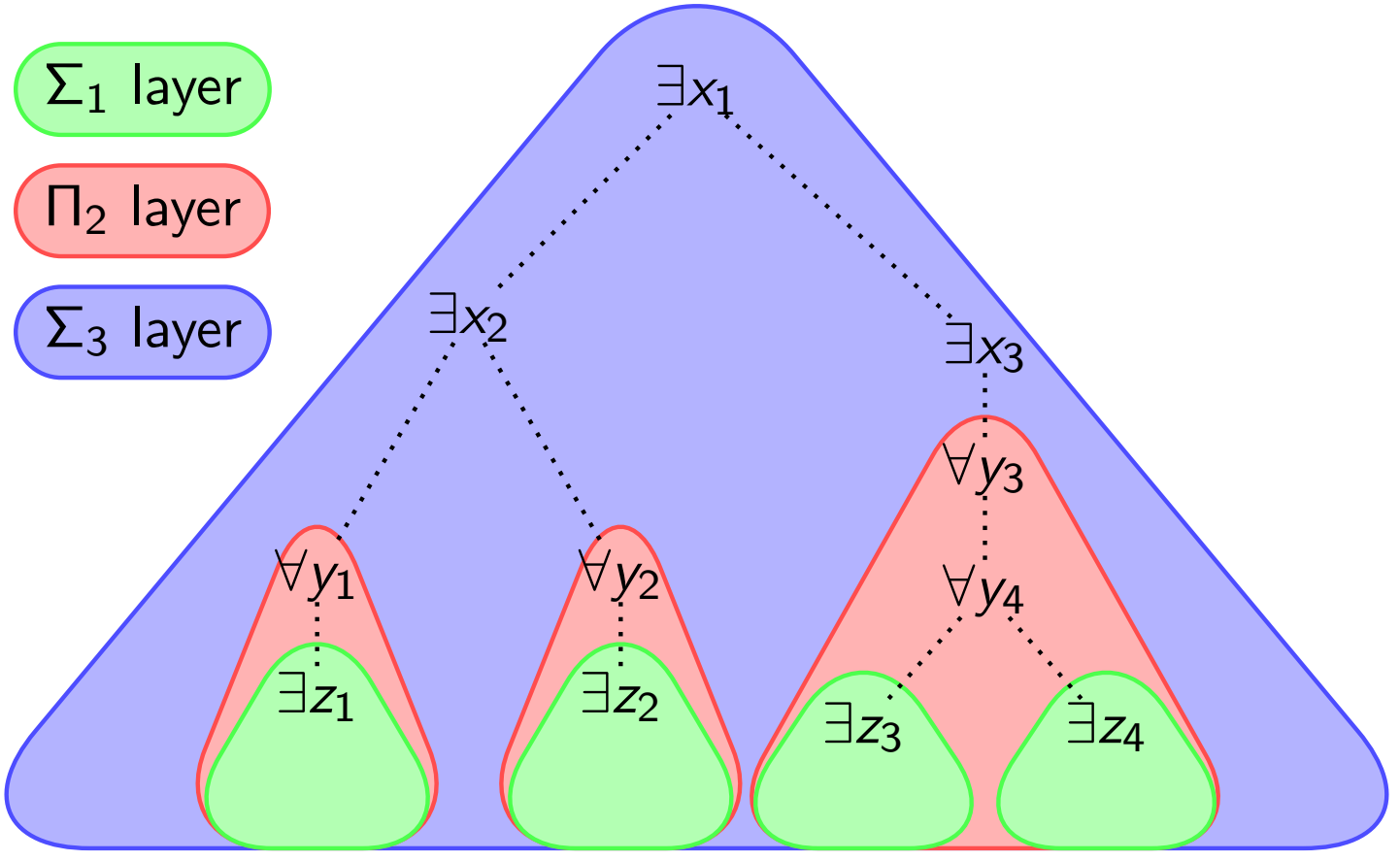
Benefits

Algorithms (and their proofs) are **based on the structure of the set of accepted languages**.

In particular, this is **essential for the inductive construction** of sentences.

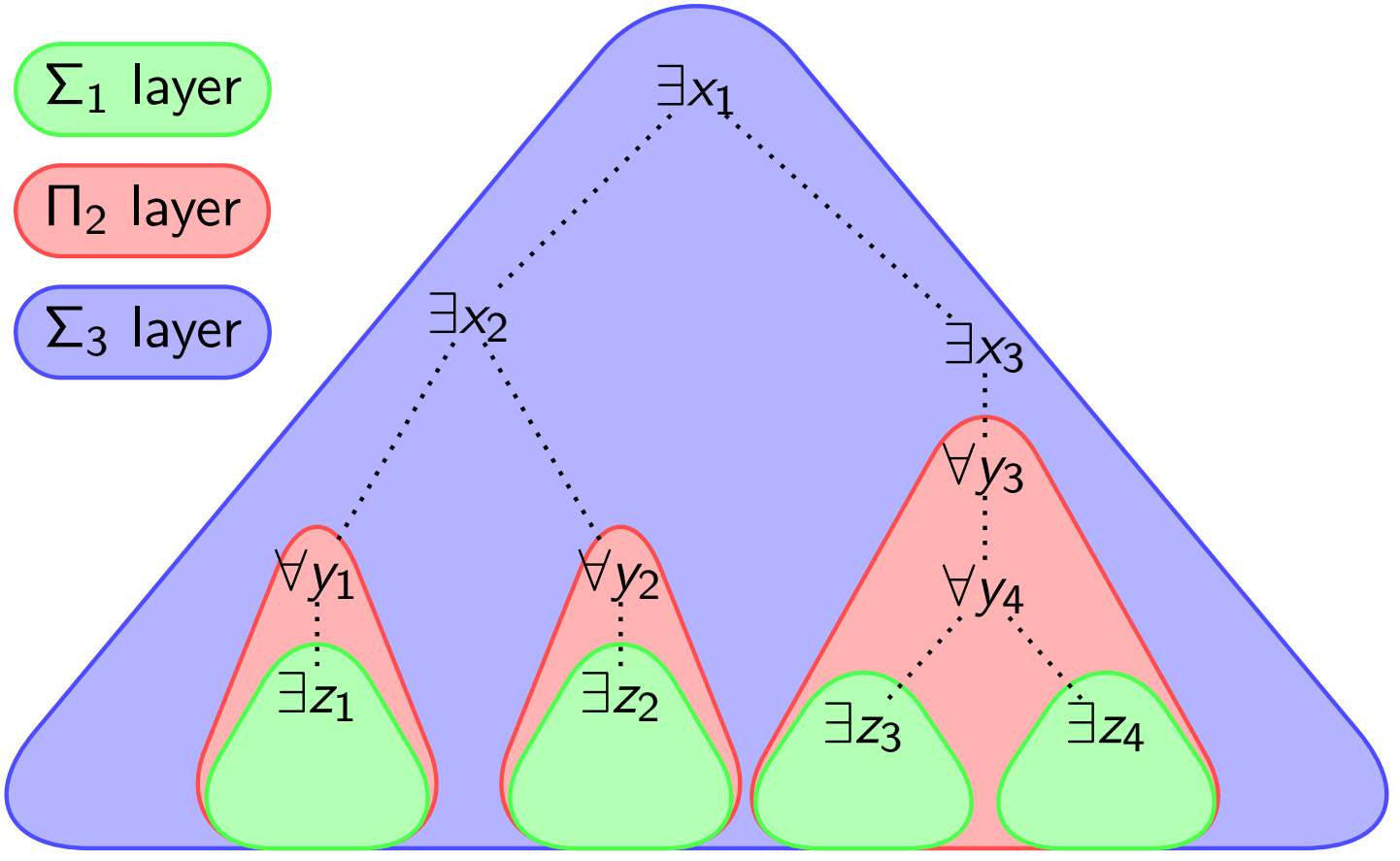
Why is this a problem for the Hierarchy ?

