A

The Tale of the Quantifier Alternation Hierarchy of First-Order Logic over Words

T. Place and M. Zeitoun
LaBRI, U. Bordeaux

In this survey, we present ideas developed until recently in order to understand the expressive power of logical fragments in the quantifier alternation hierarchy of first-order logic interpreted on finite words.

1. INTRODUCTION

This paper surveys milestones, from early to recent results, on the expressiveness of fragments of first order logic interpreted on finite words. In this context, "understanding the expressive power of a fragment of first-order logic $\mathcal{F}$" often amounts to finding an algorithm for the following decision problem:

Input A regular language of finite words L.

Question Can L be defined by a sentence of $\mathcal{F}$?

In other words, we ask whether the input language belongs to the class of languages defined by the logic, hence the name of the problem: *the $\mathcal{F}$-membership problem*. Having an $\mathcal{F}$-membership algorithm in hand amounts to having an effective description of *all* regular properties that $\mathcal{F}$ can express. This is why obtaining a membership algorithm is viewed as the goal to strive for when trying to get a precise understanding of a logic. This problem is always difficult as it is a *semantic* question: whether a regular language is definable in $\mathcal{F}$ may not be apparent in the syntax that defines it.

Our main objective is to outline advances that have led to membership algorithms for logical fragments in a well-known and natural hierarchy of first order definable languages, the *quantifier alternation hierarchy*. Intuitively, the notion of quantifier alternation classifies first order logic according to the *difficulty* we have to define languages. To obtain such a classification, we need a notion of hardness, i.e., a good measure telling how complex a property is. Quantifier alternation is a natural such measure: a language is considered complicated when we need many switches between blocks of $\exists$ quantifiers and blocks of $\forall$ quantifiers to express it. In mathematics, usually few such alternations are used (one quickly gets lost beyond 4 or 5). This motivates the study of what can be expressed with a fixed number of quantifier alternations, and, already importantly, with few of them. Observe that the notion of quantifier alternation is, again, *semantic* and not syntactic: given a first order definable language, we look at the *minimum* number of quantifier alternations that is needed to define the language.

The search for membership algorithms for each level in this hierarchy started in 1965 when an algorithm was found for full first-order logic [Schützenberger 1965; McNaughton and Papert 1971]. This investigation is still ongoing work today, with results as recent as 2015 [Place 2015; Almeida et al. 2015]. The main developments were separated by decades. This is explained by the fact that each of them required to add new conceptual ingredients to the mix. In this paper, we survey the story of this quest by organizing it around three milestones, corresponding to the three main concepts that are needed to explain the most recent results.

2. LOGICAL FRAGMENTS OF FIRST ORDER LOGIC

For the whole paper, we assume that an arbitrary alphabet A is fixed and that we consider finite words over this alphabet. As usual, the set of all these words is denoted by A^*. A *language* is simply a set of words.

One can view a word as a logical structure made of an ordered sequence positions carrying labels in A. For example the word $abaac$ is made of positions $0 < 1 < 2 < 3 < 4$ labeled by a, b, a, a and c, respectively. In first-order logic over words (FO($<$)), one can quantify over positions in a word and use the following predicates to test properties of these positions:

— for each $a \in A$, a unary predicate P_a that selects positions labeled with an a.
— a binary predicate '$<$', which is interpreted as the linear order over the positions.

Moreover, as usual, one is allowed to use boolean connectives within sentences. Each sentence of FO($<$) defines a language: the language of all words that satisfy the sentence. For instance, the sentence "$\exists x\,(P_a(x) \wedge \exists y\, x < y)$" defines the language of all words in which there exists a position which carries an a and is not the rightmost one. Hence, FO($<$) defines a class of languages: the class of all languages that can be defined using an FO($<$) sentence. We also denote this class by FO($<$). In particular, FO($<$) is a subclass of that of all regular languages. This follows from the well-known theorem of Büchi, Elgot and Trakhtenbrot, which states that being regular is equivalent to being definable in the more expressive monadic second order logic (MSO) [Büchi 1960; Elgot 1961; Trakhtenbrot 1961].

The notion of quantifier alternation is a natural way to stratify the class FO($<$) as a hierarchy. It classifies first-order sentences by counting the number of alternations between existential and universal quantifiers inside their prenex normal form. An important remark is that one can find two different "quantifier alternation hierarchies of first-order logic" in the literature. They correspond to two different (but equivalent) ways of defining first-order logic over words. We start with the hierarchy corresponding to our definition, which we call the *order hierarchy*.

Order Hierarchy. As explained, one can classify first-order sentences by counting the number of alternations between $\exists$ and $\forall$ quantifiers in the prenex normal form of the sentence. Given $n \geqslant 1$, an FO($<$) sentence is said to be $\Sigma_n(<)$ (resp. $\Pi_n(<)$) if its prenex normal form has n blocs of nested quantifiers (or, equivalently, $(n-1)$ alternations) and starts with an $\exists$ (resp. $\forall$) quantifier. That is, a $\Sigma_n(<)$ sentence is a sentence whose prenex normal form has the following shape:

$$\underbrace{\exists^*\forall^*\exists^* \dots}_{n \text{ blocks}} \varphi$$

where φ is a quantifier-free FO($<$) formula. It is straightforward to see that the classes of languages corresponding to $\Sigma_n(<)$ and $\Pi_n(<)$ are closed under union and intersection. On the other hand, these classes are *not* closed under complement (the negation of $\Sigma_n(<)$ sentence is a $\Pi_n(<)$ sentence, and conversely). This is why one also considers the $\mathcal{B}\Sigma_n(<)$ sentences, which are boolean combinations of $\Sigma_n(<)$ and $\Pi_n(<)$ formulas.

Clearly, we have $\Sigma_n(<) \subset \mathcal{B}\Sigma_n(<) \subset \Sigma_{n+1}(<)$. The first natural question is whether these inclusions are strict, and it turns out that this is the case. The hierarchy is depicted in Figure 1, where the color indicates the status of the class with respect to the membership problem.

