Going Higher in First-Order Quantifier Alternation Hierarchies on Words

THOMAS PLACE, LaBRI, Bordeaux University and Institut Universitaire de France, France
MARC ZEITOUN, LaBRI, Bordeaux University, France

We investigate quantifier alternation hierarchies in first-order logic on finite words. Levels in these hierarchies are defined by counting the number of quantifier alternations in formulas. We prove that one can decide membership of a regular language in the levels $\mathcal{B}\Sigma_2$ (finite Boolean combinations of formulas having only one alternation) and Σ_3 (formulas having only two alternations and beginning with an existential block). Our proofs work by considering a deeper problem, called *separation*, which, once solved for lower levels, allows us to solve membership for higher levels.

CCS Concepts: • **Theory of computation → Formal languages and automata theory**; **Regular languages**; **Logic**;

Additional Key Words and Phrases: First-order logic, regular languages, decidable characterization, membership problem, separation problem, quantifier alternation, logical hierarchies, dot-depth hierarchy, Straubing-Thérien hierarchy

ACM Reference format:
Thomas Place and Marc Zeitoun. 2019. Going Higher in First-Order Quantifier Alternation Hierarchies on Words. *J. ACM* 66, 2, Article 12 (March 2019), 65 pages.
https://doi.org/10.1145/3303991

1 INTRODUCTION

The connection between logic and automata theory is well known and has a fruitful history in theoretical computer science. It was first observed when Büchi [22], Elgot [31], and Trakhtenbrot [120] proved independently that regular languages of finite words are exactly languages that can be defined by a monadic second-order logic (MSO) sentence. Since then, many efforts have been devoted to the investigation and understanding of the expressive power of relevant fragments of MSO. In this field, the yardstick result is often to prove a *decidable characterization*, i.e., to design an algorithm which, given as input a regular language, decides whether it can be defined within the fragment under investigation. This decision problem is called the *membership problem*. More than the algorithm itself, the main motivation for solving it is the insight given by its proof. Indeed, in order to prove a decidable characterization, one has to consider and understand *all* properties that can be expressed in the fragment.

This work was supported by the DeLTA project (ANR-16-CE40-0007).
Authors' addresses: T. Place, LaBRI, Bordeaux University, 351 cours de la Libération, 33405 Talence Cedex, France and Institut Universitaire de France, France; email: tplace@labri.fr; M. Zeitoun, LaBRI, Bordeaux University, 351 cours de la Libération, 33405 Talence Cedex, France; email: mz@labri.fr.

0004-5411/2019/03-ART12 $15.00
https://doi.org/10.1145/3303991

The most prominent fragment of MSO is first-order logic (FO(<), or FO for short) equipped with a predicate "<" for the linear order. This logic was first investigated on finite words by McNaughton and Papert [54], who showed that a language is FO definable if and only if it is *star-free*, that is, if and only if it can be defined from singleton languages using Boolean operations and concatenation (but *not* the Kleene star, hence, the name). As such, this result just amounts to a simple syntactic translation, which does not provide any insight on the expressive power of first-order logic. However, together with an earlier result from Schützenberger [91], it yields a decidable characterization. Schützenberger's Theorem states that a regular language is star-free if and only if its syntactic monoid is *aperiodic*. The syntactic monoid is a finite algebraic structure that can be effectively computed from any representation of the language. Moreover, aperiodicity can be rephrased as an equation that needs to be satisfied by all elements of the monoid. Therefore, Schützenberger's Theorem together with McNaughton-Papert's result entails decidability of first-order definability.

Quantifier Alternation. Schützenberger's proof also provides an algorithm that, given a regular language, outputs a first-order sentence (of course, when the input language is first-order definable). However, this sentence may be unnecessarily complicated. The next natural step consists of requiring the output sentence to be "as simple as possible." To make this question precise, one needs a meaningful notion of complexity.

The most appropriate parameter for classifying first-order definable languages according to the difficulty of defining them is their *quantifier alternation*. The quantifier alternation of a *formula* is simply the maximal number of switches between blocks of existential quantifiers and blocks of universal quantifiers in its prenex normal form. The quantifier alternation of a *language* definable in FO is the smallest quantifier alternation of a first-order sentence that defines it. Observe that the quantifier alternation of a *language* is, like first-order definability, a semantic notion (in contrast to the quantifier alternation of a *formula*, which is a syntactic notion). This explains why it is not straightforward to compute it from a representation of the language.

It is intuitive that formulas involving several alternations are difficult to grasp—one usually uses only few of them to state mathematical properties. This intuition is supported by results showing that, indeed, this parameter is meaningful, i.e., that languages of high quantifier alternation are "hard" to deal with. The algorithmic treatment of first-order formulas involves an unavoidable non-elementary lower bound [86, 99, 100]. This is the case, for instance, for the satisfiability problem. Likewise, the number of states of the minimal automaton equivalent to an FO formula may be non-elementarily large in the size of the formula. This blowup is due to quantifier alternation, since restricting these problems to formulas of bounded quantifier alternation yields elementary decision procedures.

This motivates the investigation of what can be expressed with a *fixed* number of quantifier alternations and, already importantly, with few of them. This is what we do in this article: we investigate the hierarchy inside FO obtained by classifying languages according to their quantifier alternation. More precisely, the hierarchy involves the classes $\Sigma_i(<)$, $\mathcal{B}\Sigma_i(<)$, and $\Delta_i(<)$, defined as follows:

- An FO(<) formula is $\Sigma_i(<)$ if its prenex normal form has $(i-1)$ quantifier alternations and starts with a block of existential quantifiers or if it has strictly less than $(i-1)$ quantifier alternations. A language is $\Sigma_i(<)$ if it can be defined by a $\Sigma_i(<)$ sentence.
- A formula is $\mathcal{B}\Sigma_i(<)$ if it is a finite Boolean combination of $\Sigma_i(<)$ formulas. A language is $\mathcal{B}\Sigma_i(<)$ if it can be defined by a $\mathcal{B}\Sigma_i(<)$ sentence.
- Finally, a language is $\Delta_i(<)$ if it can be defined by both a $\Sigma_i(<)$ sentence and the negation of a $\Sigma_i(<)$ sentence. Note that there is no notion of a "$\Delta_i(<)$ formula."

The quantifier alternation hierarchy is known to be strict:

$$\Delta_i(<) \subsetneq \Sigma_i(<) \subsetneq \mathcal{B}\Sigma_i(<) \subsetneq \Delta_{i+1}(<).$$

This well-known hierarchy thus defines a complexity measure of first-order definable languages: complex ones are those requiring several quantifier alternations.

Another motivation for investigating this hierarchy is its ties with two other famous hierarchies in formal language theory, defined in terms of regular expressions. Roughly speaking, levels in both of these hierarchies count the number of alternations between Boolean operations and concatenation products that are necessary to express a language (recall that, by McNaughton-Papert's Theorem, every first-order definable language can be built from singleton languages using union, concatenation, and Boolean operations). In the first of these hierarchies, the *Straubing-Thérien hierarchy* [101, 109], level i exactly corresponds to the class $\mathcal{B}\Sigma_i(<)$, as shown by Perrin and Pin [58]. In the second one, the *dot-depth hierarchy*, which was actually defined earlier by Brzozowski and Cohen [18], level i corresponds to augmenting the logic $\mathcal{B}\Sigma_i(<)$ with a predicate for the successor relation, as shown by Thomas [113]. These correspondences show that proving decidability of the membership problem for $\mathcal{B}\Sigma_2(<)$ immediately entails its decidability for level 2 in the Straubing-Thérien hierarchy but also in the dot-depth hierarchy, thanks to a reduction due to Straubing [102]. We refer the reader to Section 3 for details.

Many efforts have been devoted to finding decidable characterizations for levels in the quantifier alternation hierarchy. Despite these efforts, however, only the lower ones are known to be decidable. The class $\mathcal{B}\Sigma_1(<)$ consists exactly of all piecewise testable languages, i.e., such that membership of a word depends on its scattered subwords only up to a fixed size. These languages were characterized by Simon [94] as those whose syntactic monoid is $\mathcal{J}$-trivial. A decidable characterization of $\Sigma_2(<)$—hence, of $\Delta_2(<)$ as well—was obtained by Arfi [8, 9], a problem revisited and clarified by Pin and Weil [71, 76], who also set up a generic algebraic framework to work with. For $\Delta_2(<)$, the literature is very rich: see the survey by Tesson and Thérien [108]. For example, the $\Delta_2(<)$ definable languages are exactly the ones definable in the two-variable restriction of FO(<) [112]. These are also the languages whose syntactic monoid belongs to the class DA, as shown again by Pin and Weil [71, 76] (see also [92]). For higher levels in the hierarchy, getting decidable characterizations remained a major open problem. In particular, the case of $\mathcal{B}\Sigma_2(<)$ has a very abundant history, and a series of combinatorial, logical, and algebraic conjectures have been proposed over the years. We refer to Section 3 and to several surveys cited in this section for a bibliography. So far, the only known effective result has been partial, working only when the alphabet is of size 2 [104].

Contributions. In this article, we establish decidable characterizations for the fragments $\mathcal{B}\Sigma_2(<)$, $\Delta_3(<)$, and $\Sigma_3(<)$ of first-order logic. These new results are based on a deeper decision problem than membership: the *separation problem*. Fix a class $\mathcal{C}$ of languages. The $\mathcal{C}$-separation problem amounts to deciding whether, given two input regular languages, there exists a third language in $\mathcal{C}$ containing the first language while being disjoint from the second one. Solving the $\mathcal{C}$-separation problem is more general than obtaining a decidable characterization for the class $\mathcal{C}$. Since regular languages are effectively closed under complement, testing membership in $\mathcal{C}$ can be achieved by testing whether the input is $\mathcal{C}$-separable from its complement. While this reduction immediately transfers decision procedures for one problem to the other, this is not our primary motivation for looking at separation. Although intrinsically more challenging, a solution to the separation problem requires more understanding than just getting a decidable characterization. This understanding for a given fragment can then be exploited in order to obtain decidable characterizations for extensions built on top of this fragment.

Historically, the separation problem for regular languages was first investigated as a special case of a deep problem in semigroup theory, the problem of computing the *pointlike subsets* of a finite monoid, solved for several cases by relying on purely algebraic and topological arguments [7, 35, 37]. It was only identified as a variant of the separation problem by Almeida [4]. Recently, a research effort has been made to investigate this problem from a radically different perspective, with the aim of finding new and self-contained proofs relying on elementary ideas and notions from language theory only. Such proofs were obtained for several results already known in the algebraic framework [28, 79, 80, 82, 85]. This article is a continuation of this effort for classes that were not solved even in the algebraic setting: we solve the separation problem for $\Sigma_2(<)$, and we use our solution as a basis to obtain decidable characterizations for the classes $\mathcal{B}\Sigma_2(<)$, $\Delta_3(<)$, and $\Sigma_3(<)$.

Our proof works as follows: given two regular languages, one can easily construct a morphism α from A^* into a finite monoid M that recognizes both languages. We then design an algorithm that computes, inside the finite monoid M, enough Σ_2-related information to answer the $\Sigma_2(<)$-separation question for *every* pair of languages that are recognized by α. It turns out that it is also possible to use this information to obtain decidability of $\Delta_3(<)$, $\Sigma_3(<)$, and $\mathcal{B}\Sigma_2(<)$ (though this last characterization is much more difficult). This information amounts to the notion of Σ_2-*chain*, our main tool in the article. A Σ_2-chain is an *ordered sequence* $s_1, \ldots, s_n \in M$ that witnesses a property of α with respect to $\Sigma_2(<)$. Let us give some intuition in the case $n = 2$—which is enough to make the link with Σ_2-separation. A sequence s_1, s_2 of elements of M is a Σ_2-chain if every $\Sigma_2(<)$ language containing all words in $\alpha^{-1}(s_1)$ intersects $\alpha^{-1}(s_2)$. In terms of separation, this means that $\alpha^{-1}(s_1)$ is *not* separable from $\alpha^{-1}(s_2)$ by a $\Sigma_2(<)$ definable language. This notion can actually be extended to every level of the hierarchy.

This article contains three main separate, new, and nontrivial results:

(1) An algorithm to compute Σ_2-chains—hence, $\Sigma_2(<)$-separability is decidable.
(2) A transfer result showing that an algorithm to compute Σ_i-chains of length 2 entails a decidable characterization of $\Sigma_{i+1}(<)$. In particular, by (1), membership in $\Sigma_3(<)$ is decidable. Decidability of $\Pi_3(<)$, the dual of $\Sigma_3(<)$, and of $\Delta_3(<)$ are then immediate.
(3) A decidable characterization of $\mathcal{B}\Sigma_2(<)$.

For (1), computing Σ_2-chains is achieved using a fixed-point algorithm that starts with trivial Σ_2-chains such as $s, s, \ldots, s$ and iteratively computes more Σ_2-chains until a fixed point is reached. For our technique to work, we actually have to consider a notion slightly more general than Σ_2-chains. The completeness proof of this algorithm relies on the Factorization Forest Theorem of Simon [95]. This is not surprising (even though one can actually bypass its use), as the link between this theorem and the quantifier alternation hierarchy was already observed by Pin and Weil [76] and Bojańczyk [15].

For (2), we establish a characterization of $\Sigma_3(<)$ in terms of an equation on the syntactic monoid of the language. This equation is parametrized by the set of Σ_2-chains of length 2. In other words, we use Σ_2-chains to abstract an infinite set of equations into a single one. The proof relies again on the Factorization Forest Theorem of Simon [95] and is actually generic to all levels in the hierarchy. This means that, for any level i, we define a notion of Σ_i-chain and characterize $\Sigma_{i+1}(<)$ using an equation parametrized by Σ_i-chains of length 2. However, decidability of $\Sigma_{i+1}(<)$ depends on our ability to compute all Σ_i-chains of length 2, which we can do only for $i = 2$.

Finally, for (3), the decidable characterization of $\mathcal{B}\Sigma_2(<)$ is the most difficult result of the article. As for $\Sigma_3(<)$, it is presented by two equations parametrized by Σ_2-chains (of lengths 2 and 3). However, the characterization is, this time, specific to the case $i = 2$. This is because most of our proof relies on a careful analysis of our algorithm that computes Σ_2-chains, which works only for

$i = 2$. The equations share surprising similarities with the ones used by Bojańczyk and the first author [16] to characterize a totally different formalism: Boolean combinations of open sets of infinite trees. In [16] also, the authors present their characterization as a set of equations parametrized by a notion of "chain" for open sets of infinite trees (although their "chains" are not explicitly identified as a separation relation). Since the formalisms are of a different nature, the way that these chains and our Σ_2-chains are constructed are completely independent, which means that the proofs are also mostly independent. However, once the construction analysis of chains has been done, several combinatorial arguments used to make the link with equations are analogous. In particular, we reuse and adapt definitions from [16] to present these combinatorial arguments in our proof. One could say that the proofs are both (very different) setups to apply similar combinatorial arguments in the end.

Our results are shown using the ordering relation "$<$" on positions as the only numerical predicate of the signature in the logic. In full first-order logic, one can define other natural numerical predicates, such as the first and last positions, as well as the successor relation. However, defining these predicates requires an additional quantification. It is known that enriching the signature with these predicates increases the expressiveness of each fragment in the quantifier alternation hierarchy. This yields another hierarchy inside first-order logic, which has also been investigated in the literature. In particular, it has been shown by Thomas [113] to correspond to the so-called *dot-depth* hierarchy defined by Brzozowski and Cohen [18] in terms of regular constructs needed to build a star-free language. In Section 12, we present already known results to show that all decidability statements obtained for the original hierarchy can be lifted to the hierarchy where the additional predicates are allowed. This works both for decidable characterizations [83, 102] and for separation [83, 96].