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



Why is this a problem for the Hierarchy ?

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



Reminder: For $\mathcal{C}$-membership the hard part is getting a generic way to build $\mathcal{C}$-sentences (when possible)

Why is this a problem for the Hierarchy ?

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

- Building Σ_3 sentences by induction for **all** accepted languages is also a layered construction.
- This requires to first build Π_2 sentences.

Reminder: For $\mathcal{C}$-membership the hard part is getting a generic way to build $\mathcal{C}$-sentences (when possible)

Why is this a problem for the Hierarchy ?

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

- Building Σ_3 sentences by induction for **all** accepted languages is also a layered construction.
- This requires to first build Π_2 sentences.

$\Rightarrow$ We first need to ask a "Π_2 **question**" on our input automaton.

- In general this "Π_2 question" **cannot be**: "are all accepted languages in Π_2?"

Reminder: For $\mathcal{C}$-membership the hard part is getting a generic way to build $\mathcal{C}$-sentences (when possible)

Why is this a problem for the Hierarchy ?

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

- Building Σ_3 sentences by induction for **all** accepted languages is also a layered construction.
- This requires to first build Π_2 sentences
- $\Rightarrow$ The "Π_2 question" must involve **a more general problem than membership**
- = on our input automaton.
- In general this "Π_2 question" **cannot be**: "are all accepted languages in Π_2?"

Reminder: For $\mathcal{C}$-membership the hard part is getting a generic way to build $\mathcal{C}$-sentences (when possible)

The main idea behind membership for $\mathcal{B}\Sigma_2$, Σ_3

Two steps:

1. solve a deeper problem for Σ_2 independently.
2. reuse the answer to this problem in the membership algorithm for $\mathcal{B}\Sigma_2$ or Σ_3.

What is a "more general problem than membership"?

What we want is an "**approximation problem**":

to build Σ_3 sentences, we first want compute the "best possible Π_2-approximation of our languages".

What is a "more general problem than membership"?

What we want is an "**approximation problem**":

to build Σ_3 sentences, we first want compute the "best possible Π_2-approximation of our languages".

We have several "approximation problems" (each one tailored to a particular logic). However they are all based on a common one:

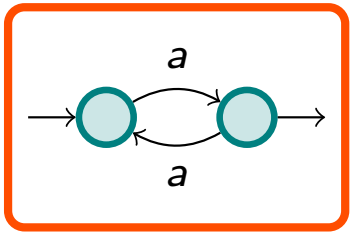
the **separation problem**

More General: Separation is Here

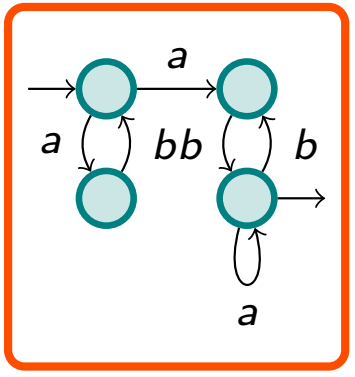
Given a level $\mathcal{C}$ in the hierarchy, decide the following problem:

L_1, L_2 two regular languages

L_1



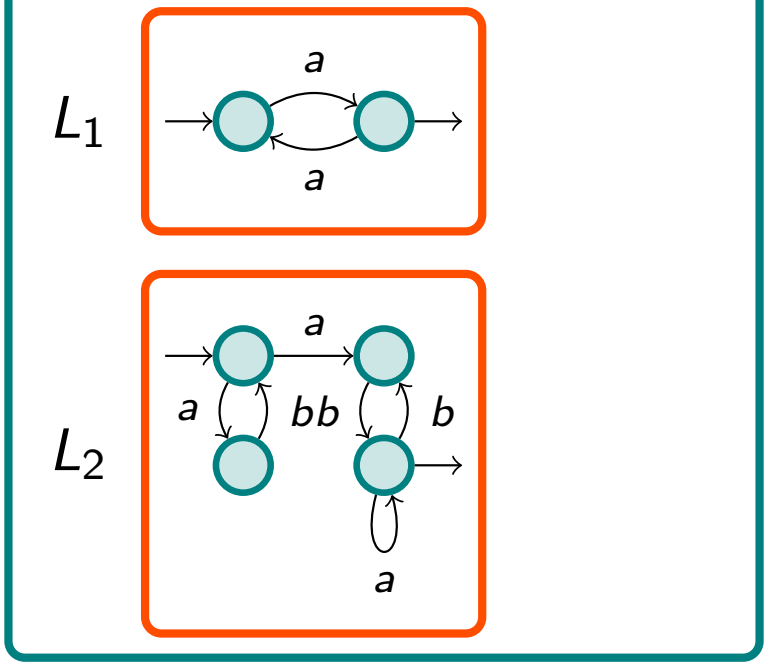
L_2



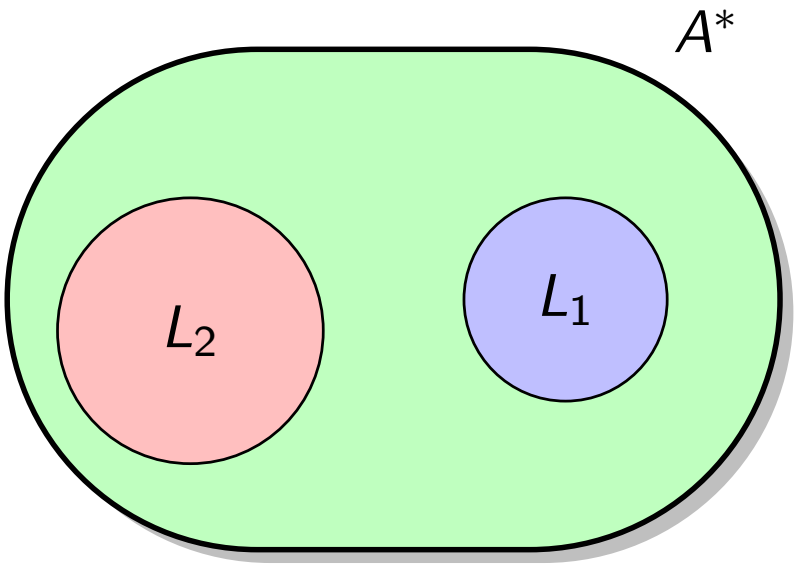
More General: Separation is Here

Given a level $\mathcal{C}$ in the hierarchy, decide the following problem:

L_1, L_2 two regular languages



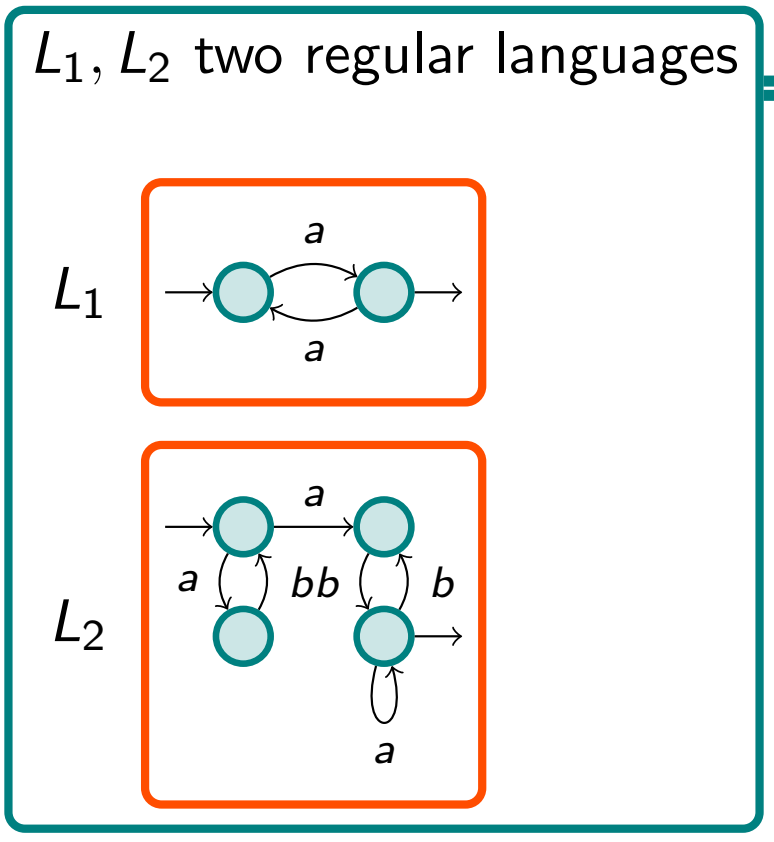
Can L_1 be separated from L_2 with a sentence of $\mathcal{C}$?



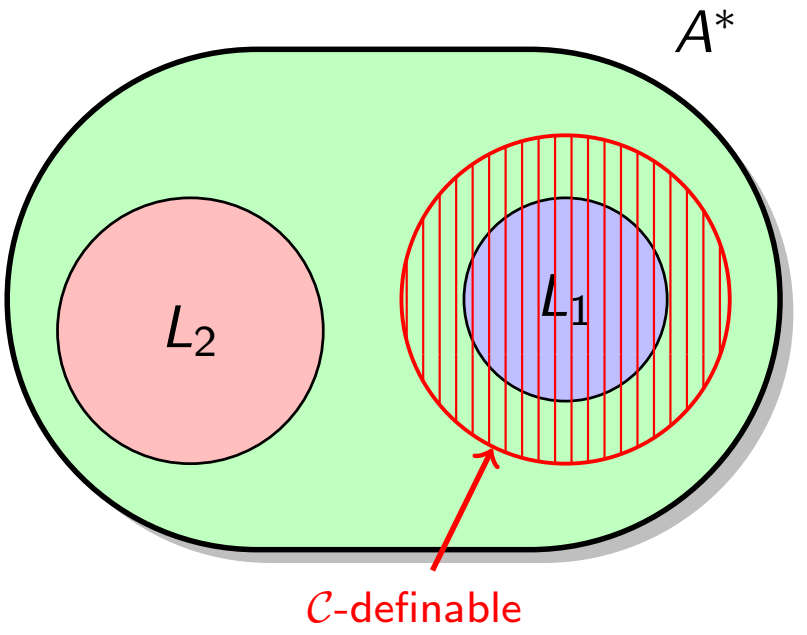
More General: Separation is Here

Given a level $\mathcal{C}$ in the hierarchy, decide the following problem:

L_1, L_2 two regular languages



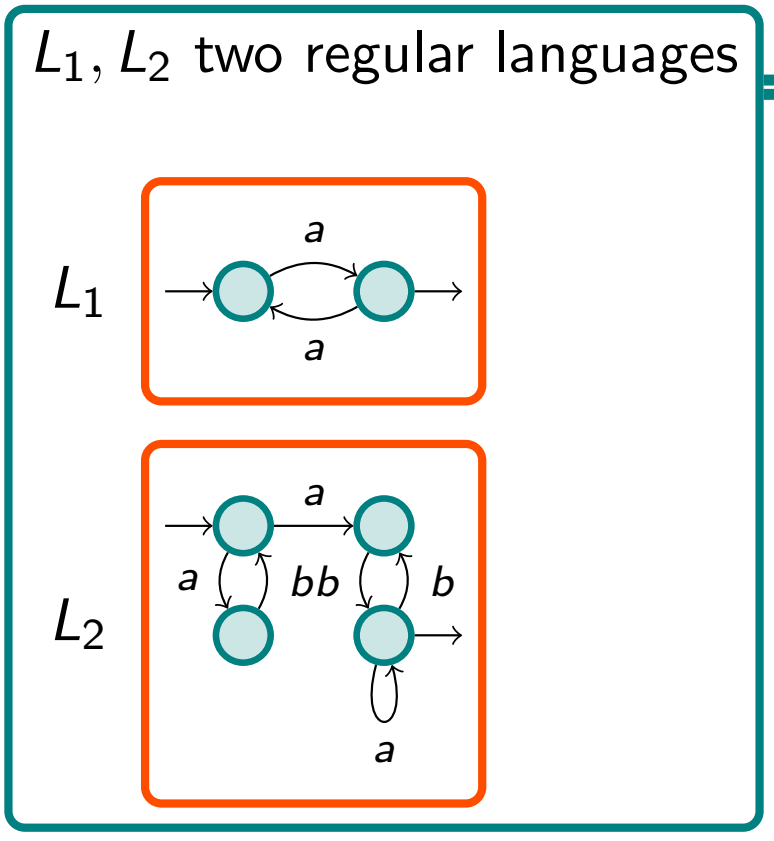
Can L_1 be separated from L_2 with a sentence of $\mathcal{C}$?