Enriched Hierarchy. In FO($<$), several natural predicates can be defined from the linear order "$<$". This is the case for the predicate testing that a position is the leftmost one in the word, by the formula $\min(x) := \neg\exists y\ y < x$, or symmetrically the rightmost one by a $\max(x)$ formula. Another natural predicate is the successor, which tests whether two positions are consecutive. It is defined by $+1(x, y) := (x < y) \wedge \neg\exists z(x < z \wedge z < y)$.

It follows that one can present an alternate definition of first-order logic over words, FO($<, +1$), in which these predicates are explicitly allowed in the signature. While the two definitions are equivalent for full first-order logic, this is not the case for

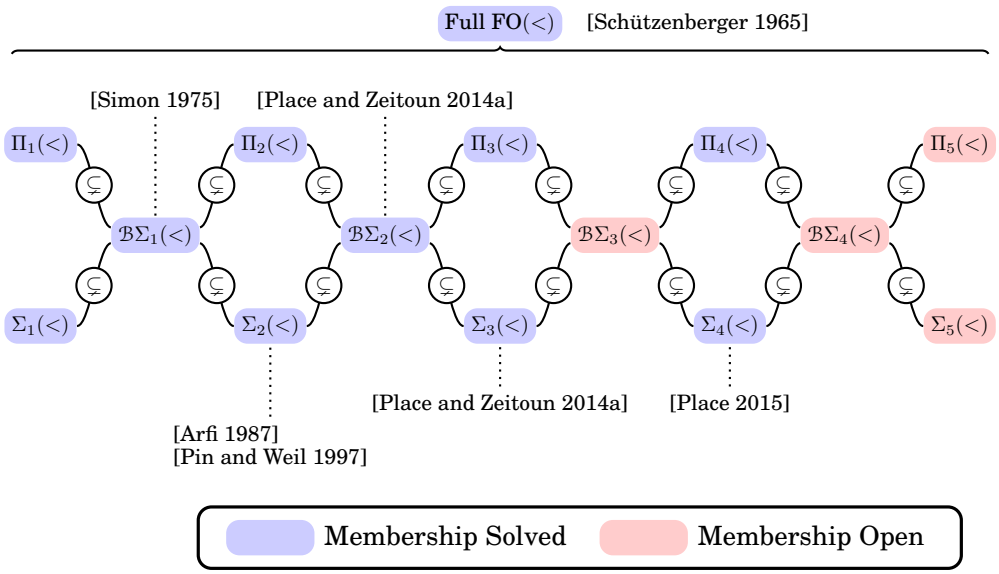

Fig. 1. Order Quantifier Alternation Hierarchy of First-Order Logic

classes of the quantifier alternation hierarchy. Indeed, replacing the predicates $\min(x)$, $\max(x)$ or $+1(x, y)$ by the formulas above inside a sentence may increase its quantifier alternation. It can be shown that this is actually unavoidable. Thus, we get a second quantifier alternation hierarchy. We call it the *enriched hierarchy* and denote its classes by $\Sigma_n(<, +1)$, $\Pi_n(<, +1)$ and $\mathcal{B}\Sigma_n(<, +1)$.

As we will see, the investigation of the membership problem for the two hierarchies is strongly related. In fact, the state of the art is the same for both, as depicted in Figure 1.

3. FIRST MILESTONE: SCHÜTZENBERGER'S THEOREM

We begin our survey with the milestone that started it all: Schützenberger's Theorem [Schützenberger 1965]. Naturally, before fragments in the quantifier alternation hierarchy of first-order logic were looked at, first-order logic was investigated as a whole. The main result in this investigation is Schützenberger's Theorem from which a membership algorithm for first-order logic can be easily obtained. This theorem was historically the first one and it defined a template that is still followed nowadays. Before we present the theorem and explain the relevance of the approach taken by Schützenberger, let us make an important remark.

In its original statement, Schützenberger's Theorem does not actually refer to first-order logic: it is about the class of *star-free* languages. This is the smallest class of languages containing all finite languages and closed under boolean operations (including complement) and concatenation product. On the other hand, the class is not closed under the Kleene star, hence the name "star-free". Therefore, originally, what followed from the theorem was a membership algorithm for the class of star-free languages. The connection with first-order logic was made later by McNaughton and Papert [McNaughton and Papert 1971] who proved that the classes of first-order definable languages and star-free languages are the same.

An important point is that while the membership algorithm for $\mathrm{FO}(<)$ is the consequence of two results, these two results are not of the same importance: the core of the argument is Schützenberger's proof. Indeed, that "$\mathrm{FO}(<) = $ Star-Free" is proved via

direct rewriting of formulas (essentially, this amounts to proving that concatenation can be simulated with existential quantification, and conversely). In other words, the proof is mainly *syntactic*: if we have an FO($<$) formula φ that defines a language L in hand, we do not have to analyze L, we can directly rewrite φ into a star-free description of L. On the other hand, proving Schützenberger's Theorem (i.e., deciding whether a regular language L is FO($<$)-definable or star-free) requires a *semantic* understanding of the input language L. In particular, given a regular language L that is first-order definable, one cannot directly compute an FO($<$) formula defining L out of any representation of L: usually this requires to first compute a specific abstract representation of L (its syntactic monoid) and then, to rebuild a first-order formula from this abstract representation.

This last remark underlines the fact that Schützenberger's Theorem is stated and proved within a general framework. This framework is tailored to the membership problem and is generic in the sense that it can be applied to many classes of languages besides FO($<$) itself. We start by presenting this framework and by explaining why it is that relevant for the membership problem.

3.1. A Framework for the Membership Problem: the Syntactic Approach

Our goal is now to define a framework that is suitable for investigating the membership problem associated to some class of languages $\mathcal{C}$. Recall that in the membership problem, we want to know whether a single input *regular* language is definable in $\mathcal{C}$. In order to understand Schützenberger's approach to this problem, one has to ask the following question: what is the advantage of having a *regular* input? The answer can be found in the following result, which is the main ingredient of Schützenberger's approach.