Organization. Sections 2 and 3 are devoted to the presentation of the problem that we investigate. In Section 2, we define the quantifier alternation hierarchies and precisely state this problem. Section 3 presents an outline of the rich history of these problems, viewed from different perspectives.

In Sections 4 and 5, we develop the machinery necessary for the statements and the proofs of our results. Section 4 is devoted to the presentation of well-known classical tools, such as Ehrenfeucht-Fraïssé games, monoids, and Simon's Factorization Forest Theorem, while Section 5 introduces a new tool specific to this article: Σ_i-chains.

The remaining sections present and prove our results. In Section 6, we reduce the membership and separation problems for all levels in the hierarchy to the problem of computing Σ_i-chains. In the following sections, we then prove that these problems can be solved for specific levels. In Section 7, we obtain a solution to separation for $\Sigma_2(<)$ and to membership for $\Sigma_3(<)$, $\Pi_3(<)$, and $\Delta_3(<)$. Then, in Section 8, we obtain a solution to membership for $\mathcal{B}\Sigma_2(<)$. Sections 9 to 11 are then devoted to the difficult proof of the decidable characterization of $\mathcal{B}\Sigma_2(<)$. In the last section, Section 12, we lift up our results to the hierarchy with successor, using previously known transfer results.

This article is the full version of [81].

2 QUANTIFIER ALTERNATION HIERARCHIES

As explained in the introduction, we study two decision problems, called *membership* and *separation*, to investigate two famous hierarchies of classes of languages. In this section, we precisely define these hierarchies and decision problems. Note that the section is devoted to definitions only. We shall present the history of these hierarchies in Section 3.

This section is organized into two parts. We begin by giving a logical definition of our two hierarchies: they classify first-order definable languages by counting the number of quantifier

alternations that are needed for defining these languages. Equivalent combinatorial definitions in terms of star-free languages will be presented in Section 3. In the second part, we define the membership problem and the separation problem.

2.1 Quantifier Alternation Hierarchies of First-Order Logic

Throughout the article, we assume fixed a finite alphabet A. We denote by A^* the set of all words over A (including the empty word ε) and by A^+ the set of all nonempty words over A. If $u, v \in A^*$ are words over A, we denote by $u \cdot v$ or uv the word obtained by concatenation of u and v and by $\mathrm{alph}(u)$ the alphabet of u, i.e., the smallest subset B of A such that $u \in B^*$. A *language over A* is a subset of A^*. In this article, we work with regular languages. These languages have several equivalent characterizations, as they can be defined by

— *monadic second-order logic*,
— *finite automata*,
— *regular expressions*,
— *finite monoids*.

The two hierarchies that we investigate in this article are contained within a strict subclass of regular languages that we define now: the class of first-order definable languages.

First-Order Logic. We view words as logical structures made of a sequence of positions. Each position has a label in the alphabet A and can be quantified. We denote by "$<$" the linear order over the positions. We work with first-order logic, FO($<$), using the following predicates:

— for each $a \in A$, a unary predicate P_a that selects positions labeled with an a.
— a binary predicate "$<$" for the linear order.

To every first-order sentence φ, one can associate the language $\{w \in A^* \mid w \models \varphi\}$ of words that satisfy φ. For instance, the sentence $\exists x P_a(x)$ defines the language of all words having at least one "a." Hence, FO($<$) defines a class of languages: the class of all languages that can be defined by an FO($<$) sentence. For the sake of simplifying the presentation, we will abuse notation and use FO($<$) to denote both the logic and the associated class of languages.

Order Hierarchy. One classifies first-order formulas by counting the number of alternations between existential and universal quantifiers in the prenex normal form of the formula. For $i \in \mathbb{N}$, a formula is said to be $\Sigma_i(<)$ (resp., $\Pi_i(<)$) if its prenex normal form has either

— *exactly* $(i-1)$ quantifier alternations (i.e., exactly i quantifier blocks) and starts with an existential quantifier (resp., with a universal quantifier) or
— *strictly less* than $(i-1)$ quantifier alternations (i.e., strictly less than i quantifier blocks).

For example, a formula whose prenex normal form is

$$\forall x_1 \forall x_2 \exists x_3 \forall x_4 \ \varphi(x_1, x_2, x_3, x_4) \quad \text{(with } \varphi \text{ quantifier-free)}$$

is $\Pi_3(<)$. Observe that a $\Pi_i(<)$ formula is by definition the negation of a $\Sigma_i(<)$ formula. Finally, a $\mathcal{B}\Sigma_i(<)$ formula is a finite Boolean combination of $\Sigma_i(<)$ formulas. As for full first-order logic, we will abuse notations and use $\Sigma_i(<)$, $\Pi_i(<)$, and $\mathcal{B}\Sigma_i(<)$ to denote both the logics and the associated classes of languages. Finally, we denote by $\Delta_i(<)$ the class of languages that can be defined by *both* a $\Sigma_i(<)$ and a $\Pi_i(<)$ formula[1]. It is known [58] that this gives a strict infinite hierarchy of classes of languages as represented in Figure 1. In the article, we call this hierarchy the *order hierarchy*.

[1]Note that, strictly speaking, $\Delta_i(<)$ is not a logic: there is no notion of a "$\Delta_i(<)$ formula."

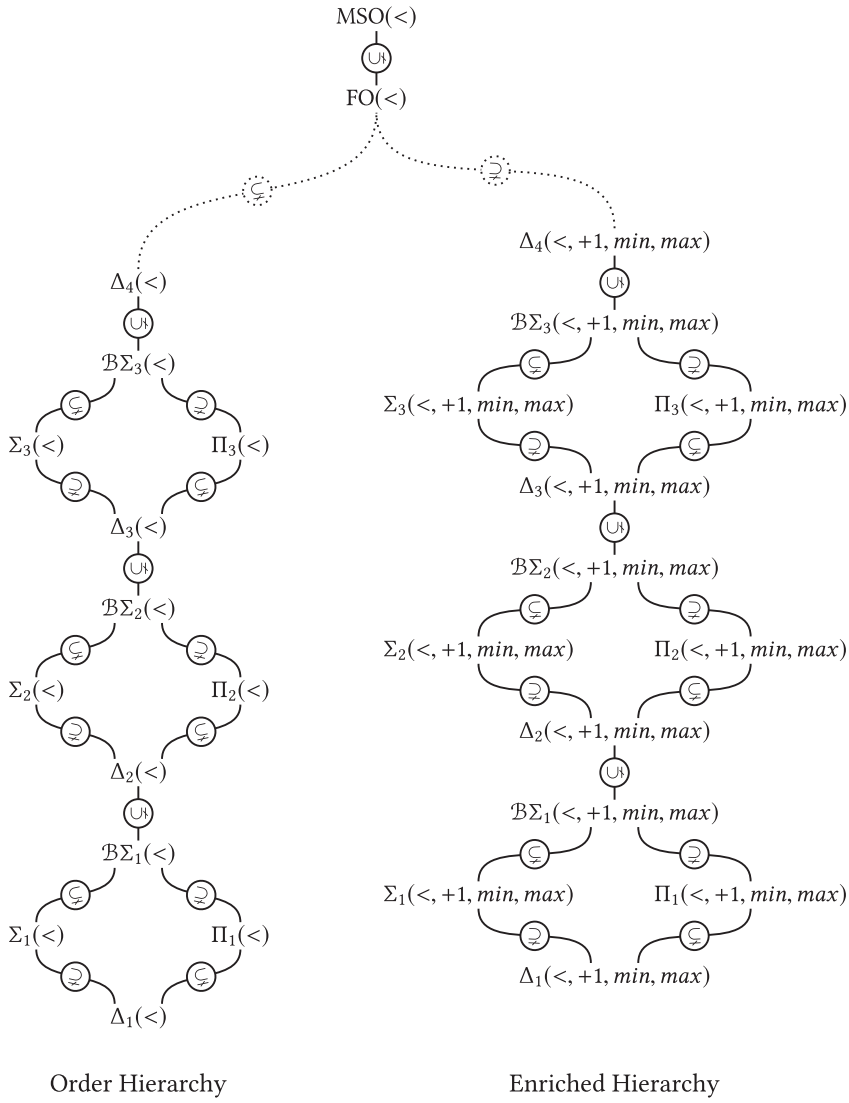


Fig. 1. Quantifier alternation hierarchies.

It turns out that quantifier alternation can be used to define another natural hierarchy within first-order logic, which we now describe.

Enriched Hierarchy. Observe that in full first-order logic, several natural relations can be defined using the linear order:

—Position x is the first one: $min(x) \stackrel{\text{def}}{=} \forall y\, \neg(y < x)$.
—Position x is the last one: $max(x) \stackrel{\text{def}}{=} \forall y\, \neg(x < y)$.
—Position y is the successor of position x: $(y = x + 1) \stackrel{\text{def}}{=} x < y \wedge \neg(\exists z\, x < z \wedge z < y)$.

Therefore, adding these relations as predicates in the signature of first-order logic does not increase its expressive power: FO(<) and the enriched logic FO(<, +1, *min*, *max*) define the same class of

languages. However, observe that replacing the predicates min, max or $+1$ with their definitions may increase the quantifier alternation of the formula. For example,

$$\exists x \exists y \quad y = x + 1 \wedge P_a(x) \wedge P_b(y) \qquad \text{has no alternation, while}$$
$$\exists x \exists y \quad (x < y \wedge \neg(\exists z\ x < z \wedge z < y)) \wedge P_a(x) \wedge P_b(y) \qquad \text{has one alternation.}$$

Hence, it is not immediate whether fragments of the order hierarchy have the same expressive power as their enriched counterpart. In fact, it is known that the predicate $+1$ cannot be freely defined in any logic of the order hierarchy. Hence, we get a second hierarchy, also depicted in Figure 1. That this hierarchy is also strict follows from the work of Brzozowski and Knast [19] and Thomas [113, 115]. In this article, we call it the *enriched hierarchy*.

2.2 The Membership and Separation Problems

We now present the two decision problems investigated in this article, called *membership* and *separation*. Both problems can be defined for any class of languages and, therefore, in particular for any class corresponding to a level in either the order or the enriched hierarchy.

The Membership Problem. Fix a class of languages $\mathcal{C}$. The *membership problem* for $\mathcal{C}$ is as follows:

INPUT: A regular language L.
OUTPUT: Does L belong to $\mathcal{C}$?

Usually, an algorithm solving the membership problem for $\mathcal{C}$ is called a *decidable characterization* of $\mathcal{C}$. Note that, in general, there is no guarantee that there exists such an algorithm. In fact, one can actually build classes of regular languages having an undecidable membership problem from decidable ones using standard operators [1, 10, 88]. However, such classes are usually *ad hoc*; we have yet to find a natural class of regular languages having an undecidable membership problem.

Decidable characterizations are known for FO(<) [54, 91] and up to the Σ_2 level in both hierarchies [8, 32, 33, 45, 71, 76, 94] (see Section 3 for more details). In this article, we expand this knowledge and prove decidable characterizations for the levels $\mathcal{B}\Sigma_2$, Δ_3, Σ_3, and Π_3 in both hierarchies. These new results rely on the investigation of a deeper problem that we now define: the separation problem.

The Separation Problem. Let L, L_0, L_1 be languages. We say that L *separates* L_0 from L_1 if

$$L_0 \subseteq L \text{ and } L_1 \cap L = \emptyset.$$

For a class $\mathcal{C}$ of languages, we say that L_0 is $\mathcal{C}$*-separable* from L_1 if some language in $\mathcal{C}$ separates L_0 from L_1. Note that when $\mathcal{C}$ is closed under complement, then $L \in \mathcal{C}$ separates L_0 from L_1 if and only if $A^* \setminus L$ (which also belongs to $\mathcal{C}$) separates L_1 from L_0. Observe, however, that when $\mathcal{C}$ is not closed under complement (e.g., when $\mathcal{C} = \Sigma_i$ or $\mathcal{C} = \Pi_i$), the definition is not symmetrical: it may be the case that L_0 is $\mathcal{C}$-separable from L_1, while L_1 is not $\mathcal{C}$-separable from L_0. The separation problem for $\mathcal{C}$ is as follows:

INPUT: Two regular languages L_0 and L_1.
OUTPUT: Is L_0 $\mathcal{C}$-separable from L_1?

The separation problem is a refinement of the membership problem. Observe that asking whether a language L is $\mathcal{C}$-separable from its complement is equivalent to asking whether $L \in \mathcal{C}$, since

the only potential separator is L itself. Hence, since regular languages are effectively closed under complement, membership immediately reduces to separation.

The separation problem is known to be decidable for full FO(<) [35, 37] thanks to a result of Almeida [4], who proved that the problems solved in these papers are equivalent to separation. A direct proof for FO(<) has been obtained recently by the authors [82, 85]. Separation is also known decidable up to Δ_2 in both hierarchies [28, 80, 83]. In this article, we present a solution for Σ_2 and Π_2 in both hierarchies.

Note that while we obtain results for both the order and enriched hierarchies, we mostly work with the order hierarchy. For the enriched hierarchy, it is known that for each level, both the membership [75, 102] and the separation problem [83, 96] can be reduced to the same problem for the level's counterpart in the order hierarchy. We present these reductions in Section 12. In other sections, we work with the order hierarchy only.

3 HISTORY

We presented in Section 2.1 two hierarchies within first-order logic, defined in purely logical terms. Historically, the very same hierarchies were first considered in a language theoretic framework and were given combinatorial definitions. In this section, we review the history related to these hierarchies, starting with this language theoretic point of view.

We face a compromise between two natural approaches. The first approach would be to present results in a purely chronological order at the risk of getting bogged down in details and thereby missing central threads. The second would be to simply highlight major ideas that have emerged throughout the years at the cost of possibly losing the time line. For the sake of readability, we choose a hybrid approach: we shall review the main trends and milestones, but we adopt a chronological view for each of them.

We organize this section as follows. In Section 3.1, we motivate why such hierarchies have been considered and present their combinatorial definitions. In Section 3.2, we connect the combinatorial and logical definitions. In Section 3.3, we focus on developments that lead to solutions of membership problems for fragments of these hierarchies. We shall explain along the way how research on these hierarchies actually influenced a wide scientific domain.

The literature on these hierarchies is abundant. In this article, we focus only on some specific aspects. For more details and a complete bibliography, refer to the papers surveying the subject, e.g., by Brzozowski [17], Eilenberg [30], Weil [121], Thomas [116], and Pin [60, 63, 64, 66, 68, 84] and to the literature cited in these papers.

3.1 From Schützenberger's Theorem to Concatenation Hierarchies

We first introduce two concatenation hierarchies defined in combinatorial terms with the motivation of classifying regular languages. Note that we recall their definition only in this section. In subsequent sections, we shall present connections between these hierarchies and logical ones and focus on tools that were developed to investigate them.

The definitions of these hierarchies have their source in Schützenberger's Theorem [91], which provides an algorithm to decide whether a regular language is star-free. Recall that star-free languages are built from singleton languages using a finite number of times.

— *Concatenation products*: If K and L are star-free, then so is $KL = \{xy \mid x \in K,\ y \in L\}$.
— *Boolean combinations*: Any finite Boolean combination of star-free languages is star-free.