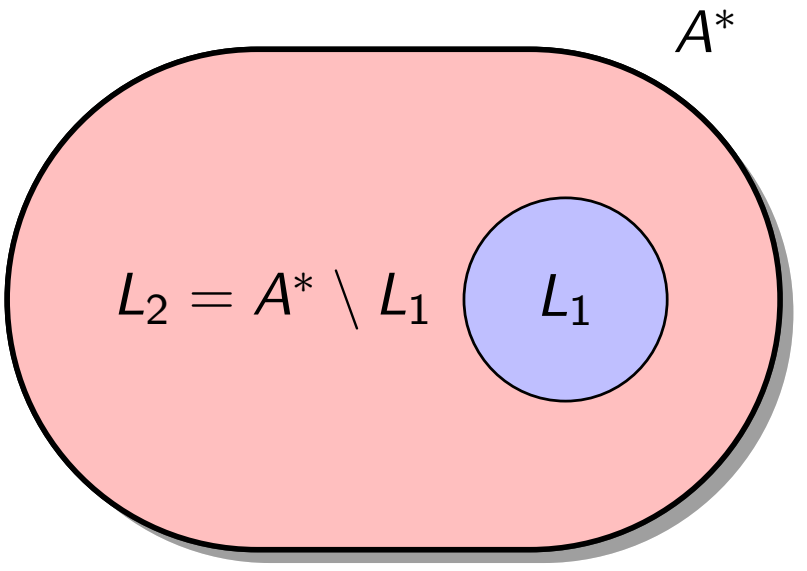
More General: Separation is Here

Given a level $\mathcal{C}$ in the hierarchy, decide the following problem:

L_1, L_2 two regular languages



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

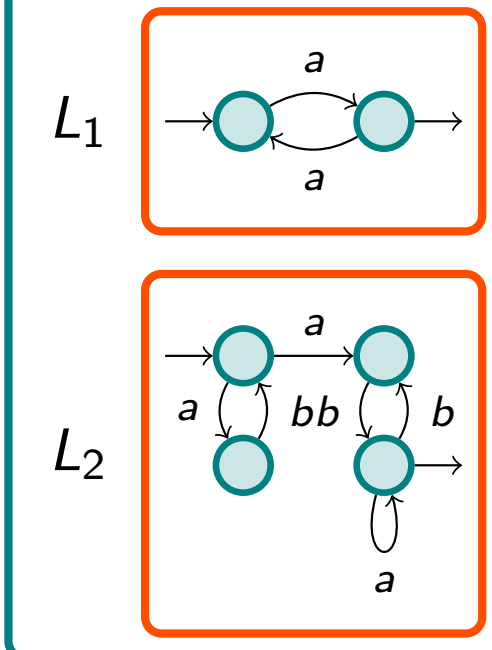


Membership can be formally reduced to separation

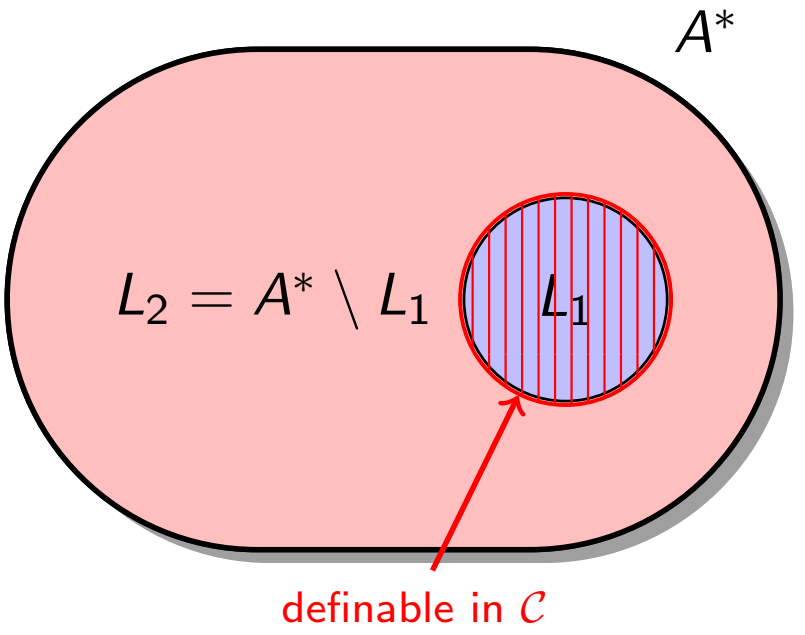
More General: Separation is Here

Given a level $\mathcal{C}$ in the hierarchy, decide the following problem:

L_1, L_2 two regular languages

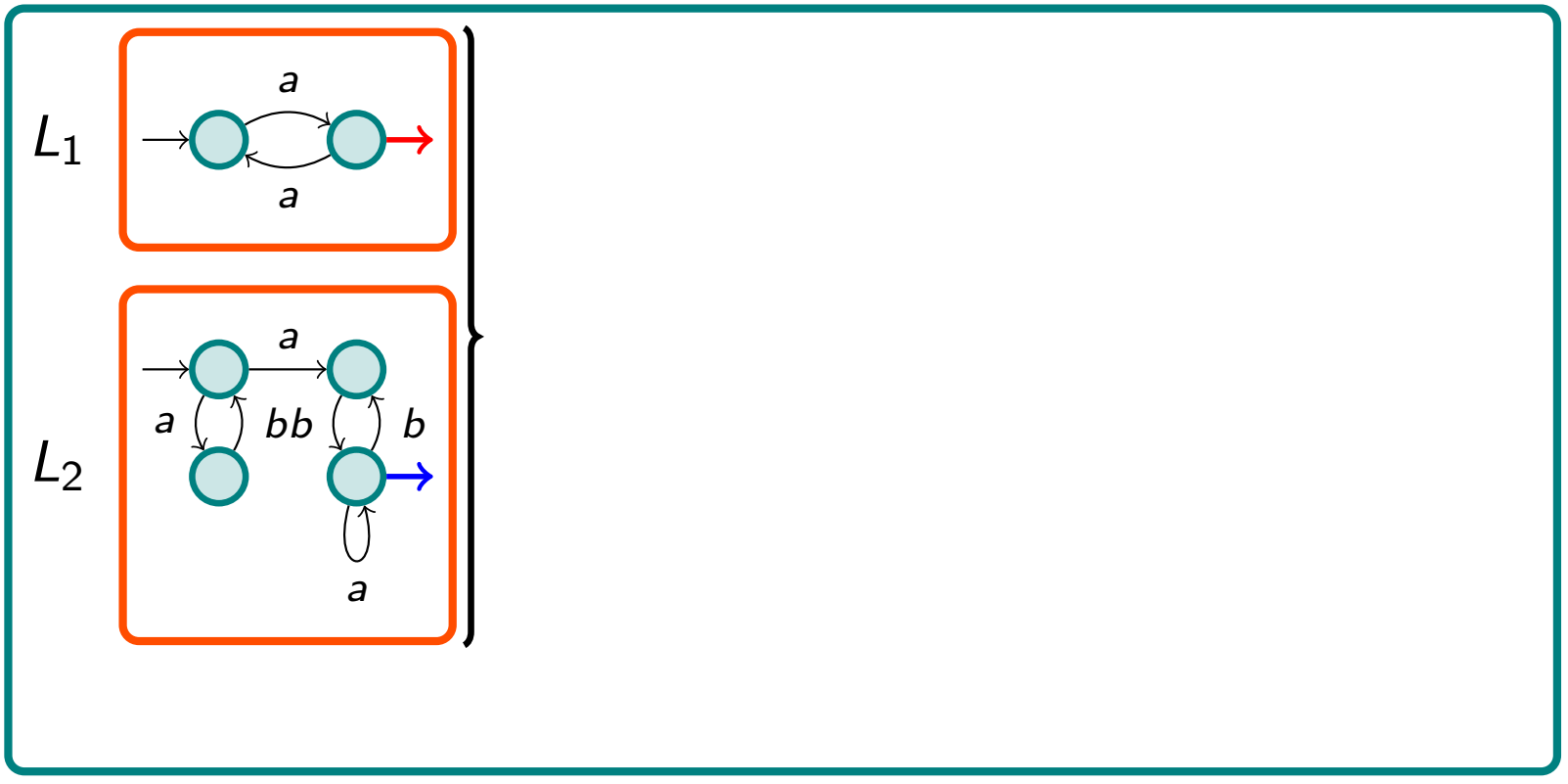


Can L_1 be separated from L_2 with a sentence of $\mathcal{C}$?