Main Ingredient: Myhill-Nerode Theorem. Myhill-Nerode Theorem is a well-known characterization of regular languages, as those whose *syntactic congruence* has finite index. To any language $L \subseteq A^*$ (not only regular ones), one can associate an equivalence relation $\sim_L$ on the set A^*: the *syntactic congruence of* L. Intuitively, two words are equivalent if they cannot be distinguished by L, even when embedded in the same context. Formally, the definition is as follows:

$$u \sim_L v \qquad \text{if and only if} \qquad \forall x, y \in A^* \quad xuy \in L \iff xvy \in L.$$

By definition, L is a union of equivalence classes of $\sim_L$. In other words, the *syntactic congruence* of L breaks down L in fundamental simpler parts. Furthermore, the set of these fundamental parts has a structure. Indeed, as the name suggests, it is simple to prove that the syntactic congruence is a congruence for the concatenation operation: if $u \sim_L v$ and $u' \sim_L v'$, then $uu' \sim_L vv'$. This has two consequences:

(1) The set of equivalence classes of $\sim_L$ is a monoid when equipped with the concatenation operation: if K, K' are equivalence classes, then KK' is included in an equivalence class. This monoid, denoted by M_L, is called the *syntactic monoid* of L.
(2) The map $\alpha_L : A^* \to M_L$ that associates its equivalence class to each word is a monoid morphism (for any $u, v \in A^*$, $\alpha_L(uv) = \alpha_L(u)\alpha_L(v)$). It is called the *syntactic morphism* of L.

The syntactic morphism α_L defines several languages: all languages which are unions of equivalence classes of $\sim_L$. We speak of the *set of languages recognized by* α_L. In particular, observe that both L and its complement are among these recognized languages. From Myhill-Nerode Theorem, we know that the syntactic monoid and the syntactic morphism of a language L become very relevant objects when L is regular.

THEOREM 3.1 (MYHILL-NERODE THEOREM). *Let L be a language. Then L is regular if and only if $\sim_L$ has finite index, i.e., if and only if the syntactic monoid of L is finite.*

Note that when M_L is finite, α_L can be finitely represented: as a morphism it is defined by its restriction to letters of the alphabet. Therefore, what Myhill-Nerode Theorem gives us is a finite canonical representation of any regular language: its syntactic morphism $\alpha_L : A^* \to M_L$. Moreover, the proof of the theorem is constructive: from any representation of a regular language L, one can compute α_L.

The main point of Schützenberger's approach to the membership problem is that the syntactic morphism is the "right" representation of a regular language when trying to decide semantic properties. In particular, it follows from Theorem 3.2 below that this representation is tailored to the investigation of the $\mathcal{C}$-membership problem for classes of languages $\mathcal{C}$ that satisfy the following conditions:

(1) $\mathcal{C}$ is nonempty and closed under boolean operations (including complement).
(2) $\mathcal{C}$ is closed under right and left quotients: for any $w \in A^*$ and $L \in \mathcal{C}$

$$w^{-1}L = \{u \mid wu \in L\} \in \mathcal{C} \quad \text{and} \quad Lw^{-1} = \{u \mid uw \in L\} \in \mathcal{C}.$$

Note that it is well-known and simple to verify that FO($<$) itself fulfills these two conditions.

THEOREM 3.2. *Let $\mathcal{C}$ be a nonempty class of regular languages that is closed under boolean operations and quotients. Then, for any regular language L, L belongs to $\mathcal{C}$ if and only if every language recognized by α_L belongs to $\mathcal{C}$.*

The statement of Theorem 3.2 might seem surprising as it is not entirely obvious that we gain something from it: we reduced the problem of deciding whether a *single* regular language L belongs to $\mathcal{C}$ to the problem of deciding whether *several* regular languages belong to $\mathcal{C}$. However, these languages are not any languages: combined, they form a finitely presentable piece of syntax that defines the input language L. In view of this, what we really obtain from Theorem 3.2 is that deciding a semantic property of L (whether L belongs to $\mathcal{C}$) can be reduced to deciding a syntactic property of the syntactic morphism α_L. Usually, such a result is called a decidable characterization of $\mathcal{C}$: one proves that membership of a regular language L in $\mathcal{C}$ is equivalent to an easily decidable syntactic property of its syntactic morphism α_L.

Naturally, the general approach that we have presented so far tells nothing of what this syntactic property should be, as it is specific to $\mathcal{C}$. This is where the real investigation on the class $\mathcal{C}$ takes place: finding the syntactic property and proving that it is equivalent to definability in $\mathcal{C}$ remains a hard problem for most classes $\mathcal{C}$. In other words, the syntactic morphism is a *suitable and convenient framework* for investigating the membership problem associated to $\mathcal{C}$:

(1) it provides an elegant way to state the algorithm and,
(2) it provides convenient tools and properties to prove its correction (we shall detail this point below).

On the other hand, *what the syntactic morphism is not* is a generic solution to the membership problem for all classes $\mathcal{C}$. The best (and historically first) example of this is Schützenberger's Theorem itself, which we now state.

THEOREM 3.3 (SCHÜTZENBERGER-MCNAUGHTON-PAPERT). *Let L be a regular language. Then, the three following conditions are equivalent:*

— *L is definable in* FO($<$).
— *L is star-free.*

— *M_L is aperiodic: for every $s \in M_L$, we have*

$$s^\omega = s^{\omega+1}. \tag{AP}$$

The symbol ω in (AP) denotes a natural number that can be computed from M_L, which satisfies the following property: for any $s \in M$, s^ω is an idempotent ($s^\omega = s^\omega s^\omega$). It follows that the last item in Theorem 3.3 is a syntactic condition which can easily be decided for M_L. Therefore, Theorem 3.3 yields the desired membership algorithm.

As we shall see in the next subsection, Theorem 3.3 is not an isolated case: many classes $\mathcal{C}$ were given decidable characterizations with theorems having an elegant statement resembling that of Theorem 3.3. This illustrates Item (1) above: the syntactic morphism provides an elegant way to state membership algorithms. However, the relevance of the syntactic approach of Schützenberger's Theorem does not stop at the statement of the result: as claimed in Item (2) above, the syntactic morphism also provides the right tools to prove such theorems. Let us explain why in more detail.

In order to prove a statement similar to Theorem 3.3 for some class $\mathcal{C}$ (such as $\mathrm{FO}(<)$ in the case of Theorem 3.3), one has to prove that for any regular language L, we have $L \in \mathcal{C}$ if and only if α_L satisfies some syntactic property (such as M_L is aperiodic in the case of Theorem 3.3). It turns out that for all known cases, including $\mathrm{FO}(<)$, the difficult direction is the "if" one: starting from an abstract representation of L satisfying an abstract property, one has to build a formula that defines L. In particular, this makes the proof of this direction very interesting: it yields a *canonical way* to construct a formula for any language of $\mathcal{C}$.