Schützenberger's Theorem [91] states that a language is star-free if and only if its syntactic monoid is *aperiodic*. The key point is that aperiodicity of a finite monoid is a decidable property. All proofs of this result, either close to the original one [25, 39, 50, 57, 69, 77] or using alternate ideas,

Table 1. Some Variations in the Definition of the Dot-Depth Hierarchy

	Domain	Level 0	Closure from level $i \in \mathbb{N}$ to level $i + \frac{1}{2}$ Union and:
Brzozowski and Cohen [18]	A^*	Finite or co-finite	$K, L \mapsto KL$
Thomas [113]	A^+	Bool$\{uA^*v \mid u, v \in A^*\}$	$K, L \mapsto KL$
Pin and Weil [76]	A^+	$\{\emptyset, A^+\}$	$K, L \mapsto uKvLw$, $u, v, w \in A^*$
Pin [66]	A^*	Bool$\{uA^*v \mid u, v \in A^*\}$	$K, L \mapsto KaL,\ a \in A$

such as [30, 55] or [29, 123], build a star-free expression from an aperiodic language. However, as explained in the introduction, this expression may be unnecessarily complicated. In particular, it may involve avoidable interleavings between the complement and concatenation operations.

The Dot-Depth Hierarchy. The question addressed by Brzozowski and Cohen [18] when they defined the dot-depth hierarchy was to classify star-free languages according to this complexity: the level assigned to a language is the minimal nesting between complement and concatenation that is necessary to express it with a star-free expression (hence, the name: "dot" means "concatenation"). Its definition is motivated by understanding the interplay between Boolean operations and one of the fundamental operations involved in the definition of rationality, namely, the concatenation product of languages, as defined above.

Defining the hierarchy amounts to (1) defining a base level, numbered 0, consisting of "simple" languages, and (2) defining how to build level $i + 1$ from level i for each natural integer i. This step can be decomposed in two substeps:

—Level $i + \frac{1}{2}$ is the closure of level i under finite unions and (possibly marked) products.
—Level i is the closure of level $i + \frac{1}{2}$ under finite Boolean combinations.

Levels of the form $i + \frac{1}{2}$ for an integer i are called *half levels*. They were missing in the original definition but introduced later by Perrin and Pin [58]. There are actually several variations of the dot-depth hierarchy in the literature (see Table 1). These variants consist of choosing the interpretation domain (A^+ or A^*), or the base level, or the precise way to go from an integer level to the next half level. To define half levels, several closure operators have been considered in addition to finite unions, such as closure under usual product of languages or marked product instead of product, defined as follows for $a \in A$:

$$KaL = \{xay \mid x \in K,\ y \in L\}.$$

These minor adjustments were motivated by the needs of each paper. For instance, the definition of Thomas [113] is convenient to establish a correspondence between this hierarchy and the enriched hierarchy at all levels, including level 0 (whose logical definition also differs slightly from ours). Likewise, Pin and Weil [76] consider only languages of nonempty words to elegantly formulate a correspondence with algebraic classes. It is easy to get lost in all of these variations, but what the reader should remember is that these changes are harmless: the definitions coincide on all levels, except possibly on level 0 (with the restriction that levels of hierarchies over A^+ consist of traces over A^+ of languages belonging to hierarchies over A^*). In particular, for each level, all variants have the same decidability status with respect to the problems that we consider.

The Polynomial Closure. Historically, the most investigated concatenation operator is the marked product used, e.g., in the definition of Pin [66] of the dot-depth hierarchy (last line of Table 1). The operation that associates to a class of languages its closure under finite unions and marked products is called *polynomial closure* [91]. It is the common operation employed for going from level i to level $i + \frac{1}{2}$ in the dot-depth and in another hierarchy that we now present. In other words, this new hierarchy differs from the dot-depth only by the choice of base level.

The Straubing-Thérien Hierarchy. Ten years after the dot-depth was defined, Straubing [101, 102] and Thérien [109] independently considered a similar and also natural hierarchy. As before, its definition is by induction: one starts from a simple class of languages (which is level 0 of this hierarchy), and one then applies, alternately, two closure operations to obtain larger classes of languages:

—The class of languages of level 0 is $\{\emptyset, A^*\}$.
—For any integer $i \geqslant 0$, level $i + \frac{1}{2}$ is the polynomial closure of level i.
—Languages of level $i + 1$ are the finite Boolean combinations of languages of level $i + \frac{1}{2}$.

Comparing the definition of the hierarchies (last line of Table 1 for the dot-depth) yields inductively that each level in the Straubing-Thérien hierarchy is contained in the corresponding level of the dot-depth hierarchy. The containment is actually strict; this makes it natural to investigate the exact relationship between the two hierarchies. Also, clearly, both hierarchies fully cover all star-free languages.

Strictness of the Hierarchies. The first natural question is whether these definitions actually yield strict (or infinite, this is equivalent in this case) hierarchies or whether they collapse. The dot-depth hierarchy was shown to be strict by Brzozowski and Knast [19] for alphabets of size at least 2 on integer levels: one can show that L_n defined inductively by $L_0 = \varepsilon$ and $L_n = (aL_{n-1}b)^*$ is at level n in the dot-depth hierarchy. Another proof of the fact that the hierarchy is strict based on algebra was given by Straubing [101]. Yet other proofs were presented by Thomas [114, 115], using arguments based on Ehrenfeucht-Fraïssé games. All of these proofs easily imply that the hierarchy is strict on *all* levels, including half levels.

Regarding the Straubing-Thérien hierarchy, strictness was established by Margolis and Pin [52] (see also [110] for a short proof). Strictness actually follows from a more general result of Straubing [102] that connects both hierarchies (see below).

The fact that both hierarchies are strict makes it relevant to investigate the membership problem at each level of each of these hierarchies. Relatively few results are known, but this question motivated a wealth of fruitful ideas. We shall describe progress in this line of research in Section 3.3. First, let us connect the combinatorial definitions with the ones relying on first-order logic, which we presented in Section 2.1.

3.2 Connections with Logic

The interest in the dot-depth and Straubing-Thérien hierarchies increased after relationships were discovered in the 1980s—first by Thomas, then by Perrin and Pin—between them and logical hierarchies. Recall that we defined two alternation hierarchies within first-order logic in Section 2: the order hierarchy, which counts alternations between blocks of existential and universal quantifiers for formulas in the signature $\{<,\ P_a \mid a \in A\}$, and the enriched hierarchy, which counts the same alternations for formulas in the signature $\{<,\ +1,\ \mathit{min},\ \mathit{max},\ P_a \mid a \in A\}$.

Recall also that Schützenberger [91] proved that star-free languages are exactly first-order definable ones. Thomas [113] discovered a more precise correspondence, level by level, between

the dot-depth hierarchy of star-free languages and the enriched quantifier alternation hierarchy within FO. Note that Thomas [113] did not actually state the result for half levels, as they were not considered. However, it can be easily derived from the arguments of the paper.

Theorem 3.1 (Thomas [113]). *Let $i \geqslant 0$. Then,*

—*A language has dot-depth i if and only if it is definable in $\mathcal{B}\Sigma_i(<, +1, min, max)$.*
—*A language has dot-depth $i + \frac{1}{2}$ if and only if it is definable in $\Sigma_{i+1}(<, +1, min, max)$.*

This connection with finite model theory and descriptive complexity sustained an earlier informal statement by Brzozowski [17] arguing that dot-depth is a relevant complexity parameter. The argument was based on the fact that star-free expressions can express feedback-free circuits, and that concatenation increases the depth of such circuits: *since concatenation (or "dot" operator) is linked to the sequential rather than the combinational nature of a language, the number of concatenation levels required to express a given aperiodic language should provide a useful measure of complexity.* Since it was known that the nonelementary complexity of standard problems for FO is tied to quantifier alternation [100], Theorem 3.1 brought mathematical evidence that the level in the dot-depth hierarchy of a language is indeed a meaningful complexity measure, thus supporting Brzozowski's intuition.

A statement similar to Theorem 3.1 was established by Perrin and Pin [58] for the Straubing-Thérien hierarchy, which corresponds to the order hierarchy.

Theorem 3.2 (Perrin and Pin [58]). *Let $i \geqslant 0$ be an integer. Then,*

—*A language has level i in the Straubing-Thérien hierarchy if and only if it is definable in $\mathcal{B}\Sigma_i(<)$.*
—*A language has level $i + \frac{1}{2}$ in the Straubing-Thérien hierarchy if and only if it is definable in $\Sigma_{i+1}(<)$.*

In addition, Perrin and Pin [58] introduced half levels as the closure under finite unions and intersections of marked products of the preceding level (it turns out that the intersection is actually useless; see [9, 67]). Finally, they extended the correspondence to infinite words.

The results obtained during the 1970s and the 1980s fostered many connections among several communities of researchers, working in automata theory, semigroup theory, or finite model theory, and laid the ground of a clean framework, with tools from these different fields. The research effort continued in the 1990s, in particular with the developments of algebraic methods to investigate membership problems.

3.3 Connections with Algebra: The Syntactic Approach

Knowing that both hierarchies are strict and that they capture a meaningful complexity measure, the most natural question is whether we can compute the level in each of these hierarchies of an input regular language. This corresponds to solving membership for each level. Even though the membership problem is standard at present, it is only after Schützenberger's work that it was identified as the salient problem to look at. Moreover, Schützenberger [91] also proposed a convenient tool to solve this problem, namely, the syntactic monoid. See [63] for a comprehensive survey on this topic.

Syntactic Monoids: Definition and Seminal Result. The *syntactic congruence* $\sim_L$ of a language L, defined by Schützenberger [90], relates those words that cannot be distinguished by the language when embedded in the same context. Formally,

$$u \sim_L v \iff (\forall x, y \in A^*,\ xuy \in L \Leftrightarrow xvy \in L).$$

The key result of Myhill and Nerode [56] implies that a language is regular if and only if this congruence has a finite index. Hence, in this case, the quotient set $A^*/\sim_L$ is a computable finite monoid, called the *syntactic monoid* of the language. An easy-to-check but important property is that L is a union of $\sim_L$-classes, so that the so-called *syntactic morphism* from A^* to $A^*/\sim_L$ that maps a word to its $\sim_L$-class recognizes L (in the sense that L is a union of $\sim_L$-classes; thus, it is the preimage of a subset of the syntactic monoid under the syntactic morphism).

Schützenberger's Theorem precisely states that a language is star-free if and only if it is *aperiodic*, i.e., its syntactic monoid satisfies the equation

$$x^\omega = x^{\omega+1},$$

where ω represents some large integer, which can be computed from the language as well. This means that for every element x of the syntactic monoid of the language, the equality $x^\omega = x^{\omega+1}$ has to hold. Since the syntactic monoid of the input language is finite and computable from any representation of the language, checking whether it is aperiodic is a *decidable* property. To sum up, Schützenberger's Theorem [91] reduces a nontrivial semantic property (to be definable in some fragment for a language) into a purely syntactic, easily testable condition (to satisfy an equation for a finite, computable algebra).

The importance of this result stems from two reasons:

—First, Schützenberger established membership as the standard problem that is worth investigating in order to understand a class of regular languages. This is justified, since obtaining a decidable characterization requires a deep insight about the class, as this amounts to capturing in a *single* algorithm *all* properties that can be expressed within the class.

—Schützenberger also proposed a methodology that proved successful in solving other membership problems. Let us briefly explain the core of his strategy. The hardest direction is to build a star-free expression for a language whose syntactic monoid is aperiodic. The key observation is that either *all* languages recognized by a syntactic monoid are star-free or *none* of them is. Hence, instead of building a star-free expression for a single language, one may rather do so for all languages recognized by its syntactic morphism. The payoff of this approach may not be immediate, as the goal is more demanding than the original one. Yet, the languages recognized by the syntactic monoid are connected to one another, which makes the method amenable to induction as soon as one can decompose each language into simpler ones using only star-free operations.

Despite the current acknowledgment of the impact of Schützenberger's methodology, about 10 more years were necessary to cement it as a fundamental approach.

Validation of the Syntactic Approach. Notable breakthroughs after Schützenberger's Theorem were obtained by Simon, a student of Brzozowski, in his PhD dissertation [93] shortly after the dot-depth hierarchy was defined. His results had a major impact on research in the theoretical computer science community; two results are central in the context of this article. They both characterize important subclasses of level 1 [20, 21, 94]:

a) The class of locally testable languages, i.e., such that membership of a word in such a language is determined only by looking at infixes, prefixes, infixes up to a given length. This result was also obtained independently by McNaughton [53]. It is easy to check that these languages form a subclass of dot-depth 1.

b) The class of piecewise testable languages, i.e., such that membership of a word in such a language is determined only by looking at its scattered subwords up to a given length. It is the Boolean algebra generated by languages of the form $A^*a_1A^* \cdots A^*a_nA^*$. This is exactly

the first level of the Straubing-Thérien hierarchy, corresponding to the class $\mathcal{B}\Sigma_1(<)$. Note, however, that this hierarchy was not already defined at that time.

Before presenting other results about the hierarchies, let us comment on these results and explain why they deeply influenced the theoretical computer science landscape.

—The main reason why Simon's results were recognized as important is that they supported Schützenberger's methodology as the "right" one to tackle membership questions, by underlining the key role played by the syntactic monoid in automata theory. Schützenberger and Simon both used the same strategy in order to obtain their decidable characterizations by reducing membership to checking equations on the syntactic monoid. This common approach was further validated by Eilenberg [30], who established a one-to-one correspondence between varieties of regular languages and varieties of finite monoids. It was complemented by a theorem of Reiterman [87] (see also [11]), which shows that these algebraic classes can be described by a (possibly infinite) set of equations (such as $x^\omega = x^{\omega+1}$ for aperiodic monoids, which characterize star-free languages). Note, however, that Eilenberg's and Reiterman's theorems are generic results, useless for actual characterizations. They do *not* provide a uniform solution to all membership problems: the actual decidable characterization depends, of course, on the class under investigation (the topic of this article is precisely to establish such characterizations for levels in the hierarchies). Yet, Eilenberg's and Reiterman's theorems entail that every class of regular languages that forms a variety can be characterized by equations satisfied by all syntactic monoids of languages of the class. All integer levels in the hierarchies are indeed varieties. This can be verified directly from the definition, and follows also from the original ones of the hierarchies, e.g., [101, 109] (see also [71, 76]). For this reason, a major research direction was to understand the variety of finite monoids associated to them. In this article, we provide equations for $\mathcal{B}\Sigma_2(<)$.

—A second reason why the study on the hierarchies in general and Simon's results in particular were recognized as important is that they connected several areas: automata theory and finite semigroup theory but also combinatorics on words (see [89] or, more recently, [42]) and finite model theory. They received a number of proofs, either reminiscent of the original ones [40, 48, 59, 69], or using arguments of different flavors. For instance, just for the case of piecewise testable languages, Straubing and Thérien [106] gave an alternate proof based on an early use of ordered monoids in automata theory. Ordered monoids turned out to be a key notion in the study of the hierarchies (see below), and the result was reproved by Henckell and Pin [36]. Almeida [2, 3] presented a proof based on profinite topology, Higgins [38] a proof using representations by transformation semigroups, and Klíma [43] a purely algebraic one. Simon's decision criteria for both classes were refined to understand the computational hardness of the associated membership problems [23, 98] and to improve the complexity of original algorithms [44, 62, 65, 97, 118, 119].

—Last, Simon's work contains ingredients that inspired several researchers to solve other membership problems. For instance, the result on locally testable languages [21] introduces the notion of graph congruence, reused by Knast [46] to give a membership algorithm for level 1 of the dot-depth hierarchy. It should be noted that McNaughton's independent proof of the same result [53] has been equally influential. Simon's result also motivated the quest for a characterization of the class of locally threshold-testable languages, where membership of a word depends not only on the set of infixes but also on the number of such infixes counted up to a threshold. This class was characterized by Beauquier and Pin [12, 13] by relying on a deep paper of Thérien and Weiss [111] that reused graph congruences (a completely different proof by Bojańczyk [14] relies on the decidability of Presburger logic

and Parikh's theorem). Graph congruences, in turn, are the premises of the framework developed by Tilson [117] based on the work of Straubing [102] and motivated by difficult decision problems in semigroup theory (in particular, the decidability of the well-known Krohn-Rhodes hierarchy, which classifies languages according to serial decomposition).