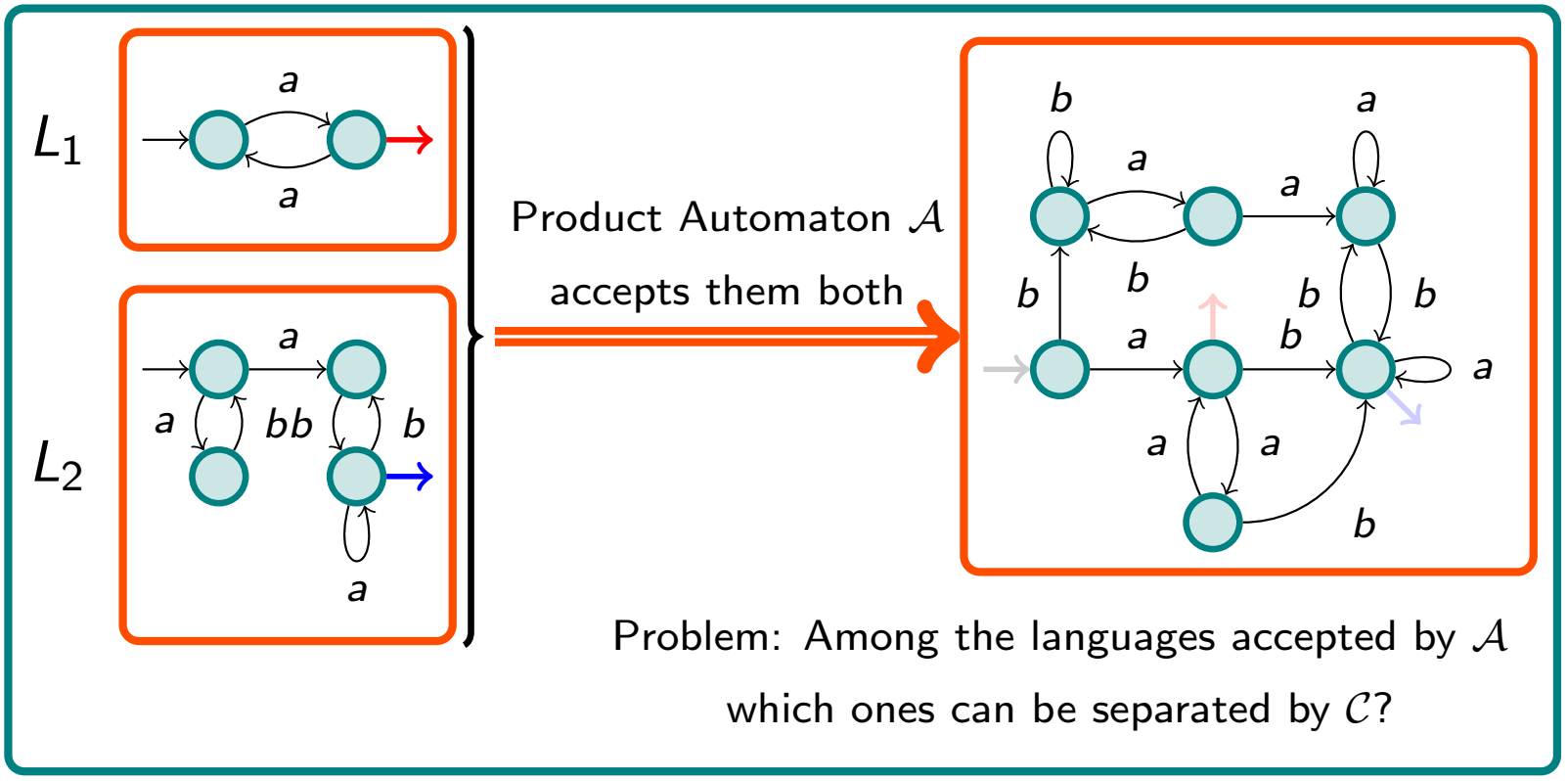


Membership can be formally reduced to separation

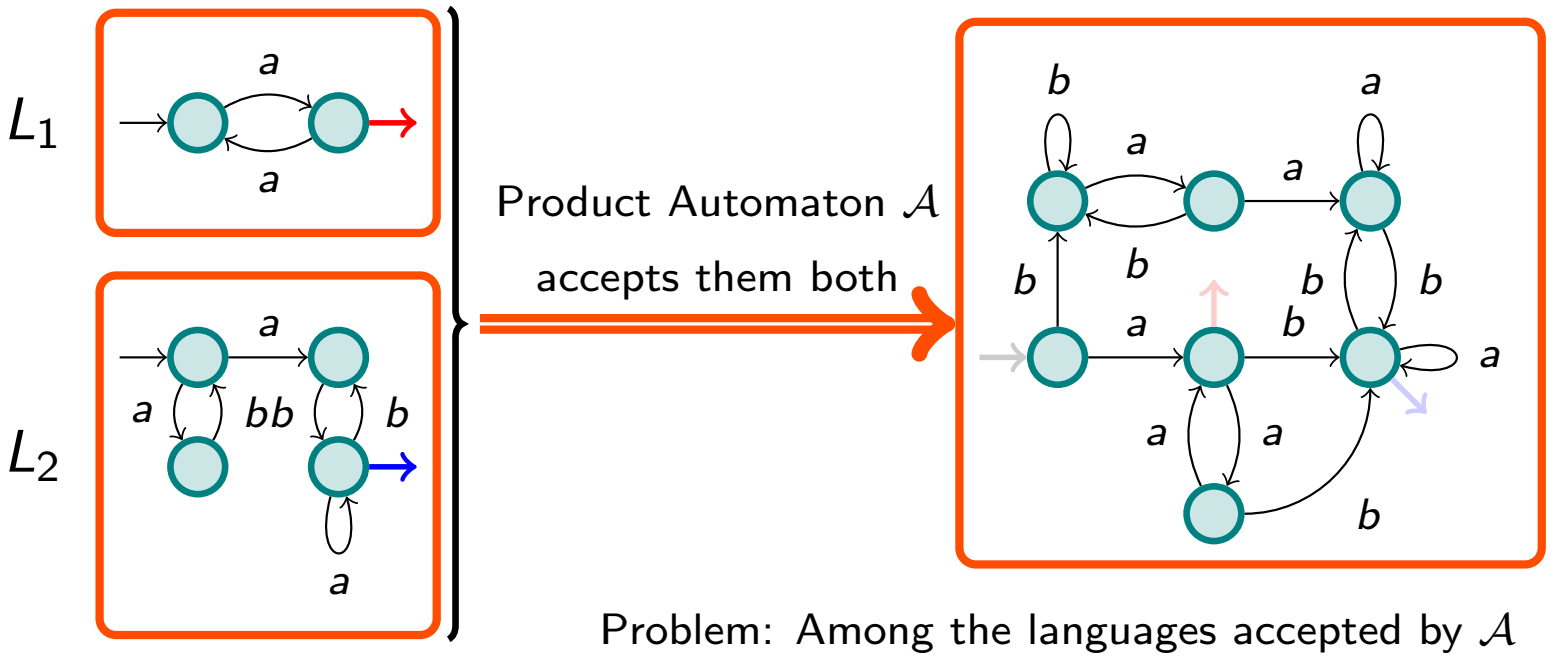
A more General Problem than Membership



A more General Problem than Membership



A more General Problem than Membership



Problem: Among the languages accepted by $\mathcal{A}$ which ones can be separated by $\mathcal{C}$?

Membership

1) Can we define **everything** about the input with $\mathcal{C}$?

2) If yes, compute sentences.

Separation

1) How well can the languages be **approximated** with $\mathcal{C}$?

2) Compute sentences realizing this best approximation.

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

$$\textit{membership for } \Sigma_{n+1} \textbf{ reduces to } \textit{separation for } \Sigma_n$$

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

membership for Σ_{n+1} **reduces to** *separation for* Σ_n

Using this theorem, an algorithm for Σ_{n+1} membership works in three steps,

1. compute the syntactic monoid of the input L.

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

membership for Σ_{n+1} **reduces to** *separation for* Σ_n

Using this theorem, an algorithm for Σ_{n+1} membership works in three steps,

1. compute the syntactic monoid of the input L.
2. among the pairs of languages recognized by this syntactic monoid, compute which ones are Σ_n-separable.
   (this requires an **independent** Σ_n-separation algorithm).

The proof for Σ_{n+1}-membership is constructive provided that the independent Σ_n-separation algorithm is constructive as well.

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

membership for Σ_{n+1} **reduces to** *separation for* Σ_n

Using this theorem, an algorithm for Σ_{n+1} membership works in three steps,

1. compute the syntactic monoid of the input L.
2. among the pairs of languages recognized by this syntactic monoid, compute which ones are Σ_n-separable.
   (this requires an **independent** Σ_n-separation algorithm).
3. Σ_{n+1} is now equivalent to a syntactic criterion on the syntactic which depends on the above pairs.

The proof for Σ_{n+1}-membership is constructive provided that the independent Σ_n-separation algorithm is constructive as well.

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

$$\textit{membership for } \Sigma_{n+1} \textbf{ reduces to } \textit{separation for } \Sigma_n$$

Our Σ_3-membership algorithm is based on an (independent) Σ_2-separation algorithm.

The example of Σ_3

Theorem ((P.,Zeitoun)'14)

For all $n \geq 1$,

$$\textit{membership for } \Sigma_{n+1} \textbf{ reduces to } \textit{separation for } \Sigma_n$$

Our Σ_3-membership algorithm is based on an (independent) Σ_2-separation algorithm.