By Theorem 3.2, what one needs to prove is that any language recognized by α_L is definable in $\mathrm{FO}(<)$. Very roughly, this is achieved by decomposing each recognized language as the composition (by concatenation, union, intersection or complement) of simpler languages (i.e., recognized by a simpler syntactic morphism to which induction can be applied). This is where the algebraic structure of the syntactic morphism is useful: it may be used to define induction parameters and find clever ways to make these decompositions.

3.2. After Schützenberger: First Classes of the Hierarchies

Schützenberger's approach served as a template that was applied (and is still applied) to many natural classes of languages. In particular, since Schützenberger's Theorem was about star-free languages, researchers started considering hierarchies of star-free languages (which would later be proved to correspond to quantifier alternation hierarchies within $\mathrm{FO}(<)$).

The Enriched Hierarchy. The first hierarchy to be considered was the *Dot-Depth Hierarchy* of Brzozowski and Cohen [Brzozowski and Cohen 1971] in which each level counts the minimal number of nested alternations between concatenation and complement operations that are needed to build a star-free language. This hierarchy was proved to be strict (each level is strictly larger than the previous one) in [Brzozowski and Knast 1978]. Note that the link between this classification method and quantifier alternation is quite intuitive: since concatenation corresponds to existential quantification, alternating complement and concatenation is connected to alternating quantifiers. The formal connection was made later by Thomas [Thomas 1982], who proved that the dot-depth hierarchy corresponds to the enriched quantifier hierarchy of first-order logic: the languages of dot-depth i are the languages definable in $\mathcal{B}\Sigma_i(<, +1)$.

Remark 3.4. Observe that we did not mention the logics $\Sigma_i(<, +1)$ and $\Pi_i(<, +1)$. This is because the original definition of the dot-depth hierarchy by Brzozowski and Cohen did not include them. They were only added later as "half-levels": $\Sigma_i(<, +1)$

corresponds to languages of dot-depth $i - \frac{1}{2}$. We will detail this point in Section 4, as treating these classes requires to generalize the syntactic approach.

Despite its early definition (1971), it was not until 1983 that a membership algorithm for the first level of the dot-depth hierarchy was found (i.e., the level corresponding to $\mathcal{B}\Sigma_1(<,+1)$). This result, due to Knast [Knast 1983], was presented and proved with the syntactic approach: it states that a language is definable in $\mathcal{B}\Sigma_1(<,+1)$ if and only its syntactic morphism satisfies an easy to decide syntactic property, which can be stated as an equation similar to (AP).

Despite this initial success, level 2 of the dot-depth hierarchy (i.e., $\mathcal{B}\Sigma_2(<,+1)$) was not given a membership algorithm until 2014 [Place and Zeitoun 2014a]. In fact, after Knast's result, the focus was quickly shifted to the second hierarchy, the order hierarchy, following a result by Straubing [Straubing 1985] suggesting that investigating $\mathcal{B}\Sigma_2(<)$ was the right approach to solving membership for $\mathcal{B}\Sigma_2(<,+1)$.

The Order Hierarchy. Surprisingly, the simpler order quantifier alternation hierarchy was not considered until much later than the dot-depth hierarchy. It was first introduced independently by Thérien [Thérien 1981] and Straubing [Straubing 1981] as another hierarchy of star-free languages: the *Straubing-Thérien hierarchy*. It was then observed by Perrin and Pin [Perrin and Pin 1986] that it actually corresponds to the order hierarchy.

It was proved by Straubing [Straubing 1985] (for the integer $\mathcal{B}\Sigma_i$ levels) and then by Pin and Weil [Pin and Weil 2002] (for the half Σ_i levels) that this hierarchy is in a sense the most fundamental of the two. More precisely, they proved that for $i \geqslant 2$, the membership problem for $\mathcal{B}\Sigma_i(<,+1)$ (resp. $\Sigma_i(<,+1)$) can be effectively reduced to the same problem for $\mathcal{B}\Sigma_i(<)$ (resp. $\Sigma_i(<)$). This explains why, after 1985, efforts have mostly aimed at obtaining membership algorithms for levels of the order hierarchy (in fact the result of [Place and Zeitoun 2014a] cited above for $\mathcal{B}\Sigma_2(<,+1)$ is actually a membership algorithm for $\mathcal{B}\Sigma_2(<)$).

Ironically, the first level of the Straubing-Thérien hierarchy corresponding to $\mathcal{B}\Sigma_1(<)$ was given a membership algorithm before the actual hierarchy was even defined. Indeed, $\mathcal{B}\Sigma_1(<)$ corresponds to the class of piecewise testable languages, which was first investigated independently from the hierarchy and given a decidable characterization by Simon in [Simon 1975]. This result is usually referred to as *Simon's Theorem* (not to be confused with Simon's Factorization Forests Theorem [Simon 1990]) and is arguably the second most famous result of this kind, after Schützenberger's Theorem.

Many attempts at generalizing these results to $\mathcal{B}\Sigma_2(<)$ were made over the years (see [Pin 2011; Almeida and Klíma 2010; Pin 1998; Pin and Straubing 1985] for example). However, until 2014, the only result that was known was partial, working only when the alphabet has size 2 [Straubing 1988]. This can be explained by the fact that the algorithm that was finally obtained in [Place and Zeitoun 2014a] relies on two additional ingredients that fall outside of the syntactic approach that we presented thus far:

(1) Investigating the "half-levels" $\Sigma_i(<)$.
(2) Considering a problem that is more general than membership for these "half-levels": the separation problem.

The two following sections are devoted to the presentation of these two ingredients.

4. SECOND MILESTONE: CLASSES THAT ARE NOT CLOSED UNDER COMPLEMENT

An issue with the general approach that we outlined in the previous section is that it can only be applied to classes of languages that are closed under complement. Given a regular language L, the set of languages that are recognized by its syntactic morphism contains both L and its complement. Therefore, a class of languages $\mathcal{C}$ that satisfies

Theorem 3.2 (which is the basis of the syntactic approach) must be closed under complement.