Following Simon's results, level 1 in the dot-depth hierarchy was successfully characterized by Knast. The proof is, however, much more technical. To sum up,

- Simon [94] characterized level 1 in Straubing-Thérien hierarchy or, equivalently, the fragment $\mathcal{B}\Sigma_1(<)$ of first-order logic.
- Knast [45, 46] characterized level 1 in the dot-depth hierarchy or, equivalently, the fragment $\mathcal{B}\Sigma_1(<, +1, min, max)$ of first-order logic.

While these results and others for classes outside the hierarchies gather evidence that the syntactic approach is relevant to tackle membership problems, the time intervals between significant contributions regarding levels in the hierarchy show that the problem is difficult. Despite a wealth of results toward a solution for level 2, the last complete statement until the conference version of the present article [81] regarding integer levels went back to Knast [45, 46].

Connections between the Hierarchies. It should be noted that after Knast's result, researchers became primarily focused on the Straubing-Thérien hierarchy rather than on the dot-depth one. This is due to an important result of Straubing [102], who proved that it is the most fundamental of the two from the membership point of view. More precisely, for any integer level $i \geqslant 2$, membership for level i in the dot-depth hierarchy can be effectively reduced to membership for the same level in the Straubing-Thérien hierarchy. This was generalized to half levels by Pin and Weil [75]. This explains why we also work with the Straubing-Thérien hierarchy in this article. We shall detail the actual reductions in Section 12.

Limits of the Syntactic Approach. In view of Straubing's result, the principal objective of researchers became to solve membership for level 2 in the Straubing-Thérien hierarchy (at that time, half levels were not already defined). While a lot of effort was devoted to solving this problem, this proved very difficult. Over the years, several attempts were made:

- First, partial results were obtained by restricting the set of possible input languages. For example, level 2 was characterized by Straubing [104] for languages over an alphabet of size 2. Other partial results were obtained by Cowan [27], building on results of Weil [122] and Straubing and Weil [107].
- Second, many upper bounds of the actual level 2 were introduced. Usually defined by a set of equations and having a decidable membership problem, these upper bounds were often presented as conjectures. When such a conjecture was disproved, a new one was proposed to tighten the gap between the proposed candidate and the actual level 2. For instance, Straubing [103, 104] proposed such a candidate and proved with Weil that it holds in some particular cases [107]. Another version was proposed by Pin and Weil [72], and refined by them in [74]. More recently, Almeida and Klíma [5] disproved the conjecture of Straubing and proposed a new candidate [6]. All of these conjectures actually provided strict upper bounds for level 2.
- A third approach was to reduce the decidability of level 2 to distinct mathematical problems. A remarkable example is the relationship between the decidability of level 2 and a purely algebraic problem. This connection was discovered by Pin and Straubing [70]: they considered the variety generated by all finite monoids of upper triangular Boolean

matrices, and proved that it corresponds exactly to level 2 in the Straubing-Thérien hierarchy. Unfortunately, this problem turned out to be as hard as the original one.

— This last approach led to another purely algebraic problem, also equivalent to deciding level 2 in the dot-depth hierarchy. It asks to decide membership of a finite monoid in the variety of monoids denoted by **PJ**. In a nutshell, if **V** is a variety of finite monoids, the class **PV** stands for the variety of finite monoids generated by powersets of elements of **V** (indeed, the powerset 2^M of any monoid M can be naturally endowed itself with a monoid structure). Such varieties were deeply investigated; see, for example, [51]. The variety of finite monoids **J** is simply the class associated with the variety of piecewise testable languages in Eilenberg's correspondence.

To sum up, solving membership for level two of the Straubing-Thérien and dot-depth hierarchies also yields a solution to several major open problems in three different domains: automata and formal languages, logic and finite model theory, and finite semigroup theory.

Half Levels and Ordered Monoids. All of these attempts underlined that level 2 was difficult to attack directly. This motivated the investigation of the half levels, introduced by Perrin and Pin [58]. At first glance, they may seem to be just an additional refinement, but this is not the case. First, half levels are arguably more fundamental than integer levels, since each integer level can be reconstructed from the preceding half level by closure under Boolean operations. Also, importantly, half levels are simpler to deal with, and understanding them is a first step toward membership algorithms for integer levels. For instance, gathering enough information about level $\frac{3}{2}$ is crucial in our approach to the solution of the membership problem at level 2.

The main issue with half levels is that they are not closed under complement. This is a problem for generalizing Schützenberger's methodology, which translates the semantic membership problem into a property of the syntactic monoid. The reason why the syntactic approach works for "varieties" is that, for such a class C, either all or none of the languages recognized by a syntactic monoid belong to C. This is precisely what fails for half levels, since a language is recognized by a monoid if and only if its complement is recognized as well. In other words, the syntactic monoid is not well suited to capture classes that are not closed under complement and, therefore, has to be adapted if one wants to generalize Eilenberg's Theorem.

Nonetheless, Arfi [8, 9] managed to show that levels $\frac{1}{2}$ and $\frac{3}{2}$ of the Straubing-Thérien hierarchy have decidable membership and to describe the associated classes of languages. This is very easy for level $\frac{1}{2}$. A downside of this approach for level $\frac{3}{2}$ is that it relied on involved results of Hashiguchi [34], thus hiding the core of the argument. Note also that Straubing's transfer result did not apply to half levels, since it relied on the correspondence between varieties of languages and varieties of finite semigroups. This made it relevant to investigate level $\frac{3}{2}$ in the dot-depth hierarchy as well, a task successfully achieved by Glaßer and Schmitz [32, 33]. However, this combinatorial and technical proof is not easily amenable to generalization.

This made it crucial to understand what could be saved from Schützenberger's approach. In fact, Arfi's characterization for level $\frac{1}{2}$ is explicitly stated as a property to be satisfied by the syntactic monoid. This property is not an equation but rather is a closure property of the accepting set of the language. This led Pin [61] to develop an Eilenberg-Schützenberger methodology for classes that are not closed under complement. Pin's idea was to equip monoids with a partial order relation compatible with multiplication and to constrain accepting sets to be upward closed. This yields an adapted notion of recognizability, for which the set of languages recognized by an ordered syntactic monoid is no longer closed under complement but still carries enough structure and information to recover the generic methodology [61], including equational descriptions of such classes [73]. Cleaner decidability membership algorithms were subsequently re-obtained for level

$\frac{3}{2}$ [15, 71, 76]. Instead of Hashiguchi's black box result, the proofs in these papers rely on a simple tool that we shall also use: the factorization forest Theorem of Simon [95]. Finally, Straubing's results connecting the Straubing-Thérien and the dot-depth hierarchies were also generalized to this new setting [75], thus giving an alternate proof of the decidability of level $\frac{3}{2}$ in the dot-depth hierarchy.

Level 2 and Above. This article continues this research effort. As explained in the introduction, a key ingredient in our approach is to consider the *separation problem*, which is more demanding than membership. The core of our results is a solution to this problem for $\Sigma_2(<)$, i.e., the level $\frac{3}{2}$ in the Straubing-Thérien hierarchy (for which membership is already known to be decidable). We are then able to obtain membership algorithms for both $\Sigma_3(<)$ and $\mathcal{B}\Sigma_2(<)$ by building upon this first separation algorithm. This highlights the fact that a solution for the separation problem associated to some class carries information that can be exploited to tackle weaker problems (such as membership) for more complicated classes. In particular, a good illustration of this is the fact that our membership algorithm for $\Sigma_3(<)$ follows from a generic connection between separation and membership: for *any integer* i, a separation algorithm for $\Sigma_i(<)$ yields a membership algorithm for $\Sigma_{i+1}(<)$. On the other hand, our membership algorithm for $\mathcal{B}\Sigma_2(<)$ results from a specific and detailed analysis of the separation algorithm for level $\Sigma_2(<)$.

Finally, note that while we work with the order hierarchy (i.e., the Straubing-Thérien hierarchy) in most sections, we come back to the enriched hierarchy (i.e., the dot-depth hierarchy) at the end of the article. Using previously known transfer theorems, we are able to lift all results that we have proved for levels in the order hierarchy to the same levels in the enriched hierarchy.

4 TOOLS

In this section, we recall the definitions of two well-known combinatorial tools used several times in the article:

- —The *Ehrenfeucht-Fraïssé game* variant corresponding to levels of the order hierarchy, which are a means to capture their expressive power in terms of games. For more on Ehrenfeucht-Fraïssé games, see [41, 49, 105].
- —The definition of regular languages in terms of *monoids*. This definition makes it possible to use convenient combinatorial results, in particular, Simon's Factorization Forests Theorem, which we also present in this section.

4.1 Logical Tools: Ehrenfeucht-Fraïssé Games

It is standard practice to classify first-order formulas according to their *quantifier rank*, i.e., the length of the longest sequence of nested quantifiers in the formula. For example, the formula

$$\forall x\, P_a(x) \Rightarrow ((\exists y\, (y < x \wedge P_c(y)) \wedge (\exists y \exists z\, (x < y < z) \wedge P_b(y)))$$

has quantifier rank 3.

Note that the notion of quantifier *rank* is different from the notion of quantifier *alternation*. In particular, one may verify that there is a finite number of nonequivalent first-order sentences of any fixed rank, while there are infinitely many nonequivalent sentences of quantifier alternation 0.

We use the quantifier rank to associate to our logics binary relations over the set A^*. We begin with the logics $\Sigma_i(<)$. Let $k, i \in \mathbb{N}$ and $w, w' \in A^*$. We write

$$w \lesssim_i^k w'$$

if *every* $\Sigma_i(<)$ formula of quantifier rank k satisfied by w is also satisfied by w'. Observe that since a $\Pi_i(<)$ formula is the negation of a $\Sigma_i(<)$ formula, we have that $w \lesssim_i^k w'$ if and only if every $\Pi_i(<)$ formula of quantifier rank k satisfied by w' is also satisfied by w. Moreover, the following facts are immediate from the definition.

Fact 4.1. *For all $k, i \in \mathbb{N}$, $\lesssim_i^k$ is a preorder.*

Fact 4.2. *For all $k, i \in \mathbb{N}$, a language $L \subseteq A^*$ can be defined by a $\Sigma_i(<)$ formula of rank k if and only if L is saturated by $\lesssim_i^k$, i.e., if and only if*

$$L = \{w \mid \text{there exists } w' \in L \text{ such that } w' \lesssim_i^k w\}.$$

We now extend this definition to the logics $\mathcal{B}\Sigma_i(<)$. Let $k, i \in \mathbb{N}$ and $w, w' \in A^*$. We write $w \cong_i^k w'$ if w and w' satisfy the same $\mathcal{B}\Sigma_i(<)$ formulas of quantifier rank k. By definition, $\mathcal{B}\Sigma_i(<)$ formulas are finite Boolean combinations of $\Sigma_i(<)$ formulas. We thus obtain the following two facts:

Fact 4.3. *For all $k, i \in \mathbb{N}$, $\cong_i^k$ is the equivalence relation induced by $\lesssim_i^k$, i.e.,*

$$w \cong_i^k w' \text{ if and only if } w \lesssim_i^k w' \text{ and } w' \lesssim_i^k w.$$

Moreover, for all fixed $k, i \in \mathbb{N}$, $\cong_i^k$ has finite index.

Fact 4.4. *For all $k, i \in \mathbb{N}$, a language $L \subseteq A^*$ can be defined by a $\mathcal{B}\Sigma_i(<)$ formula of rank k if and only if L is a union of equivalence classes of $\cong_i^k$, that is, if and only if*

$$L = \{w \mid \text{there exists } w' \in L \text{ such that } w' \cong_i^k w\}.$$

We can now define Ehrenfeucht-Fraïssé games. A specific Ehrenfeucht-Fraïssé game can be associated to every logic. Here, we define the game tailored to the logics $\Sigma_i(<)$ in the quantifier alternation hierarchy. This means that these games characterize the preorders $\lesssim_i^k$ (and, therefore, by Fact 4.3, also the equivalence $\cong_i^k$).

Ehrenfeucht-Fraïssé Games. Before giving the definition, a remark is in order. There are actually two ways to define the class of $\Sigma_i(<)$-definable languages. First, one can consider *all* first-order formulas and say that a formula is $\Sigma_i(<)$ if it has at most i blocks of quantifiers once rewritten in prenex normal form. However, one could also restrict the set of allowed formulas to only those that are already in prenex form and have at most i blocks of quantifiers. While this does not change the class of $\Sigma_i(<)$-definable languages as a whole, this changes the set of formulas of quantifier rank k for a fixed k. Therefore, this changes the preorder $\lesssim_i^k$. This means that there is a version of the Ehrenfeucht-Fraïssé game for each definition. In this article, we use the version corresponding to the definition that considers *all* first-order formulas.

Let $i \geqslant 1$. We define the game associated to $\Sigma_i(<)$. The board of the game consists of two words $w, w' \in A^*$ and there are two players, called *Spoiler* and *Duplicator*. Moreover, initially, there exists a distinguished word among w, w' that we call the *active word* (this word may change as the play progresses). The game is set to last a predefined number k of rounds. When the play starts, both players have k pebbles. Finally, there is a parameter that gets updated during the game, a counter c called the *alternation counter*. Initially, c is set to 0 and has to be bounded by $i - 1$.

At the start of each round j, Spoiler chooses a word, either w or w'. Spoiler can always choose the active word, in which case both c and the active word remain unchanged. However, Spoiler can only choose the word that is not active when $c < i - 1$, in which case the active word is switched and c is incremented by 1 (this means that the active word can be switched at most $i - 1$ times). If Spoiler chooses w (resp., w'), Spoiler puts a pebble on a position x_j in w (resp., x'_j in w').

Duplicator must answer by putting a pebble at a position x'_j in w' (resp., x_j in w). Moreover, Duplicator must ensure that all pebbles that have been placed up to this point satisfy the following condition: for all $\ell_1, \ell_2 \leqslant j$, the labels at positions x_{ℓ_1}, x'_{ℓ_1} are the same, and $x_{\ell_1} < x_{\ell_2}$ if and only if $x'_{\ell_1} < x'_{\ell_2}$.

Duplicator wins if Duplicator manages to play for all k rounds, and Spoiler wins as soon as Duplicator is unable to play.

Lemma 4.5 (Game definition of $\lesssim_i^k$ (Ehrenfeucht-Fraïssé)). *For all $k, i \in \mathbb{N}$ and $w, w' \in A^*$, $w \lesssim_i^k w'$ if and only if Duplicator has a winning strategy for playing k rounds in the $\Sigma_i(<)$ game played over w, w' with w as the initial active word.*

Note that we will often use Lemma 4.5 implicitly and alternate between the original and the game definition of $\lesssim_i^k$. We now present a few classical lemmas on Ehrenfeucht-Fraïssé games that we reuse several times in our proofs. We begin with a lemma stating that $\lesssim_i^k$ is a precongruence, i.e., that it is compatible with the concatenation product.

Lemma 4.6 (Precongruence Lemma). *Let $i \in \mathbb{N}$ and let $w_1, w'_1, w_2, w'_2 \in A^*$. Then,*

$$(w_1 \lesssim_i^k w'_1 \text{ and } w_2 \lesssim_i^k w'_2) \Rightarrow w_1 w_2 \lesssim_i^k w'_1 w'_2.$$

Proof. By Lemma 4.5, Duplicator has winning strategies in the $\Sigma_i(<)$ games over w_1, w'_1 and w_2, w'_2, with w_1, w_2 as initial active words, respectively. These strategies can be easily combined into a strategy for the $\Sigma_i(<)$ game over $w_1 w_2$ and $w'_1 w'_2$, with $w_1 w_2$ as initial active word. We conclude that $w_1 w_2 \lesssim_i^k w'_1 w'_2$. □

The second lemma is a well-known property of full first-order logic, which implies that, unlike monadic second-order logic, first-order logic cannot express modulo counting. This property is called *aperiodicity*.