Important Remark

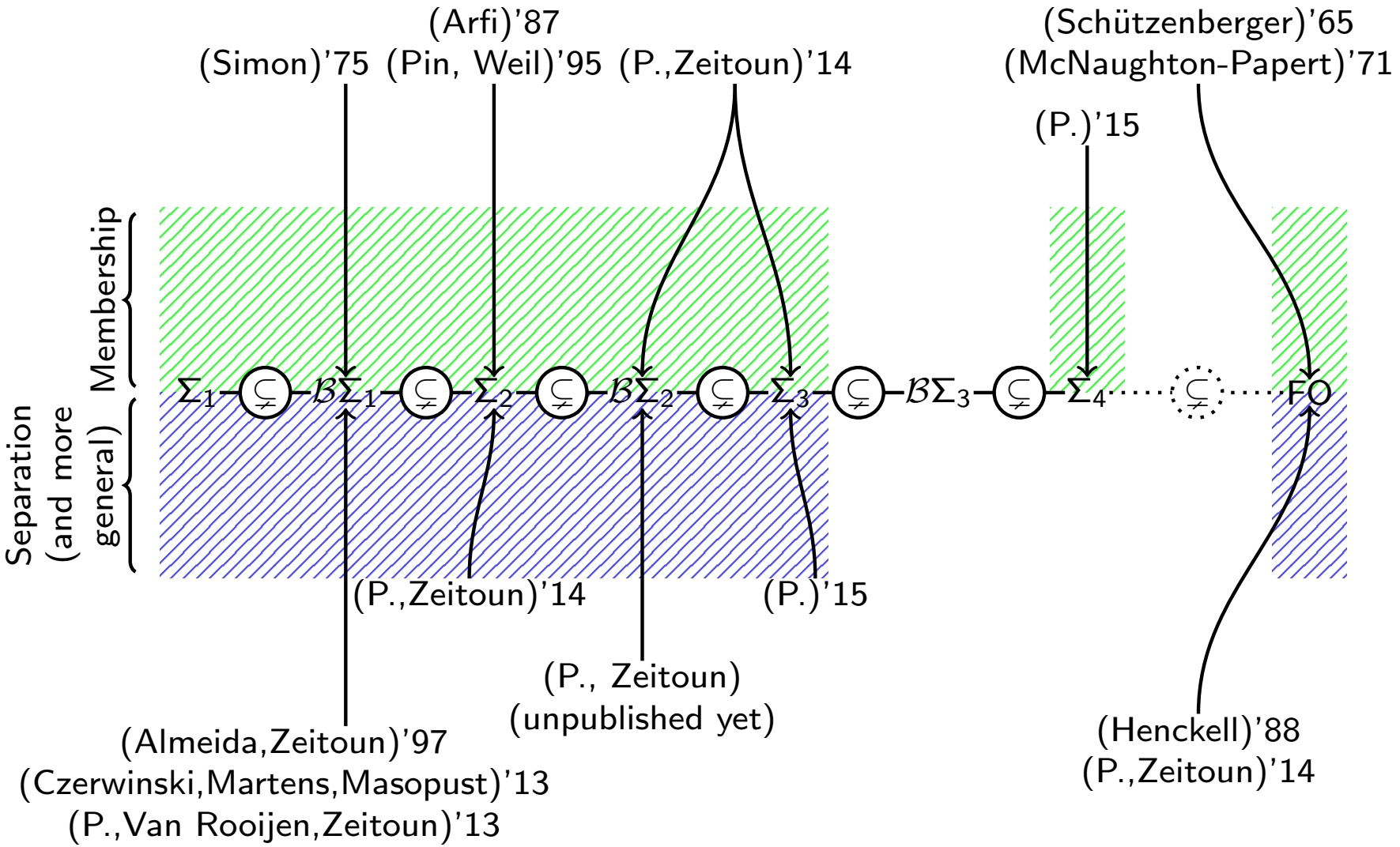
Separation is **harder** than membership. The theorem above does **not** solve the whole hierarchy.

The example of $\mathcal{B}\Sigma_2$

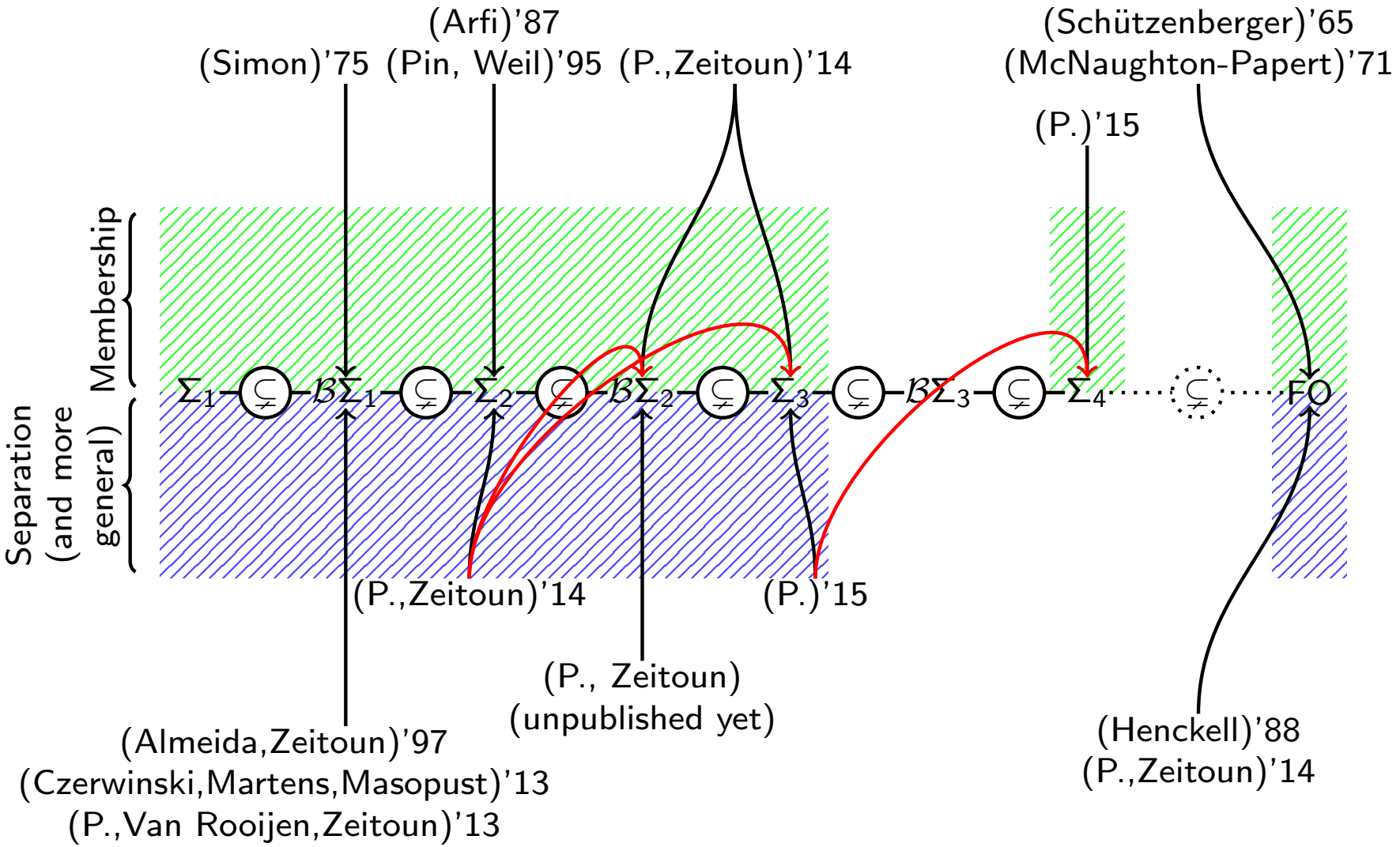
A similar theorem holds between Σ_2 and $\mathcal{B}\Sigma_2$. However,

1. the theorem is specific to level $n = 2$.
2. the connection is based on a problem **even more general** than separation (which we call "pointed covering").