This is a problem for the classes $\Sigma_i(<)$ and $\Sigma_i(<,+1)$ within the quantifier alternation hierarchies of first-order logic, as the associated classes of languages are not closed under complement (the negation of a $\Sigma_i(<)$ formula is a $\Pi_i(<)$ formula). At first, this was not a visible issue since the dot-depth hierarchy of star-free languages did not include the classes $\Sigma_i(<,+1)$ in its original definition (see Remark 3.4).

For the purpose of describing languages, this omission is natural: one always prefers to be allowed to use negation. However, one can argue that the logics Σ_i are the most fundamental ones in the two hierarchies. Indeed, the logics in the hierarchy (Σ_i, Π_i, and $\mathcal{B}\Sigma_i$) are all built directly from Σ_i. As we explained in the previous section, this argument is also validated empirically as all further membership results concerning both hierarchies are derived from the investigation of the classes $\Sigma_i(<)$.

The first question to be asked is whether the syntactic approach of Section 3 can be generalized to encompass classes that are not closed under complement, such as $\Sigma_i(<)$. This question was answered positively by Pin [Pin 1995] and the generalized approach was then used by Pin and Weil [Pin and Weil 1997] to obtain a membership algorithm for $\Sigma_2(<)$. In this section, we explain these results.

Let $\mathcal{C}$ be an arbitrary class of languages that satisfies the following conditions:

(1) $\mathcal{C}$ contains the languages $\emptyset$ and A^* and is closed under union and intersection (but not necessarily under complement).
(2) $\mathcal{C}$ is closed under right and left quotients: for any $w \in A^*$ and $L \in \mathcal{C}$

$$w^{-1}L = \{u \mid wu \in L\} \in \mathcal{C} \quad \text{and} \quad Lw^{-1} = \{u \mid uw \in L\} \in \mathcal{C}$$

Naturally, the interesting case is when $\mathcal{C}$ is not closed under complement, since other cases were treated in the previous section. Let us first detail why the syntactic approach fails in this case and outline Pin's solution to this problem.

Essentially, the relevance of the syntactic approach is justified by Theorem 3.2. As explained, when $\mathcal{C}$ is not closed under complement, we cannot hope to prove that L is definable in $\mathcal{C}$ if and only if **all** languages recognized by its syntactic morphism are (since these languages include the complement of L).

Pin's solution to this problem was to relax the "*all languages are definable*" condition in the theorem. He observed that the set of all recognized languages can be replaced by a subset that still retains a lot of structure. To define this subset, one needs to add a new ingredient to the mix: a canonical partial order that can be defined on the syntactic monoid of the language. The main idea behind the definition is that given a class $\mathcal{C}$, whether a regular language belongs to $\mathcal{C}$ only depends on the syntactic morphism of the language *and* on this order.

New Ingredient: The Syntactic Ordered Monoid. One can modify the definition of the syntactic congruence of a language to define a pre-congruence (i.e., a preorder that is compatible with concatenation):

$$u \leq_L v \qquad \text{if and only if} \qquad \forall x, y \in A^* \quad xuy \in L \Longrightarrow xvy \in L.$$

In turn, this defines a partial order $\leq$ on the set of equivalence classes of $\sim_L$ (i.e., the syntactic monoid M_L of L). It is also simple to verify that $\leq$ is compatible with the multiplication of M_L: $s \leq t$ and $s' \leq t'$ imply $st \leq s't'$. This means that the pair $(M_L, \leq)$ is an ordered monoid, called the *syntactic ordered monoid* of the language.

With our goal in mind, an important observation is that while a language L and its complement $\overline{L}$ have the same syntactic monoid ($M_L = M_{\overline{L}}$), they do not have the same syntactic *ordered* monoid. Indeed, it can be observed from the definition of the preorder

$\leq_L$ that the orders on M_L and $M_{\overline{L}}$ are dual:

$$u \leq_L v \quad \text{if and only if} \quad \forall x, y \in A^* \quad xvy \notin L \Longrightarrow xuy \notin L \quad \text{if and only if} \quad v \leq_{\overline{L}} u.$$

The following result proves that adding the syntactic ordered monoid makes it possible to recover Theorem 3.2 and the syntactic approach, even when the class $\mathcal{C}$ is not closed under complement. What one needs to consider is not the set of all languages recognized by α_L but only a subset that is defined from the order $\leq$ on M_L. We say that a language recognized by α_L is *upward-closed* if and only if it is a union of languages of the form $\cup_{K \leq K'} K'$ for some equivalence class K of $\sim_L$ (i.e., some element K of M_L). One can verify that L itself is upward-closed.

THEOREM 4.1 (PIN). *Let $\mathcal{C}$ be a class of languages containing $\emptyset$ and A^* and closed under union, intersection and quotients. Then, for any regular language L, L belongs to $\mathcal{C}$ if and only if all upward-closed languages recognized by α_L belong to $\mathcal{C}$.*

With this generalized theorem, one can hope to generalize the template established by Theorem 3.3 to classes $\mathcal{C}$ that are not closed under complement: deciding a semantic property of a regular language L (whether L belongs to $\mathcal{C}$) can be reduced to deciding a syntactic property of the syntactic morphism α_L. However, this syntactic property should depend on the newly introduced order on M_L. This is illustrated by the Theorem of Pin and Weil [Pin and Weil 1997], which gives a decidable characterization of $\Sigma_2(<)$. In the theorem, given a word $u \in A^*$, we denote by $\mathsf{alph}(u)$ its *alphabet* (i.e., the set of letters it contains).

THEOREM 4.2 (PIN-WEIL). *Let L be a regular language, then the two following conditions are equivalent:*

(*1*) *L is definable in $\Sigma_2(<)$.*
(*2*) *α_L satisfies the following property, for every $s, t \in M_L$ such that there exist two words $u \in \alpha_L^{-1}(s)$ and $v \in \alpha_L^{-1}(t)$ with $\mathsf{alph}(v) \subseteq \mathsf{alph}(u)$:*

$$s^\omega \leq s^\omega t s^\omega \tag{1}$$

This generalization of the syntactic approach was quite fruitful. First, by combining Theorem 4.2 with the transfer theorem of [Pin and Weil 2002], one obtains simple membership algorithms for $\Sigma_2(<)$, $\Pi_2(<)$, $\Sigma_2(<, +1)$ and $\Pi_2(<, +1)$.