Lemma 4.7 (Aperiodicity Lemma). *Let $k, k_1, k_2 \in \mathbb{N}$ be such that $k_1, k_2 \geqslant 2^k - 1$. Let $v \in A^*$. Then,*

$$\text{for all } i \in \mathbb{N}, \quad v^{k_1} \lesssim_i^k v^{k_2}.$$

Proof. This is well known for full first-order logic and can be verified by induction on k (see [105, pp. 44–46] for details). □

We finish with another classical property, which we call the Σ_i*-property*. Contrary to the precongruence or aperiodicity properties, the Σ_i-property is specific to $\Sigma_i(<)$. It will be central in the proofs.

Lemma 4.8 (Σ_i-property Lemma). *Let $i \in \mathbb{N}$, and let $k, \ell, r, \ell', r' \in \mathbb{N}$ be such that $\ell, r, \ell', r' \geqslant 2^k$ and let $u, v \in A^*$ such that $v \lesssim_i^k u$. Then, we have: that*

$$u^\ell u^r \lesssim_{i+1}^k u^{\ell'} v u^{r'}.$$

Proof. Let $w = u^\ell u^r$ and $w' = u^{\ell'} v u^{r'}$. We prove that $w \lesssim_{i+1}^k w'$ using an Ehrenfeucht-Fraïssé argument: we prove that Duplicator has a winning strategy for the game in k rounds for $\Sigma_{i+1}(<)$ played on w, w' with w as the initial active word. The proof goes by induction on k. We distinguish two cases depending on the value, 0 or 1, of the alternation counter c after Spoiler has played the first round.

Case 1: $c = 1$. In this case, by definition of the game, it suffices to prove that $w' \lesssim_i^k w$. From our hypothesis, we already know that $v \lesssim_i^k u$. Moreover, it follows from Lemma 4.7 that $u^{\ell'} \lesssim_i^k u^\ell$ and $u^{r'} \lesssim_i^k u^{r-1}$. It then follows from Lemma 4.6 that $w' \lesssim_i^k w$.

Case 2: $c = 0$. By definition, this means that Spoiler has played on some position x in w. Therefore, x is inside a copy of the word u. Since w contains at least 2^{k+1} copies of u, by symmetry, we can assume that there are at least 2^k copies of u to the right of x. We now define a position x' inside w' that will serve as Duplicator's answer. We choose x' so that it belongs to a copy of u inside w' and is at the same relative position inside this copy, as x is in its own copy of u. Therefore, to fully define x', it only remains to define the copy of u in which we choose x'. Let n be the number of copies of u to the *left* of x in w, i.e., x belongs to the $(n+1)$th copy of u starting from the left of w. If $n < 2^{k-1} - 1$, then x' is chosen inside the $(n+1)$th copy of u starting from the left of w'. Otherwise, x' is chosen inside the 2^{k-1}th copy of u starting from the left of w'. Observe that these copies always exist and occur before the factor v, since $\ell' \geqslant 2^k$.

Let $w = w_p u w_q$ and $w' = w'_p u w'_q$, where the two distinguished u factors are the copies containing positions x, x'. By definition of the game, it suffices to prove that $w_p \lesssim_{i+1}^{k-1} w'_p$ and $w_q \lesssim_{i+1}^{k-1} w'_q$ to conclude that Duplicator can play for the remaining $k-1$ rounds. If $n < 2^{k-1} - 1$, then, by definition, $w_p = w'_p$; therefore, it is immediate that $w_p \lesssim_{i+1}^{k-1} w'_p$. Otherwise, both w_p and w'_p are concatenations of at least $2^{k-1} - 1$ copies of u. Therefore, $w_p \lesssim_{i+1}^{k-1} w'_p$ follows Lemma 4.7. Finally, observe that by definition, w_q and w'_q are of the form $w_q = u^{\ell_1} u^r$ and $w'_q = u^{\ell'_1} v u^{r'}$ for some ℓ_1 and ℓ'_1 such that $\ell_1 + r \geqslant 2^k$ (by the assumption made at the beginning of Case 2) and $\ell'_1, r' \geqslant 2^{k-1}$ (by the choice made by Duplicator and hypothesis on r'). Therefore, it is immediate by induction on k that $w_q \lesssim_{i+1}^{k-1} w'_q$. □

4.2 Algebraic Tools: Monoids and Simon's Factorization Forests Theorem

A *semigroup* is a set S equipped with an associative multiplication $(s,t) \mapsto st$. A *monoid* M is a semigroup in which there exists a neutral element denoted 1_M. Observe that A^* is a monoid with concatenation as the multiplication and ε as the neutral element.

An element e of a semigroup is *idempotent* if $ee = e$. Given any finite semigroup S, it is well known that there is a number $\omega(S)$, denoted by ω when S is understood from the context, such that s^ω is an idempotent for each element s of S: $s^\omega = s^\omega s^\omega$.

Monoids are a standard tool to recognize regular languages. Let L be a language and M be a monoid. We say that L *is recognized by* M if there exists a monoid morphism $\alpha : A^* \to M$ and an *accepting set* $F \subseteq M$ such that $L = \alpha^{-1}(F)$. Kleene's theorem states that a language is regular if and only if it can be recognized by a *finite monoid*.

The usual approach to characterizing a class of regular languages is to abstract it as a class of monoids, each recognizing only languages in the class and such that, conversely, any language is recognized by one of these monoids. For such an approach to work, the class of languages has to fulfill some properties. In particular, since any monoid recognizing a language also recognizes its complement, this approach makes sense only when the class of languages is closed under complement (among other operations).

In this article, however, we investigate classes of languages, such as $\Sigma_i(<)$, that are *not* closed under complement. For such classes, one needs to use *ordered monoids* as recognizing structures. An ordered monoid is a monoid endowed with a partial order $\leqslant$, which is compatible with multiplication: $s \leqslant t$ and $s' \leqslant t'$ imply $ss' \leqslant tt'$.

We say that L is recognized by an ordered monoid M if there exist a monoid morphism $\alpha : A^* \to M$ and an *upward closed* accepting set $F \subseteq M$ such that $L = \alpha^{-1}(F)$. One also says that α recognizes L. The condition for F of being upward closed means that if $s \in F$ and $s \leqslant t$, then also $t \in F$. Note that if α recognizes L, although $A^* \setminus L = \alpha^{-1}(M \setminus F)$, the set $M \setminus F$ is not necessarily upward closed; hence, $A^* \setminus L$ is not necessarily recognized by α. Note that if a language L is recognized by an ordered monoid $(M, \leqslant)$, then the complement of L is recognized by the ordered monoid $(M, \geqslant)$.

Syntactic Ordered Monoid of a Language. Given a regular language L, one may compute a canonical finite ordered monoid that recognizes it as follows. The *syntactic preorder* $\leqslant_L$ of a language L is defined on pairs of words in A^* by $w \leqslant_L w'$ if, for all $u, v \in A^*$, $uwv \in L \Rightarrow uw'v \in L$. Similarly, we define $\equiv_L$, the *syntactic equivalence* of L, as follows: $w \equiv_L w'$ if $w \leqslant_L w'$ and $w' \leqslant_L w$. One can verify that $\leqslant_L$ and $\equiv_L$ are compatible with multiplication. Therefore, the quotient M_L of A^* by $\equiv_L$ is a monoid that we call the *syntactic monoid of* L. The associated morphism is called the *syntactic morphism of* L. Moreover, the partial order on M_L induced by the preorder $\leqslant_L$ makes M_L an ordered monoid. It is called the *syntactic ordered monoid* of L (the notion was first introduced by Schützenberger [90]). One can check that the syntactic ordered monoid can be effectively computed from L. See [61] for details.

Morphisms and Separation. When working on separation, we consider as input two regular languages L_0, L_1. It will be convenient to have a *single* monoid recognizing both of them rather than having to deal with two objects. This can always be assumed without loss of generality, as such a monoid can easily be constructed as follows. Let M_0, M_1 be monoids recognizing L_0, L_1 together with the morphisms α_0, α_1, respectively. Then, $M_0 \times M_1$ equipped with the componentwise multiplication $(s_0, s_1)(t_0, t_1) = (s_0t_0, s_1t_1)$ is a monoid that recognizes both L_0 and L_1 with the morphism $\alpha : w \mapsto (\alpha_0(w), \alpha_1(w))$.

Alphabet-Compatible Morphisms. In our Σ_2-separation algorithm, it will be convenient to work with morphisms that satisfy an additional property. A morphism $\alpha : A^* \to M$ is said to be *alphabet compatible* if, for all $u, v \in A^*$, $\alpha(u) = \alpha(v)$ implies that $\mathsf{alph}(u) = \mathsf{alph}(v)$. Note that when α is alphabet compatible, $\mathsf{alph}(s)$ is well defined for all $s \in M$ as the unique subset B of A such that, for all $u \in \alpha^{-1}(s)$, we have that $\mathsf{alph}(u) = B$ (if s has no preimage, then we simply let $\mathsf{alph}(s) = \emptyset$).

To any morphism $\alpha : A^* \to M$ into a finite monoid M, we associate a morphism β, called the *alphabet completion* of α, that recognizes all languages recognized by α and is alphabet compatible. If α is already alphabet compatible, then $\beta = \alpha$. Otherwise, observe that 2^A is a monoid with union as the multiplication. Hence, we can define β as the morphism:

$$\begin{array}{rccc} \beta : & A^* & \to & M \times 2^A \\ & w & \mapsto & (\alpha(w), \mathsf{alph}(w)). \end{array}$$

It is straightforward to verify that any language recognized by a morphism into a finite (ordered) monoid is also recognized by its alphabet completion.

Simon's Factorization Forests Theorem. In several of our proofs, we make use of a combinatorial result on monoids: Simon's Factorization Forests Theorem [95]. We state this theorem here. For more details on factorization forests and a proof of the theorem, see [15, 24, 26, 47].

Let M be a finite monoid and $\alpha : A^* \to M$ a morphism. An *α-factorization forest* is an ordered unranked tree whose nodes are labeled by words in A^* and such that for any inner node x with label w, if $x_1, \ldots, x_n$ are its children listed from left to right with labels $w_1, \ldots, w_n$, then $w = w_1 \cdots w_n$. Moreover, any node in the forest must be of one of the three following kinds:

- *leaf nodes*, which are labeled by either a single letter or the empty word;
- *binary nodes*, which have exactly two children; or
- *idempotent nodes*, which have an arbitrary number of children whose labels $w_1, \ldots, w_n$ satisfy $\alpha(w_1) = \cdots = \alpha(w_n) = e$ for some idempotent $e \in M$.

If $w \in A^*$, an *α-factorization forest for w* is an α-factorization forest whose root is labeled by w.

Theorem 4.9 ([47, 95]). *For all words $w \in A^*$, there exists an α-factorization forest for w of height at most $3|M| - 1$.*

5 Σ_i-CHAINS AND Σ_i-JUNCTURES

In this section, we introduce our last tool, the set of Σ_i-*chains*. It is specific to the paper and is central to all of our results. Such a set can be associated to any morphism $\alpha : A^* \to M$, and the notion is designed with the separation problem for Σ_i and $\mathcal{B}\Sigma_i$ in mind: both problems can be reduced to the computation of this set.

In this section, we only give the definition of Σ_i-chains. The link with separation and membership is discussed in Section 6. We split the presentation into two parts. In the first part, we define Σ_i-*chains*. In the second part, we define a refined notion: Σ_i-*junctures*. This second notion carries more information than standard Σ_i-chains and is actually more than we need to make the link with separation. However, we will have to work with this stronger notion in order to be able to compute Σ_2-chains in Section 7.

5.1 Σ_i-Chains

Chains. Let M be a finite monoid. A *chain* for M is a word over the alphabet M, i.e., an element of M^*. A remark about notation is in order here. A word is usually denoted as the concatenation of its letters. Since M is a monoid, this would be ambiguous here since st could either mean a word with 2 letters s and t, or the product of s and t in M. To avoid confusion, we will write $(s_1, \ldots, s_n)$ a chain for M of length n. Note that when M is clear from the context, we will simply speak of chains, leaving M implicit.

For all $n \in \mathbb{N}$, observe that M^n, the set of chains of length n, is a monoid when equipped with the component-jwise multiplication. In this article, we denote chains by $\bar{s}, \bar{t}, \ldots$ and sets of chains by $\mathcal{S}, \mathcal{T}, \ldots$. As explained above, given a monoid M, we are not interested in all chains for M but rather only in those that carry information with respect to the logic $\Sigma_i(<)$ and some morphism $\alpha : A^* \to M$, which we call the Σ_i-*chains for* α.

Σ_i-Chains. Fix $i \in \mathbb{N}$, we begin by defining a set of $\Sigma_i[k]$-*chains* for each fixed quantifier rank k. The set of Σ_i-chains will then be the intersection of all sets of $\Sigma_i[k]$-chains.

When $i = 0$, we let by convention $C_i^k[\alpha] = M^*$ for all k. Otherwise, when $i \geqslant 1$, we let

$$(s_1, \ldots, s_n) \in C_i^k[\alpha] \text{ if there exist } w_1, \ldots, w_n \in A^* \text{ with } \begin{cases} w_1 \lesssim_i^k \cdots \lesssim_i^k w_n \text{ and} \\ \forall j,\ \alpha(w_j) = s_j. \end{cases}$$

We can now define the *set of* Σ_i-*chains for* α as the set

$$C_i[\alpha] = \bigcap_k C_i^k[\alpha].$$

Remark. Observe that the set of $\Sigma_i[k]$-chains of length 2 can be viewed as an abstraction of $\lesssim_i^k$ over the set M with respect to α. An important observation is that this abstraction is no longer a preorder: in general, this is a nontransitive relation. This is because (r, s) and (s, t) are $\Sigma_i[k]$-chains of length 2 if and only if there are words u, v mapped to r, s and v', w mapped to s, t, respectively, such that $u \lesssim_i^k v$ and $v' \lesssim_i^k w$. However, v and v' may be completely unrelated. In particular, this means that the whole set of $\Sigma_i[k]$-chains carries more information than the set of $\Sigma_i[k]$-chains of length 2 only.

It will often be convenient to speak only of Σ_i-chains of a given fixed length. For any fixed $n \in \mathbb{N}$, we let $C_{i,n}^k[\alpha]$ be the set of $\Sigma_i[k]$-chains of length n for α, i.e., $C_{i,n}^k[\alpha] = C_i^k[\alpha] \cap M^n$. We define $C_{i,n}[\alpha]$ similarly. We have the following lemma.

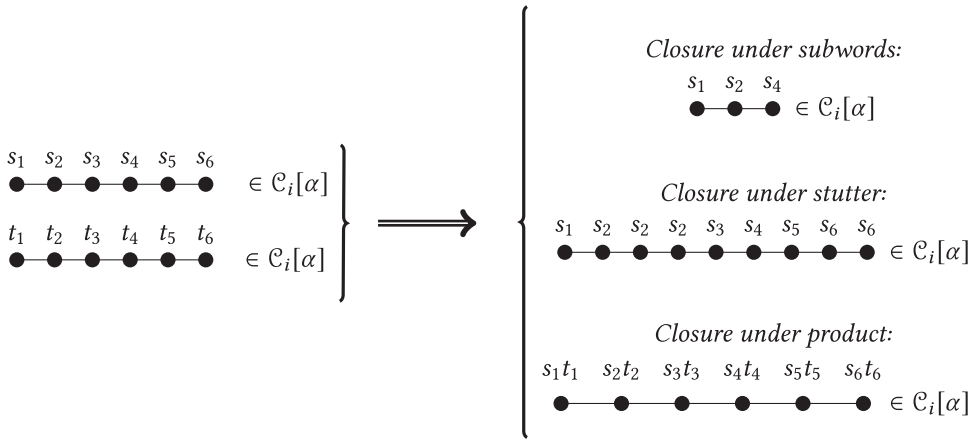


Fig. 2. Closure properties of Σ_i-chains (example of $C_i[\alpha]$).