Current state of the art: Approximation Problems



Current state of the art: Approximation Problems



Conclusion

Conclusion

What we learned:

We have to consider **more general problems** than membership.

Conclusion

What we learned:

We have to consider **more general problems** than membership.

Current and future work:

- the family of "approximation problems" is a jungle.
- our original separation algorithms are non-constructive.

We think, we have found the solution to these issues with a new problem called "the **covering problem**".

Conclusion

What we learned:

We have to consider **more general problems** than membership.

Current and future work:

- the family of "approximation problems" is a jungle.
- our original separation algorithms are non-constructive.

We think, we have found the solution to these issues with a new problem called "the **covering problem**".

- higher levels and other structures.

Conclusion

What we learned:

We have to consider **more general problems** than membership.

Current and future work:

- the family of "approximation problems" is a jungle.
- our original separation algorithms are non-constructive.

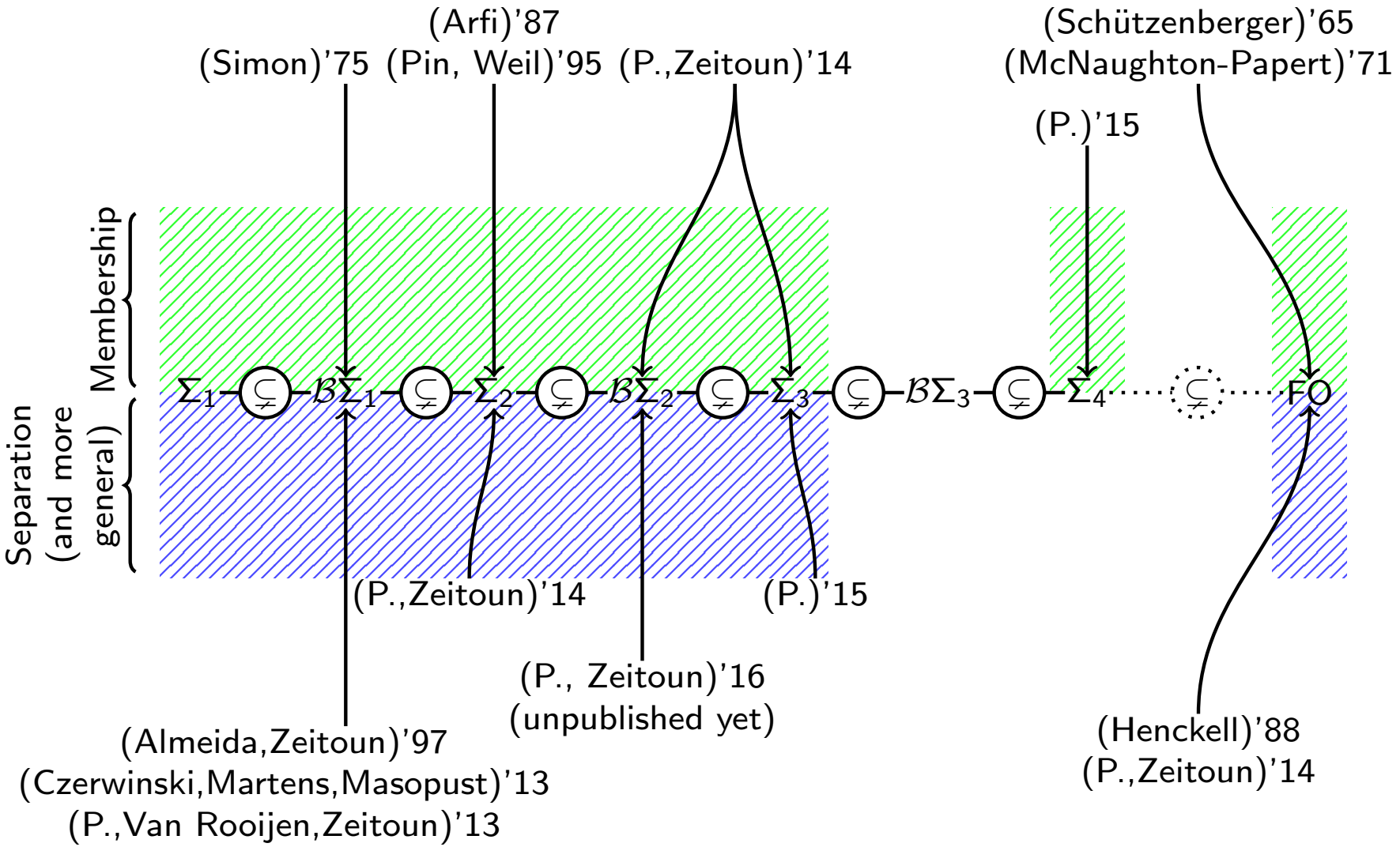
We think, we have found the solution to these issues with a new problem called "the **covering problem**".

- higher levels and other structures.

The big problem right now:

No transfer result seems possible for approximation problems

Conclusion


(Arfi)'87
(Simon)'75
(Pin, Weil)'95
(P.,Zeitoun)'14
(Schützenberger)'65
(McNaughton-Papert)'71
(P.)'15
Membership
Separation
(and more
general)
Σ_1
$\mathcal{B}\Sigma_1$
Σ_2
$\mathcal{B}\Sigma_2$
Σ_3
$\mathcal{B}\Sigma_3$
Σ_4
FO
(P.,Zeitoun)'14
(P.)'15
(P., Zeitoun)'16
(unpublished yet)
(Almeida,Zeitoun)'97
(Czerwinski,Martens,Masopust)'13
(P.,Van Rooijen,Zeitoun)'13
(Henckell)'88
(P.,Zeitoun)'14

Thank You