Remark 4.3. Note that for these logics, alternate algorithms working outside of the syntactic approach are also known. An algorithm for $\Sigma_2(<)$ and $\Pi_2(<)$ is due to Arfi [Arfi 1987] and an algorithm for $\Pi_2(<, +1)$ and $\Pi_2(<, +1)$ is due to Glaßer and Schmitz [Glaßer and Schmitz 2000]. However, these algorithms are *ad hoc* (they use techniques that are specific to the logic they consider), and are much harder to present.

Another success of this generalization is that it allowed to obtain a new proof of Simon's Theorem (i.e., the decidable characterization of $\mathcal{B}\Sigma_1(<)$). While not a new result, this new proof [Henckell and Pin 2000] is of particular interest as it suggests a link between the investigation of $\Sigma_i(<)$ and that of $\mathcal{B}\Sigma_i(<)$. However, in order to formalize this relationship, one needs to consider a more general problem than membership. This is the purpose of the next section.

5. THIRD MILESTONE: SEPARATION

After the membership problem was solved for $\Sigma_2(<)$ in [Pin and Weil 1997], the next interesting classes, $\mathcal{B}\Sigma_2(<)$ and $\Sigma_3(<)$, turned out to be harder to tackle. As explained, while a partial solution for $\mathcal{B}\Sigma_2(<)$ (limited to languages over alphabets containing at most two letters) was found in [Straubing 1988], the general case remained open until

very recently. In [Place and Zeitoun 2014a], membership algorithms were found for both $\mathcal{B}\Sigma_2(<)$ and $\Sigma_3(<)$.

These two results required to add a new ingredient: a new decision problem, more general than membership and called the *separation problem*. It is the investigation of this new problem for the classes $\Sigma_2(<)$ and $\Pi_2(<)$ that made it possible to obtain the membership algorithms for $\mathcal{B}\Sigma_2(<)$ and $\Sigma_3(<)$. Let us start with the definition of the separation problem.

5.1. The Separation Problem

The separation problem is a generalization of the membership problem. This time, we are given *two* input regular languages rather than one. Given three languages $L_1, L_2, K \subseteq A^*$, we say that K *separates* L_1 from L_2 when K contains L_1 and is disjoint from L_2, as shown in Figure 2.

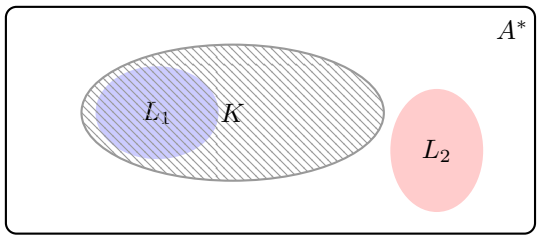

Fig. 2. K separates L_1 from L_2

Given an arbitrary class of languages $\mathcal{C}$, the $\mathcal{C}$-separation problem is defined as follows.

Input. Two regular languages L_1, L_2.
Question. Is L_1 $\mathcal{C}$-separable from L_2, i.e., does there exist $K \in \mathcal{C}$ separating L_1 from L_2?

The separation problem is an immediate generalization of the membership problem: one can effectively reduce the $\mathcal{C}$-membership problem to the $\mathcal{C}$-separation problem, since a regular language belongs to $\mathcal{C}$ if and only if it is $\mathcal{C}$-separable from its complement (which is regular as well).

Note that separation is also a problem that is intuitively harder than membership. In the membership problem, one can directly perform tests on the input language to discover whether or not it belongs to $\mathcal{C}$. In contrast, for the separation problem, we are searching for a hypothetical language in $\mathcal{C}$ that is unknown at first. In fact, given a class $\mathcal{C}$, obtaining an algorithm for the $\mathcal{C}$-separation problem usually requires to consider a broader framework, more general than what we have presented so far. We do not tackle this question and refer the reader to [Place and Zeitoun 2014a; Place and Zeitoun 2014b; Place 2015] for details, and to [Almeida 1999; Henckell 1988; Henckell et al. 2010] for a view of this problem in finite semigroup theory. Instead, we concentrate on why considering this problem allowed to make progress on the membership problem for $\Sigma_3(<)$ and $\mathcal{B}\Sigma_2(<)$. For reference, the situation for the order hierarchy updated with separation is presented in Figure 3.

The important point is that the core result in [Place and Zeitoun 2014a] is the separation algorithm for $\Sigma_2(<)$ and $\Pi_2(<)$. It is from this algorithm that membership algorithms for $\Sigma_3(<)$ and $\mathcal{B}\Sigma_2(<)$ are derived. We illustrate this by detailing the example of $\Sigma_3(<)$, which is easier to present, and actually allows to transfer decidability in a generic way.

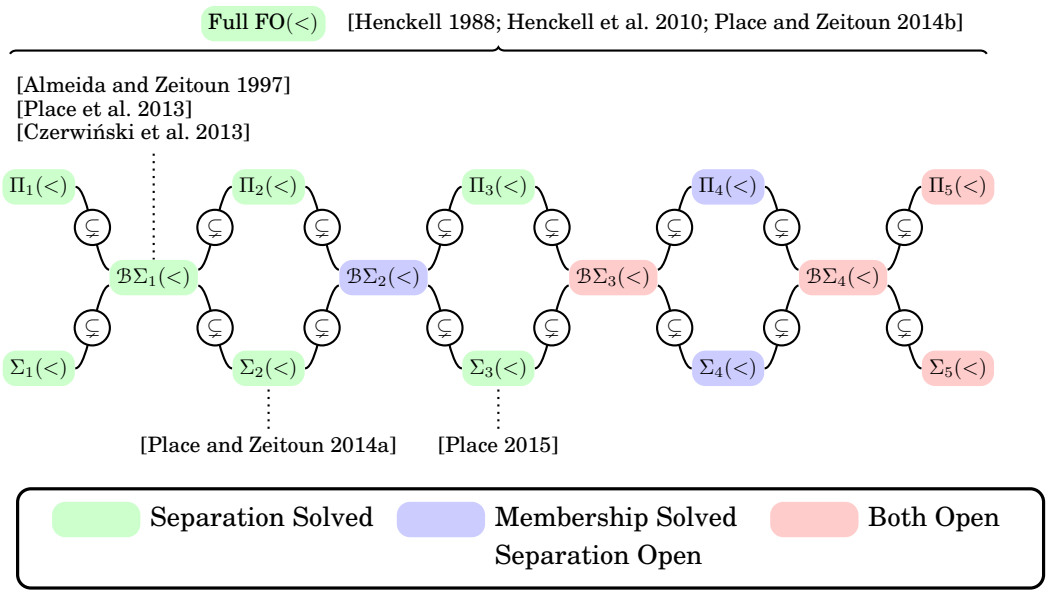

Fig. 3. Order Quantifier Alternation Hierarchy of First-Order Logic (references are for the separation results)