Lemma 5.1. *For every $i, k, n \in \mathbb{N}$,*

$$\begin{array}{ccccc} C_i[\alpha] & \subseteq & C_i^{k+1}[\alpha] & \subseteq & C_i^k[\alpha]. \\ C_{i,n}[\alpha] & \subseteq & C_{i,n}^{k+1}[\alpha] & \subseteq & C_{i,n}^k[\alpha]. \end{array}$$

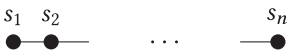
Moreover, for all $i, n \in \mathbb{N}$, there exists $\kappa_{i,n} \in \mathbb{N}$ such that, for every $k \geqslant \kappa_{i,n}$,

$$C_{i,n}[\alpha] \quad = \quad C_{i,n}^{\kappa_{i,n}}[\alpha] \quad = \quad C_{i,n}^k[\alpha].$$

Proof. The first property is immediate from the definitions. The existence of $\kappa_{i,n}$ follows from the first property and the fact that, for all fixed n, M^n is a finite set. □

Note that, for a given k, the set $C_{i,n}^k[\alpha]$ can be computed by brute force by calculating all $\cong_i^k$-classes in A^* (which can be done by enumerating all of the finitely many nonequivalent formulas of rank k in $\Sigma_i(<)$). Therefore, computing (an upper bound on) $\kappa_{i,n}$ immediately yields computability of $C_{i,n}[\alpha]$. However, while the existence of $\kappa_{i,n}$ is easy to prove, its computation is nontrivial. It may happen that $C_{i,n}^k[\alpha] = C_{i,n}^{k+1}[\alpha]$, but $C_{i,n}^{k+1}[\alpha] \supsetneq C_{i,n}^{k+2}[\alpha]$. We will obtain a bound on $\kappa_{i,n}$ as a by-product of our algorithm for computing Σ_i-chains presented in Section 7.

Closure Properties. We finish the definitions by stating simple closure properties of the sets $C_i^k[\alpha]$ and $C_i[\alpha]$: closure under subwords, closure under stutter, and closure under product. These three properties are illustrated in an example in Figure 2, where chains are represented pictorially: we draw the chain $(s_1, s_2, \ldots, s_n)$ as



Observe first that since the relation $\lesssim_i^k$ is transitive for all i, k, the sets $C_i^k[\alpha]$ and $C_i[\alpha]$ are closed under subwords.

Fact 5.2. *Let $i, k \in \mathbb{N}$ and let $\mathcal{X} = C_i[\alpha]$ or $\mathcal{X} = C_i^k[\alpha]$. Then, $\mathcal{X}$ is closed under subwords. That is, for all $(s_1, \ldots, s_n) \in \mathcal{X}$ and all $j \leqslant n$, we have that $(s_1, \ldots, s_{j-1}, s_{j+1}, \ldots, s_n) \in \mathcal{X}$.*

An interesting consequence of Fact 5.2 is that, by Higman's lemma, $C_i[\alpha]$ and $C_i^k[\alpha]$ are both regular languages over the alphabet M. However, this observation is essentially useless in our argument, as Higman's lemma provides no way for actually computing a recognizing device for the language $C_i[\alpha]$.

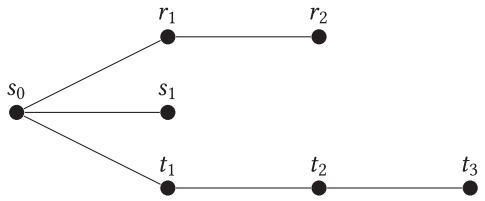


Fig. 3. Juncture $\big(s_0, \{(r_1, r_2), (s_1), (t_1, t_2, t_3)\}\big)$.

Another immediate property of $C_i[\alpha]$ and $C_i^k[\alpha]$ is closure under duplication of letters (also called *stutter*).

Fact 5.3. *Let $i, k \in \mathbb{N}$ and let $\mathcal{X} = C_i[\alpha]$ or $\mathcal{X} = C_i^k[\alpha]$. Then, $\mathcal{X}$ is closed under stutter. That is, for all $(s_1, \ldots, s_n) \in \mathcal{X}$ and all $j \leqslant n$, we have that $(s_1, \ldots, s_j, s_j, \ldots, s_n) \in \mathcal{X}$.*

Finally, since $\lesssim_i^k$ is compatible with the concatenation operation for every k (see Lemma 4.6, the precongruence Lemma), it is immediate that Σ_i-chains of length n are closed under product (*i.e.*, component-wise multiplication).

Fact 5.4. *For all $i, k, n \in \mathbb{N}$, both $C_{i,n}[\alpha]$ and $C_{i,n}^k[\alpha]$ are submonoids of M^n.*

This ends the definition of Σ_i-chains. This leaves two issues.

—First, we need to explain the link between the computation of Σ_i-chains and our decision problems. We establish this link in Section 6. For example, we show that the separation problem for $\Sigma_i(<)$ reduces to the computation of all Σ_i-chains of length 2.
—The second issue is finding an algorithm that, given a morphism α, computes the set of associated Σ_i-chains. We will present such an algorithm for Σ_2-chains in Section 7. However, this algorithm has to work with a refined notion called *"Σ_i-junctures."* We now define this notion.

5.2 Σ_i-Junctures

Our algorithm computes more than we actually need to solve separation. The crucial information is to determine when several Σ_i-chains with the same first element can be "synchronized." To explain what we mean, consider two Σ_i-chains (s, t_1) and (s, t_2) of length 2. By definition, for all k, there exist words w, w_1, w', w_2 whose images under α are s, t_1, s, t_2, respectively, and such that $w \lesssim_i^k w_1$ and $w' \lesssim_i^k w_2$. In some cases (but not all), it will be possible to choose $w = w'$ for all k. The goal of the notion of Σ_i-junctures is to record the cases in which this is true. The reason why we need to capture this extra information is that (1) it can be computed inductively, which is not clear for Σ_i-chains, and (2) it contains more information than Σ_i-chains do.

We first define the generic notion of *juncture*, and then a specific notion, dedicated to our problem, called *Σ_i-juncture*.

Junctures. Let M be a finite monoid. A *juncture* for M is a pair $(s, \mathcal{S})$ where $s \in M$ and $\mathcal{S} \subseteq M^*$ is a set of chains. If $(s, \mathcal{S})$ is a juncture and $\bar{t} = (t_1, \ldots, t_n)$ is a chain, we write $\bar{t} \in (s, \mathcal{S})$ if

$$t_1 = s \quad \textit{and} \quad (t_2, \ldots, t_n) \in \mathcal{S}.$$

Thus, a juncture $(s, \mathcal{S})$ abstracts a set of chains all having the same first element, namely, s. Although we will not use it, it is convenient to view a juncture as an M-labeled tree, where only the root is branching. Figure 3 pictures the juncture $(s_0, \{(r_1, r_2), (s_1), (t_1, t_2, t_3)\})$, which abstracts the set of chains $\{(s_0, r_1, r_2), (s_0, s_1), (s_0, t_1, t_2, t_3)\}$ "synchronized" at s_0.

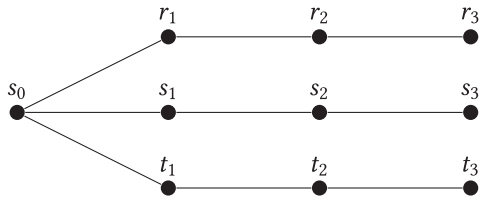


Fig. 4. The juncture $\left\{s_0, \left\{(r_1, r_2, r_3), (s_1, s_2, s_3), (t_1, t_2, t_3)\right\}\right\} \in M \times 2^{M^3}$.

If $(s, \mathcal{S})$ and $(t, \mathcal{T})$ are junctures, we write $(s, \mathcal{S}) \subseteq (t, \mathcal{T})$ when $s = t$ and $\mathcal{S} \subseteq \mathcal{T}$, i.e., when $\{\bar{s} \mid \bar{s} \in (s, \mathcal{S})\} \subseteq \{\bar{t} \mid \bar{t} \in (t, \mathcal{T})\}$. In other words, this means that the tree representing $(s, \mathcal{S})$ is obtained from the tree representing $(t, \mathcal{T})$ by simply removing some branches from the root.

Note that a chain is, in particular, a juncture. For this reason, we use the same notation for sets of chains and sets of junctures: $\mathcal{R}, \mathcal{S}, \mathcal{T}, \ldots$. If $\mathcal{T}$ is a set of junctures, we define $\downarrow\mathcal{T}$, the *downset* of $\mathcal{T}$, as the set

$$\downarrow\mathcal{T} = \left\{(r, \mathcal{R}) \mid \exists (s, \mathcal{S}) \in \mathcal{T},\ (r, \mathcal{R}) \subseteq (s, \mathcal{S})\right\}.$$

In other words, $\downarrow\mathcal{T}$ is the set of junctures represented by trees obtained by possibly removing some branches to trees in $\mathcal{T}$.

Finally, for any $n \geqslant 1$, a juncture $(s, \mathcal{S})$ is said to have *length* n when $\mathcal{S} \subseteq M^{n-1}$, i.e., when chains $\bar{s} \in (s, \mathcal{S})$ all have the same length n. In this article, we shall use only these junctures, such as the one pictured in Figure 4.

Observe that, for all $n \geqslant 1$, the set $M \times 2^{M^{n-1}}$ of junctures of length n is a monoid for the following operation:

$$(s, \mathcal{S}) \cdot (t, \mathcal{T}) = (s \cdot t,\ \mathcal{S} \cdot \mathcal{T}) = \left(st, \{\bar{s}\bar{t} \in M^{n-1} \mid \bar{s} \in \mathcal{S}, \quad \bar{t} \in \mathcal{T}\}\right).$$

As for chains, we are not interested in all junctures but rather only in those that carry information with respect to $\Sigma_i(<)$ for some i. We call these particular junctures Σ_i*-junctures*.

Σ_i-Junctures. To define Σ_i-junctures, we mimic the definition of Σ_i-chains. Fix $i \geqslant 1$. We begin by defining a set of $\Sigma_i[k]$-junctures for each fixed quantifier rank k. For all $k \in \mathbb{N}$, we define the set $\mathcal{J}_i^k[\alpha]$ of $\Sigma_i[k]$*-junctures for* α. Let $(t, \mathcal{T})$ be a juncture. We let $(t, \mathcal{T}) \in \mathcal{J}_i^k[\alpha]$ if

— all chains in $\mathcal{T}$ have the same length, say, $n - 1$, and
— there exists $w \in A^*$ such that $\alpha(w) = t$, and for all chains $(t_2, \ldots, t_n) \in \mathcal{T}$, there exist $w_2, \ldots, w_n \in A^*$ satisfying

$$w \lesssim_i^k w_2 \lesssim_i^k \cdots \lesssim_i^k w_n, \tag{1}$$

and for all $j = 2, \ldots, n$,

$$\alpha(w_j) = t_j. \tag{2}$$

We call such a word w a k*-witness* of the $\Sigma_i[k]$-juncture. With the tree representation of $\Sigma_i[k]$-junctures, as in Figure 4, this means that we can actually label each node by *two* values: one in the finite monoid M (the same value as in Figure 4), and one in A^*, such that

— for every node whose labeling in M is s and whose labeling in A^* is u, we have that $\alpha(u) = s$, and
— for every edge from a node labeled $u \in A^*$ to one of its children labeled $v \in A^*$, we have that $u \lesssim_i^k v$.

Thus, a k-witness of the $\Sigma_i[k]$-juncture is a possible word-labeling of the root.

We finally define $\mathcal{J}_i[\alpha]$, the *set of Σ_i-junctures for α* as the set

$$\mathcal{J}_i[\alpha] = \bigcap_k \mathcal{J}_i^k[\alpha].$$

An immediate observation, already mentioned above, is that Σ_i-junctures carry at least as much information as Σ_i-chains, as stated in the following fact.

Fact 5.5. *Let $i \geqslant 1$ and let $(s_1, \ldots, s_n)$ be a chain. Then,*

$$(s_1, \ldots, s_n) \in C_i[\alpha] \text{ if and only if } \Big(s_1, \big\{(s_2, \ldots, s_n)\big\}\Big) \in \mathcal{J}_i[\alpha].$$

Recall that we will restrict ourselves to junctures of a fixed length $n \geqslant 1$. We denote by $\mathcal{J}_{i,n}^k[\alpha]$ and $\mathcal{J}_{i,n}[\alpha]$ the corresponding restrictions:

$$\mathcal{J}_{i,n}^k[\alpha] = \mathcal{J}_i^k[\alpha] \cap (M \times 2^{M^{n-1}}),$$
$$\mathcal{J}_{i,n}[\alpha] = \mathcal{J}_i[\alpha] \cap (M \times 2^{M^{n-1}}).$$

For example, the Σ_i-juncture of Figure 4 belongs to $\mathcal{J}_{i,4}[\alpha]$. Observe that Lemma 5.1 can be generalized to Σ_i-junctures, as stated in the following lemma.

Lemma 5.6. *For every $i \geqslant 1$ and $k, n \in \mathbb{N}$, we have that*

$$\mathcal{J}_i[\alpha] \subseteq \mathcal{J}_i^{k+1}[\alpha] \subseteq \mathcal{J}_i^k[\alpha].$$
$$\mathcal{J}_{i,n}[\alpha] \subseteq \mathcal{J}_{i,n}^{k+1}[\alpha] \subseteq \mathcal{J}_{i,n}^k[\alpha].$$

Moreover, for all $i, n \in \mathbb{N}$, there exists $\ell_{i,n} \in \mathbb{N}$ such that for every $k \geqslant \ell_{i,n}$,

$$\mathcal{J}_{i,n}[\alpha] = \mathcal{J}_{i,n}^{\ell_{i,n}}[\alpha] = \mathcal{J}_{i,n}^k[\alpha].$$

Note that it is immediate from the definitions that the bound $\ell_{i,n}$ in Lemma 5.6 is also an upper bound on $\kappa_{i,n}$ in Lemma 5.1. We will obtain an upper bound on $\ell_{2,n}$ when proving the completeness of our algorithm computing Σ_2-junctures in Section 7.

Closure Properties. We finish the section by generalizing the closure properties of Σ_i-chains to Σ_i-junctures. An illustration of all four closure properties can be found in Figure 5.

The first property we state is closure under subsets.

Fact 5.7. *Let $i, k \geqslant 1$ and let $\mathcal{X} = \mathcal{J}_i[\alpha]$ or $\mathcal{X} = \mathcal{J}_i^k[\alpha]$. Then, $\mathcal{X}$ is closed under the following operation: for all $(r, \mathcal{R}) \in \mathcal{X}$ and all $\mathcal{R}' \subseteq \mathcal{R}$, we have that $(r, \mathcal{R}') \in \mathcal{X}$. In other words, we have that $\downarrow\mathcal{X} = \mathcal{X}$.*

We now generalize closure under subwords to Σ_i-junctures.

Fact 5.8. *Let $i, k \geqslant 1$ and let $\mathcal{X} = \mathcal{J}_i[\alpha]$ or $\mathcal{X} = \mathcal{J}_i^k[\alpha]$. Then, $\mathcal{X}$ is closed under the following operation: let $(r, \mathcal{R}) \in \mathcal{X}$ and let $\mathcal{R}'$ be a set of chains of the same length that are all subwords of chains in $\mathcal{R}$. Then, $(r, \mathcal{R}') \in \mathcal{X}$.*

We next generalize closure under stutter to Σ_i-junctures.

Fact 5.9. *Let $i, k \geqslant 1$ and let $\mathcal{X} = \mathcal{J}_i[\alpha]$ or $\mathcal{X} = \mathcal{J}_i^k[\alpha]$. Then, $\mathcal{X}$ is closed under the following operation: let $(r, \mathcal{R}) \in \mathcal{X}$ and let $\mathcal{R}'$ be a set of chains of the same length, each of the form $r^j \bar{r}'$, where $j \geqslant 0$ and $\bar{r}'$ is a stutter of some chain in $\mathcal{R}$. Then, $(r, \mathcal{R}') \in \mathcal{X}$.*

It remains to generalize closure under product.

Fact 5.10. *For all $k, n \in \mathbb{N}$, $\mathcal{J}_{i,n}[\alpha]$ and $\mathcal{J}_{i,n}^k[\alpha]$ are submonoids of $M \times 2^{M^{n-1}}$.*

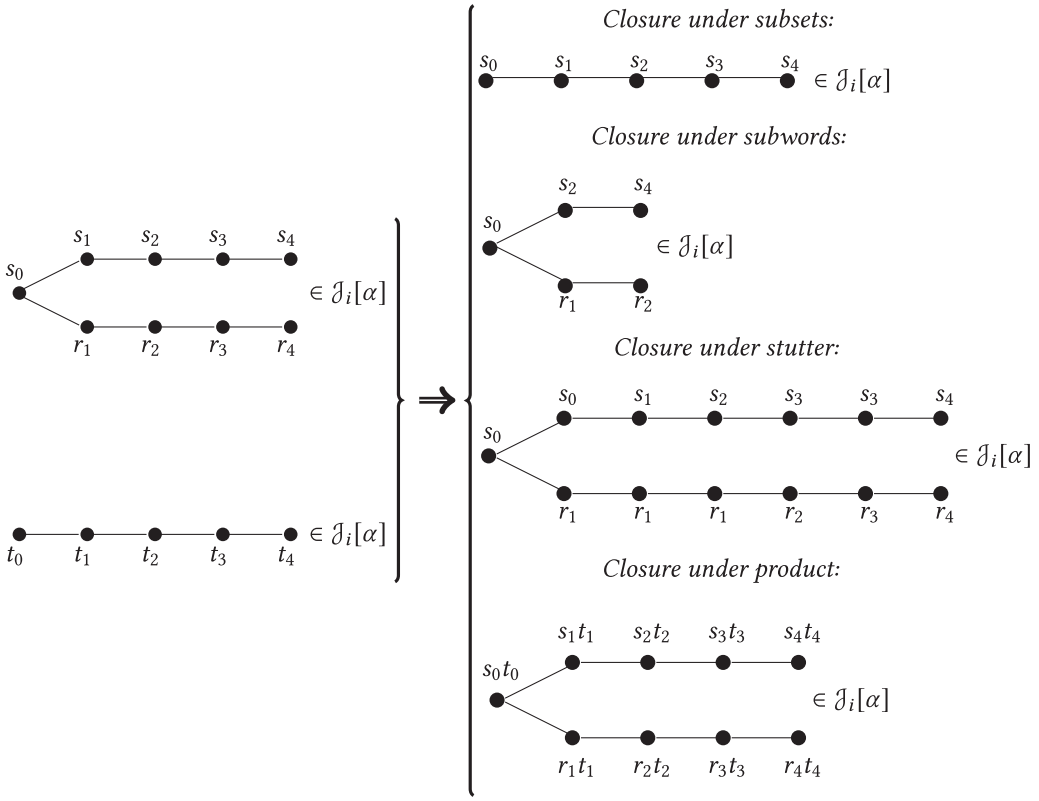


Fig. 5. Closure properties of Σ_i-junctures (example on $\mathcal{J}_i[\alpha]$).

6 GENERIC RESULTS: FROM Σ_i-CHAINS TO SEPARATION AND MEMBERSHIP

In this section, we make explicit the connection between Σ_i-chains and our two decision problems: membership and separation. For all levels in the hierarchy, we prove that both problems can be reduced to the computation of specific information about the set of Σ_i-chains associated to a morphism recognizing both input languages. Of course, the amount of required information depends on whether we consider $\Sigma_i(<)$ or $\mathcal{B}\Sigma_i(<)$ and on whether we consider membership or separation. Note that in order to be stated and proved, all of these theorems require only Σ_i-chains: Σ_i-junctures are not needed. The section is organized into three parts.

—In the first one, we explain the most immediate link: separation for $\Sigma_i(<)$ reduces to computing all Σ_i-chains of length 2.
—In the second part, we prove that deciding membership for $\Sigma_i(<)$ requires less information: only the Σ_{i-1}-chains of length 2 are needed.
—In the last part, we prove reductions for $\mathcal{B}\Sigma_i(<)$.

6.1 The Separation Problem for $\Sigma_i(<)$

Theorem 6.1. *Let L_1, L_2 be regular languages that are both recognized by a morphism $\alpha : A^* \to M$ into a finite monoid M and let $F_1, F_2 \subseteq M$ be the corresponding accepting sets. Let $i \in \mathbb{N}$. Then, the following properties hold:*

(1) *L_1 is $\Sigma_i(<)$-separable from L_2 if and only if, for all $s_1, s_2 \in F_1, F_2$, we have that $(s_1, s_2) \notin C_i[\alpha]$.*
(2) *L_1 is $\Pi_i(<)$-separable from L_2 if and only if, for all $s_1, s_2 \in F_1, F_2$, we have that $(s_2, s_1) \notin C_i[\alpha]$.*

Theorem 6.1 reduces $\Sigma_i(<)$-separation to finding an algorithm that, given a morphism α, computes all of the associated Σ_i-chains of length 2. This computation is simple when $i = 1$, and actually already known [80]. In Section 7, we present an algorithm for the case $i = 2$. In fact, we do not compute Σ_i-chains directly: our algorithm computes the more general set of Σ_i-junctures, $\mathcal{J}_i[\alpha]$, and Σ_i-chains are then recovered from this set using Fact 5.5. This makes Theorem 6.1 effective for $i \leqslant 2$. The problem has also been solved recently for $i \geqslant 3$, although the proof is much more involved [78]. We finish this section with the proof of Theorem 6.1.

Proof of Theorem 6.1. We prove Item (1). Item (2) is obtained by symmetry. Assume first that L_1 is $\Sigma_i(<)$-separable from L_2 and let K be a separator. By contradiction, suppose that there exist $s_1, s_2 \in F_1, F_2$ such that $(s_1, s_2) \in C_i[\alpha]$. By definition, we know that K can be defined by a $\Sigma_i(<)$ formula. Let k be its quantifier rank. By hypothesis, we have that $(s_1, s_2) \in C_i^k[\alpha]$ so that there exist w_1, w_2 mapped by α to s_1, s_2, respectively, such that $w_1 \lesssim_i^k w_2$. In particular, we obtain $w_1 \in L_1 \subseteq K$ and $w_2 \in L_2$. Moreover, by choice of k and since $w_1 \lesssim_i^k w_2$, we also have that $w_2 \in K$. This is a contradiction since K is by hypothesis a separator; thus, it cannot intersect L_2.

It remains to prove the other direction. Assume that, for all $s_1, s_2 \in F_1, F_2$, we have that $(s_1, s_2) \notin C_i[\alpha]$ and let $\ell = \kappa_{i,2}$ be as defined in Lemma 5.1, that is, such that $C_{i,2}[\alpha] = C_{i,2}^{\kappa_{i,2}}[\alpha]$. We claim that the language

$$K = \{w \mid \exists w_1 \in L_1 \text{ such that } w_1 \lesssim_i^\ell w\},$$

which is $\Sigma_i(<)$-definable by Fact 4.2, is a separator. Indeed, K clearly contains L_1. If K intersects L_2, then, by definition of ℓ, there would exist $s_1, s_2 \in F_1, F_2$ such that $(s_1, s_2) \in C_i[\alpha]$, which is false by hypothesis. □

Remark. Note that the above proof of Theorem 6.1 shows that if two languages recognized by α are $\Sigma_i(<)$-separable, then they are separable by a $\Sigma_i(<)$ formula of rank at most $\kappa_{i,2}$. In other words, the rank k at which the sets $C_{i,2}^k[\alpha]$ stabilize is an upper bound for the rank of possible separators of languages recognized by α.

6.2 The Membership Problem for $\Sigma_i(<)$

We now prove that solving membership for $\Sigma_i(<)$ requires less information than separation: only the Σ_{i-1}-chains of length 2 need to be computed.

Theorem 6.2. *Let $i \geqslant 1$ and let L be a regular language and $\alpha : A^* \to M$ be its syntactic morphism. For all $i \geqslant 1$, L is definable in $\Sigma_i(<)$ if and only if M satisfies the following property:*

$$s^\omega \leqslant s^\omega t s^\omega \quad \text{for all } (t, s) \in C_{i-1}[\alpha]. \tag{3}$$

It follows from Theorem 6.2 that it suffices to compute the Σ_{i-1}-chains of length 2 in order to decide whether a language is definable in $\Sigma_i(<)$. Also, observe that when $i = 1$, by definition we have that $(t, 1_M) \in C_0[\alpha]$ for all $t \in M$. Therefore, Equation (3) implies that $1_M \leqslant t$ for all $t \in M$. Conversely, multiplying this inequality on the left and on the right by s^ω yields back (3) for all $s, t \in C_0[\alpha]$. Consequently, Equation (3) may be rephrased as $1_M \leqslant t$ for all $t \in M$, which is the already known equation for $\Sigma_1(<)$ [76]. Similarly, when $i = 2$, (3) can be rephrased as $s^\omega \leqslant s^\omega t s^\omega$ whenever t is a scattered subword of s (meaning that t is the image under α of a word $a_1 \cdots a_n$ and s the image of a word of the form $v_0 a_1 v_1 \cdots v_{n-1} a_n v_n$, where the a_is are letters and the v_is are possibly empty words). Again, this is the previously known equation for $\Sigma_2(<)$ [15, 76].

Observe that by definition of $\Pi_i(<)$ and $\Delta_i(<)$, we get characterizations for these classes as immediate corollaries: recall that a language is $\Pi_i(<)$-definable if its complement is $\Sigma_i(<)$-definable, and that it is $\Delta_i(<)$-definable if it is both $\Sigma_i(<)$-definable and $\Pi_i(<)$-definable.

Corollary 6.3. *Let L be a regular language and let $\alpha : A^* \to M$ be its syntactic morphism. For all $i \geqslant 1$, the following properties hold:*

—L is definable in $\Pi_i(<)$ if and only if M satisfies $s^\omega \geqslant s^\omega t s^\omega$ for all $(t, s) \in C_{i-1}[\alpha]$.
—L is definable in $\Delta_i(<)$ if and only if M satisfies $s^\omega = s^\omega t s^\omega$ for all $(t, s) \in C_{i-1}[\alpha]$.

It now remains to prove Theorem 6.2. For the proof, we assume that $i \geqslant 2$ (a proof for the case $i = 1$ can be found in [76]). We begin with the simpler "only if" direction, which is an application of Lemma 4.8 and is stated in the next proposition.

Proposition 6.4. *Let L be a $\Sigma_i(<)$-definable language and let $\alpha : A^* \to M$ be its syntactic morphism. Then, α satisfies (3).*

Proof. By hypothesis, L is defined by some $\Sigma_i(<)$ formula φ. Let k be its quantifier rank. Let $(t, s) \in C_{i-1}[\alpha]$. We need to prove that $s^\omega \leqslant s^\omega t s^\omega$. Since $(t, s) \in C_{i-1}[\alpha]$, by definition, there exist v, u such that $\alpha(v) = t$, $\alpha(u) = s$ and $v \lesssim_{i-1}^k u$. By the Σ_i-Property Lemma (Lemma 4.8), we immediately obtain

$$u^{2^k\omega} u^{2^k\omega} \lesssim_i^k u^{2^k\omega} v u^{2^k\omega}.$$

It follows from the Precongruence Lemma (Lemma 4.6) that, for any $w_1, w_2 \in A^*$, we have that

$$w_1 u^{2^k\omega} u^{2^k\omega} w_2 \lesssim_i^k w_1 u^{2^k\omega} v u^{2^k\omega} w_2.$$

By choice of k and definition of $\lesssim_i^k$, this means that $w_1 u^{2^k\omega} w_2 \in L$ implies that $w_1 u^{2^k\omega} v u^{2^k\omega} w_2 \in L$. By definition of the syntactic preorder, this means that $s^\omega \leqslant s^\omega t s^\omega$. □

It now remains to prove the harder "if" direction of Theorem 6.2. We use induction to construct a formula for the language L. We rely on Simon's Factorization Forest Theorem for the induction, which we state in the following proposition.

Proposition 6.5. *Let $i \geqslant 2$ and let $\alpha : A^* \to M$ be a morphism into a finite ordered monoid M that satisfies (3). Then for all $h \geqslant 1$ and all $s \in M$, there exists a $\Sigma_i(<)$ formula φ such that for all $w \in A^*$:*

—if $w \models \varphi$ then $s \leqslant \alpha(w)$.
—if $\alpha(w) = s$ and w admits an α-factorization forest of height at most h then $w \models \varphi$.

Assume for now that Proposition 6.5 holds and let L be a regular language whose syntactic morphism $\alpha : A^* \to M$ satisfies Equation (3). Given $h = 3|M| - 1$, for all $s \in M$, we denote by φ_s the Σ_i formula associated to s by Proposition 6.5. Since, by Theorem 4.9, all words admit an α-factorization forest of height at most $3|M| - 1$, we have that

(1) if $w \models \varphi_s$, then $s \leqslant \alpha(w)$.
(2) if $\alpha(w) = s$, then $w \models \varphi_s$.

Let F be the accepting set of L and define $\varphi = \bigvee_{s \in F} \varphi_s$. By Item (2) above, we have that $L \subseteq \{w \mid w \models \varphi\}$. Moreover, by definition of recognizability by an ordered monoid, the set F is upward closed, that is, if $s \in F$ and $s \leqslant t$, then $t \in F$. Hence, Item (1) above implies that $\{w \mid w \models \varphi\} \subseteq L$. We conclude that φ defines L. This finishes the proof of Theorem 6.2. It now remains to prove Proposition 6.5.

Proof of Proposition 6.5. Let $h \geqslant 1$ and $s \in M$. We construct the formula by induction on h. Assume first that $h = 1$. Note that the words having an α-factorization forest of height at most 1 are either single letters or the empty word. Consider the language $L_s = \{w \mid |w| \leqslant 1 \text{ and } \alpha(w) = s\}$. Since L_s is finite, it can be defined by a $\Sigma_i(<)$ formula φ (since $i \geqslant 2$, for any word w, one can easily define a $\Sigma_2(<)$ formula whose only model is w). By definition, φ satisfies the conditions of Proposition 6.5.

Assume now that $h > 1$. There are two cases depending on whether s is idempotent or not. We treat the idempotent case (the other case is essentially a simpler version of this proof). Hence, we assume that s is an idempotent, which we denote by e. We first construct φ and then prove that it satisfies the conditions of the proposition. It is defined as the disjunction of several formulas that we define first.

Using Induction. For all $t \in M$, one can use induction to construct a $\Sigma_i(<)$ formula ψ_t such that, for all $w \in A^*$,

- if $w \models \psi_t$, then $t \leqslant \alpha(w)$.
- if $\alpha(w) = t$ and w admits an α-factorization forest of height at most $(h-1)$, then $w \models \psi_t$.