Remark 5.1. Our choice of presenting $\Sigma_3(<)$ over $\mathcal{B}\Sigma_2(<)$ is mainly practical. As we will see, the membership algorithm for $\Sigma_3(<)$ stems from a true transfer result: membership for $\Sigma_i(<)$ can be *effectively* reduced to separation for $\Sigma_{i-1}(<)$. This makes it possible to present this result without having to present the techniques used for solving $\Sigma_2(<)$-separation.

On the other hand, while there is also generic connection between separation for $\Sigma_i(<)$ and membership for $\mathcal{B}\Sigma_i(<)$ (actually even separation for $\mathcal{B}\Sigma_i(<)$), it is not immediate that this connection is effective. In fact, this is open for $i \geqslant 3$. In particular, this means that, effectively connecting membership for $\mathcal{B}\Sigma_2(<)$ to separation for $\Sigma_2(<)$ requires specific arguments, which turn out to be strongly tied to the $\Sigma_2(<)$-separation algorithm.

5.2. The Separation Problem and $\Sigma_i(<)$

We begin by explaining why considering the separation problem for $\Sigma_i(<)$ (or equivalently $\Pi_i(<)$) is relevant when considering the membership problem for $\Sigma_{i+1}(<)$. Recall the principle of the syntactic approach as outlined by Theorem 4.1: we want to use the fact that L is definable in $\Sigma_3(<)$ if and only if all upward-closed languages recognized by α_L are. By definition, the structure of a $\Sigma_i(<)$ sentence is "layered": the first layer is composed of $\Sigma_i(<)$ formulas, the second of $\Pi_{i-1}(<)$ formulas, the third of $\Sigma_{i-2}(<)$ formulas, and so on. The consequence of this is that deciding whether all upward-closed languages recognized by α_L are definable in $\Sigma_3(<)$ requires to first investigate some "$\Pi_{i-1}(<)$ problem" for these languages. Naturally, this problem has to be different and more general than membership: when L is $\Sigma_i(<)$ but not $\Pi_{i-1}(<)$, we cannot hope to prove that all languages definable by α_L are $\Pi_{i-1}(<)$ definable.

The results of [Place and Zeitoun 2014a] state that the separation problem for $\Pi_{i-1}(<)$ is a suitable more general problem. The $\Pi_{i-1}(<)$ layer is handled by deciding which pairs of languages among those recognized by α_L are $\Pi_{i-1}(<)$-separable. Intuitively, this amounts to computing the "best possible $\Pi_{i-1}(<)$-definable approximation" of the

languages recognized by α_L. This is illustrated by the statement of the theorem of [Place and Zeitoun 2014a] for $\Sigma_i(<)$ membership.

THEOREM 5.2 (PLACE-ZEITOUN). *Let L be a regular language and $i \geqslant 2$. Then the two following conditions are equivalent:*

(*1*) *L is definable in $\Sigma_i(<)$.*
(*2*) *α_L satisfies the following property, for every $s, t \in M_L$ such that $\alpha_L^{-1}(s)$ is* **not** *$\Pi_{i-1}(<)$-separable from $\alpha_L^{-1}(t)$:*

$$s^\omega \leq s^\omega t s^\omega \tag{2}$$

Theorem 5.2 states that for any i, the $\Sigma_i(<)$-membership problem can be effectively reduced to the $\Pi_{i-1}(<)$-separation problem. In [Place and Zeitoun 2014a], this theorem is combined with a separation algorithm for $\Sigma_2(<)$ and $\Pi_2(<)$ which yields a membership algorithm for $\Sigma_3(<)$ and $\Pi_3(<)$ (and in turn for $\Sigma_3(<, +1)$ and $\Pi_3(<, +1)$ by the transfer result of [Pin and Weil 2002]). Additionally, a separation algorithm was found recently for $\Sigma_3(<)$ and $\Pi_3(<)$ [Place 2015]. In the same way this algorithm yields membership algorithms for $\Sigma_4(<)$, $\Pi_4(<)$, $\Sigma_4(<, +1)$ and $\Pi_4(<, +1)$.

Remark 5.3. Note that Theorem 5.2 also provides a new $\Sigma_2(<)$-membership algorithm. However, the connection with separation was not noticed until $\Sigma_3(<)$ was investigated. The main reason for this is that the $\Pi_1(<)$-separation problem is quite simple. This simplicity entails that the condition: "$\alpha_L^{-1}(s)$ is **not** $\Pi_1(<)$-separable from $\alpha_L^{-1}(t)$" can be replaced by a much more elementary condition, as seen in Theorem 4.2.

6. THE FUTURE

The results of [Place and Zeitoun 2014a] seem to indicate that the membership problem might not be a general enough framework to investigate the quantifier hierarchies of first-order logic and that separation may be more appropriate. However, the situation is more complicated. After Theorem 5.2 was proved, a natural question was to know if it could be lifted to separation: can we define a problem $\mathcal{P}$, such that separation for $\Sigma_{i+1}(<)$ can be effectively reduced to $\mathcal{P}$ for $\Sigma_i(<)$? Or even better, can we choose separation as $\mathcal{P}$?

This question was investigated in [Place 2015]. While no definitive answer was found, the results of the paper suggest that the answer should be "*no*". Indeed, while a separation algorithm for $\Sigma_3(<)$ and $\Pi_3(<)$ is presented in [Place 2015], the technique used to obtain it does not work by reduction to an independent problem for $\Sigma_2(<)$. Instead, a new problem inside which the separation problems for both $\Sigma_2(<)$ and $\Sigma_3(<)$ are tied together is considered.

In short, this most likely means that the story of the quantifier alternation hierarchy is far from being over and could very well continue for several more decades.

REFERENCES

ALMEIDA, J. 1999. Some algorithmic problems for pseudovarieties. *Publicationes Mathematicae Debrecen 54*, 531–552.

ALMEIDA, J., BARTONOVA, J., KLIMA, O., AND KUNC, M. 2015. On decidability of intermediate levels of concatenation hierarchies. In *Proceedings of the 19th International Conference on Developments in Language Theory (DLT 2015)*. Springer, Berlin, Heidelberg.

ALMEIDA, J. AND KLÍMA, O. 2010. New decidable upper bound of the 2nd level in the Straubing-Thérien concatenation hierarchy of star-free languages. *Discrete Mathematics & Theoretical Computer Science 12,* 4, 41–58.

ALMEIDA, J. AND ZEITOUN, M. 1997. The pseudovariety **J** is hyperdecidable. *RAIRO Inform. Théor. Appl. 31,* 5, 457–482.

ARFI, M. 1987. Polynomial operations on rational languages. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science*. STACS'87. Springer-Verlag, Berlin, Heidelberg, 198–206.

BRZOZOWSKI, J. A. AND COHEN, R. S. 1971. Dot-depth of star-free events. *Journal of Computer and System Sciences 5,* 1, 1–16.

BRZOZOWSKI, J. A. AND KNAST, R. 1978. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences 16,* 1, 37–55.

BÜCHI, J. R. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly 6,* 1-6, 66–92.

CZERWIŃSKI, W., MARTENS, W., AND MASOPUST, T. 2013. Efficient separability of regular languages by subsequences and suffixes. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming*. ICALP'13. Springer-Verlag, Berlin, Heidelberg, 150–161.

ELGOT, C. C. 1961. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society 98,* 1, 21–51.

GLASSER, C. AND SCHMITZ, H. 2000. Languages of dot-depth 3/2. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science*. (STACS'00). Springer, Berlin, Heidelberg, 555–566.

HENCKELL, K. 1988. Pointlike sets: the finest aperiodic cover of a finite semigroup. *J. Pure Appl. Algebra 55*, 85–126.

HENCKELL, K. AND PIN, J.-E. 2000. Ordered monoids and j-trivial monoids. In *Algorithmic Problems in Groups and Semigroups*, J.-C. Birget, S. Margolis, J. Meakin, and M. Sapir, Eds. Trends in Mathematics. Birkhauser, 121–137.

HENCKELL, K., RHODES, J., AND STEINBERG, B. 2010. Aperiodic pointlikes and beyond. *Internat. J. Algebra Comput. 20,* 2, 287–305.

KNAST, R. 1983. A semigroup characterization of dot-depth one languages. *RAIRO - Theoretical Informatics and Applications 17,* 4, 321–330.

MCNAUGHTON, R. AND PAPERT, S. A. 1971. *Counter-Free Automata*. MIT Press.

PERRIN, D. AND PIN, J.-E. 1986. First-order logic and star-free sets. *Journal of Computer and System Sciences 32,* 3, 393–406.

PIN, J.-E. 1995. A variety theorem without complementation. *Russian Mathem. (Iz. VUZ) 39*, 74–83.

PIN, J.-E. 1998. Bridges for concatenation hierarchies. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*. ICALP'98. Springer-Verlag, Berlin, Heidelberg, 431–442.

PIN, J.-E. 2011. Theme and variations on the concatenation product. In *Proceedings of the 4th International Conference on Algebraic Informatics*. CAI'11. Springer-Verlag, Berlin, Heidelberg, 44–64.

PIN, J.-E. AND STRAUBING, H. 1985. Monoids of upper triangular boolean matrices. In *Semigroups. Structure and Universal Algebraic Problems*. Vol. 39. North-Holland, 259–272.

PIN, J.-E. AND WEIL, P. 1997. Polynomial closure and unambiguous product. *Theory of Computing Systems 30,* 4, 383–422.

PIN, J.-E. AND WEIL, P. 2002. The Wreath Product Principle for Ordered Semigroups. *Communications in Algebra 30*, 5677–5713.

PLACE, T. 2015. Separating regular languages with two quantifier alternations. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'15)*. ACM, New York, NY, USA.

PLACE, T., VAN ROOIJEN, L., AND ZEITOUN, M. 2013. Separating regular languages by piecewise testable and unambiguous languages. In *Proceedings of the 38th International Symposium on Mathematical Foundations of Computer Science*. MFCS'13. Springer-Verlag, Berlin, Heidelberg, 729–740.

PLACE, T. AND ZEITOUN, M. 2014a. Going higher in the first-order quantifier alternation hierarchy on words. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming*. ICALP'14. Springer-Verlag, Berlin, Heidelberg, 342–353.

PLACE, T. AND ZEITOUN, M. 2014b. Separating regular languages with first-order logic. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic (CSL'14) and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS'14)*. ACM, New York, NY, USA, 75:1–75:10.

SCHÜTZENBERGER, M. P. 1965. On finite monoids having only trivial subgroups. *Information and Control 8,* 2, 190–194.

SIMON, I. 1975. Piecewise testable events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*. Springer-Verlag, Berlin, Heidelberg, 214–222.

SIMON, I. 1990. Factorization forests of finite height. *Theoritical Computer Science 72,* 1, 65–94.

STRAUBING, H. 1981. A generalization of the schützenberger product of finite monoids. *Theoretical Computer Science 13,* 2, 137–150.

STRAUBING, H. 1985. Finite semigroup varieties of the form V $*$ D. *Journal of Pure and Applied Algebra 36*, 53–94.

STRAUBING, H. 1988. Semigroups and languages of dot-depth two. *Theoretical Computer Science 58,* 1-3, 361–378.

THÉRIEN, D. 1981. Classification of finite monoids: The language approach. *Theoretical Computer Science 14,* 2, 195–208.

THOMAS, W. 1982. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences 25,* 3, 360–376.

TRAKHTENBROT, B. A. 1961. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR 149*, 326–329. In Russian.