By restricting quantifications, one can modify each of these formulas to construct two other formulas $\psi_t^\ell(x)$ and $\psi_t^r(x)$, both having a single free variable x and such that

- $w, x \models \psi_t^\ell(x)$ if and only if the prefix u of w obtained by keeping only positions $y < x$ satisfies ψ_t.
- $w, x \models \psi_t^r(x)$ if and only if the suffix v of w obtained by keeping only positions $y \geqslant x$ satisfies ψ_t.

Note that these formulas do not have extra quantifiers so that they also belong to $\Sigma_i(<)$.

Using Π_{i-1}. Recall that by Lemma 5.1, there exists an integer κ such that, for all $k \geqslant \kappa$

$$C_{i-1,2}^k[\alpha] = C_{i-1,2}[\alpha]$$

Consider the language

$$K = \bigcup_{w \in \alpha^{-1}(e)} \{u \mid u \lessapprox_{i-1}^{\kappa} w\}.$$

By choice of κ, for any $u \in K$, we have that $(\alpha(u), e) \in C_{i-1,2}^\kappa[\alpha] = C_{i-1,2}[\alpha]$. Since $e = e^2$, one may use Equation (3) to obtain that for all $u \in K$

$$e \leqslant e\alpha(u)e. \tag{4}$$

Moreover, by the dual version of Fact 4.2, K can be defined by a $\Pi_{i-1}(<)$ formula Γ (Γ is $\Sigma_i(<)$). We define $\Gamma(x, y)$ as the formula with two free variables x, y such that $w, x, y \models \Gamma(x, y)$ if and only if $x < y$ and the infix u obtained by keeping all positions z in w such that $x \leqslant z < y$ satisfies Γ. Note again that this formula can be chosen in $\Sigma_i(<)$.

Definition of φ. Finally, we can define the desired formula. It is the disjunction of three subformulas. Intuitively, the first captures words having an α-factorization forest of height at most $h-1$; the second captures words having an α-factorization forest of height h and whose root is a binary node; and the thirdcaptures words with an α-factorization forest of height h and whose root is an idempotent node.

$$\varphi = \psi_e \vee \left(\bigvee_{t_1 t_2 = e} \exists x \, \psi_{t_1}^\ell(x) \wedge \psi_{t_2}^r(x) \right) \vee \left(\exists x \exists y \; x < y \wedge \psi_e^\ell(x) \wedge \Gamma(x, y) \wedge \psi_e^r(y) \right)$$

Note that, by definition, φ is a $\Sigma_i(<)$ formula. We need to prove that it satisfies the conditions of the proposition.

Choose some $w \in A^*$ and assume first that $w \models \varphi$. We need to prove that $e \leqslant \alpha(w)$.

- If $w \models \psi_e$, then this is by definition of ψ_e.
- If $w \models \exists x\ \psi^\ell_{t_1}(x) \wedge \psi^r_{t_2}(x)$ for $t_1 t_2 = e$, then, by definition, $w = w_1 w_2$ with $t_1 \leqslant \alpha(w_1)$ and $t_2 \leqslant \alpha(w_2)$. It follows that $e = t_1 t_2 \leqslant \alpha(w_1 w_2) = \alpha(w)$.
- Finally, if $w \models \exists x \exists y\ x < y \wedge \psi_e(x) \wedge \Gamma(x, y) \wedge \psi_e(y)$, we obtain that $w = w_1 u w_2$ with $e \leqslant \alpha(w_1)$, $u \in K$, and $e \leqslant \alpha(w_2)$. By Equation (4), we know that $e \leqslant e\alpha(u)e \leqslant \alpha(w_1 u w_2) = \alpha(w)$, which terminates this direction.

Conversely, assume that $\alpha(w) = e$ and that w admits an α-factorization forest of height at most h. We have to prove that w satisfies φ. There are again three cases.

- First, if w has an α-factorization forest of height at most $h - 1$, then $w \models \psi_e$, so $w \models \varphi$.
- Second, if w admits an α-factorization forest of height h whose root is a binary node, then $w = w_1 w_2$ with w_1, w_2 admitting forests of height at most $h - 1$. Let $t_1 = \alpha(w_1)$ and $t_2 = \alpha(w_2)$. Observe that $t_1 t_2 = \alpha(w) = e$. By the induction hypothesis and definition of the formulas ψ_t, we have that $w_1 \models \psi_{t_1}$ and $w_2 \models \psi_{t_2}$; hence, $w \models \exists x\ \psi^\ell_{t_1}(x) \wedge \psi^r_{t_2}(x)$. It follows that $w \models \varphi$ since $t_1 t_2 = e$.
- Finally, if w admits an α-factorization forest of height h whose root is an idempotent node, then $w = w_1 u w_2$ with $\alpha(w_1) = \alpha(u) = \alpha(w_2) = e$ and w_1, w_2 admitting forests of height at most $h - 1$. It follows that $w_1 \models \psi_e$ and $w_2 \models \psi_e$. Moreover, since $\alpha(u) = e$, it is immediate that $u \in K$; hence, $u \models \Gamma$. We conclude that $w \models \exists x \exists y\ x < y \wedge \psi^\ell_e(x) \wedge \Gamma(x, y) \wedge \psi^r_e(y)$, whence $w \models \varphi$.

This concludes the proof of Proposition 6.5. □

6.3 Separation and Membership for $\mathcal{B}\Sigma_i(<)$

In this last part, we prove that being able to compute more information about the set of Σ_i-chains yields solutions to both separation and membership for $\mathcal{B}\Sigma_i(<)$. What is needed is a property called *alternation*, which we define now.

Alternation. Let M be a finite monoid. We say that a chain $(s_1, \ldots, s_n) \in M^*$ has *alternation* ℓ if there are exactly ℓ indices i such that $s_i \neq s_{i+1}$. We say that a set of chains $\mathcal{S}$ has *bounded alternation* if there exists a bound $\ell \in \mathbb{N}$ such that all chains in $\mathcal{S}$ have alternation at most ℓ.

THEOREM 6.6. *Let L_1, L_2 be regular languages, both recognized by the same morphism $\alpha : A^* \to M$ into a finite monoid M and let $F_1, F_2 \subseteq M$ be their respective accepting sets. Let $i \in \mathbb{N}$. Then, L_1 is $\mathcal{B}\Sigma_i(<)$-separable from L_2 if and only if, for all $s_1, s_2 \in F_1, F_2$, $s_1 \neq s_2$ and $C_i[\alpha] \cap \{s_1, s_2\}^*$ has bounded alternation.*

Theorem 6.6 reduces the separation problem for $\mathcal{B}\Sigma_i(<)$ to finding an algorithm that, given a morphism α, computes all pairs $(s_1, s_2) \in M^2$ such that $C_i[\alpha] \cap \{s_1, s_2\}^*$ has bounded alternation. The problem has been solved when $i = 1$ in [80]. Above $i = 1$, the problem remains open, even when $i = 2$. Note that, due to closure of $C_i[\alpha]$ under subwords, $C_i[\alpha] \cap \{s_1, s_2\}^*$ has unbounded alternation if and only if it contains the language of all chains $(s_1, s_2, s_1, s_2, \ldots, s_1, s_2)$, which we denote by $(s_1, s_2)^*$.

Before proving Theorem 6.6, we establish a simple corollary stating that solving *membership* for $\mathcal{B}\Sigma_i(<)$ requires slightly less information. This statement will allow us to solve membership for $\mathcal{B}\Sigma_2(<)$ in Section 8.

Corollary 6.7. *Let L be a regular language and let $\alpha : A^* \to M$ be its syntactic morphism. Then, L is definable in $\mathcal{B}\Sigma_i(<)$ if and only if $C_i[\alpha]$ has bounded alternation.*

Proof. Recall that L is $\mathcal{B}\Sigma_i(<)$-definable if and only if L is $\mathcal{B}\Sigma_i(<)$-separable from its complement. We prove both directions by contrapositive. Let $F = \alpha(L)$ be the accepting set of L.

Assume first that L is not definable in $\mathcal{B}\Sigma_i(<)$. By Theorem 6.6, this means that there exist $s, t \in M$ such that $s \in F$ and $t \notin F$ and $C_i[\alpha] \cap \{s, t\}^*$ has unbounded alternation. Hence, $C_i[\alpha]$ has unbounded alternation.

For the converse, we use the fact that α is the syntactic morphism of L. Assume that $C_i[\alpha]$ has unbounded alternation. By definition and since $C_i[\alpha]$ is closed under subwords, this means that there exist $s, t \in M$ such that $C_i[\alpha] \cap \{s, t\}^*$ has unbounded alternation. Since α is the syntactic morphism of L, there exist $r, r' \in M$ such that either $rsr' \in F$ and $rtr' \notin F$ or $rtr' \in F$ and $rsr' \notin F$. In both cases, $C_i[\alpha] \cap \{rsr', rtr'\}^*$ has unbounded alternation, since Σ_i-chains are closed under product. By Theorem 6.6, it follows that L is not $\mathcal{B}\Sigma_i(<)$-separable from its complement, whence it is not definable in $\mathcal{B}\Sigma_i(<)$. □

It remains to prove Theorem 6.6, which we do in the rest of this section.

Proof of Theorem 6.6. There are two directions, both proved by contrapositive.

Assume first that L_1 is not $\mathcal{B}\Sigma_i(<)$-separable from L_2. We have to find $s_1, s_2 \in F_1, F_2$ such that $s_1 = s_2$ or such that $C_i[\alpha] \cap \{s_1, s_2\}^*$ has unbounded alternation. Using Fact 4.4, for all k, one can find $w_{1,k} \in L_1$ and $w_{2,k} \in L_2$ such that $w_{1,k} \cong_i^k w_{2,k}$. Since M is finite, we may assume without loss of generality that there exist $s_1, s_2 \in M$ such that, for all k, $\alpha(w_{1,k}) = s_1$ and $\alpha(w_{2,k}) = s_2$. Observe that, by definition, $s_1 \in F_1$ and $s_2 \in F_2$. If $s_1 = s_2$, then we are done. Otherwise, $s_1 \neq s_2$ and we prove that $C_i[\alpha] \cap \{s_1, s_2\}^*$ has unbounded alternation. Indeed, for all k, we have that

$$w_{1,k} \lesssim_i^k w_{2,k} \lesssim_i^k w_{1,k} \lesssim_i^k w_{2,k} \lesssim_i^k w_{1,k} \lesssim_i^k w_{2,k} \lesssim_i^k \cdots$$

Hence, by definition, $(s_1, s_2)^* \subseteq C_i[\alpha]$, which terminates the proof of this direction.

Conversely, assume that there exist $s_1 \in F_1$ and $s_2 \in F_2$ such that $C_i[\alpha] \cap \{s_1, s_2\}^*$ has unbounded alternation. We prove that L_1 and L_2 are not $\mathcal{B}\Sigma_i(<)$-separable. More precisely, we show that, for all $k \in \mathbb{N}$, there exist $w_1 \in L_1$ and $w_2 \in L_2$ such that $w_1 \cong_i^k w_2$. The result will then follow from Fact 4.4 again.

Let $k \in \mathbb{N}$ and let n be the number of equivalence classes of $\cong_i^k$ (recall that $\cong_i^k$ has finite index). Consider the chain $(s_1, s_2)^{n+1} \in C_i^k[\alpha]$, that is, the chain $(s_1, s_2, s_1, s_2, \ldots, s_1, s_2)$ of length $2(n+1)$. By definition, there exist words $u_1, \ldots, u_{n+1}$ mapped to s_1 under α and $v_1, \ldots, v_{n+1}$ mapped to s_2 under α, such that

$$u_1 \lesssim_i^k v_1 \lesssim_i^k u_2 \lesssim_i^k v_2 \lesssim_i^k \cdots \lesssim_i^k u_{n+1} \lesssim_i^k v_{n+1}.$$

By choice of n and by the pigeonhole principle, we get that $j < j'$ such that $u_j \cong_i^k u_{j'}$. Hence,

$$u_j \lesssim_i^k v_j \lesssim_i^k u_{j'} \lesssim_i^k u_j.$$

It follows that $u_j \cong_i^k v_j$ and it suffices to choose $w_1 = u_j$ and $w_2 = v_j$ to terminate the proof. □

7 COMPUTING Σ_2-CHAINS

In this section, we present an algorithm that, given a morphism and an integer $n \geqslant 1$ as input, computes all associated Σ_2-chains of length n. We already know by Theorems 6.1 and 6.2 that achieving this for $n = 2$ yields an algorithm deciding the separation problem for $\Sigma_2(<)$ and $\Pi_2(<)$ and algorithms deciding the membership problem for $\Sigma_3(<)$, $\Pi_3(<)$, and $\Delta_3(<)$. In Section 8, we will obtain as well an algorithm deciding the membership problem for $\mathcal{B}\Sigma_2(<)$.

Note that our algorithm is designed to work with *alphabet-compatible* morphisms only. As shown in the next lemma, this not restrictive: the problem of computing the Σ_2-chains associated to any morphism can always be reduced to this case.

Lemma 7.1. *Let $i \geqslant 1$ and $n \geqslant 1$. Given $\alpha : A^* \to M$ a morphism into a finite monoid M and $\beta : A^* \to M \times 2^A$ its alphabet completion, we have the following property:*

$$C_{i,n}[\alpha] = \Big\{(s_1, \ldots, s_n) \mid \textit{there exist } B_1, \ldots, B_n \in 2^A \textit{ such that } ((s_1, B_1), \ldots, (s_n, B_n)) \in C_{i,n}[\beta]\Big\}.$$

Proof. It is immediate from the definitions that for all $k > 0$, we have that

$$C^k_{i,n}[\alpha] = \Big\{(s_1, \ldots, s_n) \mid \text{there exist } B_1, \ldots, B_n \in 2^A \text{ such that } ((s_1, B_1), \ldots, (s_n, B_n)) \in C^k_{i,n}[\beta]\Big\}.$$

Now, from Lemma 5.1, there exists some $k \in \mathbb{N}$ such that $C^k_{i,n}[\alpha] = C_{i,n}[\alpha]$ and $C^k_{i,n}[\beta] = C_{i,n}[\beta]$. □

We can now present the algorithm. We organize the section into three parts. In the first, we describe the separation algorithm itself. The two remaining parts are devoted to the proofs of its soundness and completeness.

7.1 An Algorithm that Computes Σ_2-chains

For the remainder of this section, we fix an *alphabet-compatible* morphism $\alpha : A^* \to M$ into a finite monoid M. Recall that this means that, for every $s \in M$, $\mathsf{alph}(s)$ is well defined as $\mathsf{alph}(w)$ for any $w \in \alpha^{-1}(s)$. For any fixed $n \geqslant 1$, we explain how to compute the following two sets:

(1) the set $C_{2,n}[\alpha]$ of Σ_2-chains of length n for α.
(2) the set $\mathcal{J}_{2,n}[\alpha]$ of Σ_2-junctures of length n for α.

In fact, our algorithm directly computes the second item, i.e., $\mathcal{J}_{2,n}[\alpha]$. Recall that, by Fact 5.5, this is enough to obtain the first item as well. Note that considering Σ_2-junctures is necessary for the technique to work, even if we are interested only in computing Σ_2-chains.

Outline. We begin by explaining what our algorithm does. For this outline, assume that $n = 2$. Observe that, for all $w \in A^*$, we have that $(\alpha(w), \{\alpha(w)\}) \in \mathcal{J}_{2,2}[\alpha]$. The algorithm starts from the set containing only these trivial Σ_2-junctures and then saturates this set with two operations, which both preserve membership in $\mathcal{J}_{2,2}[\alpha]$. Let us describe these two operations.

— The first one is multiplication: by Fact 5.10, $\mathcal{J}_{2,2}[\alpha]$ is a submonoid of $M \times 2^M$.
— The second operation exploits the following specific property of $\Sigma_2(<)$, which is a consequence of the Σ_i-property Lemma (Lemma 4.8): for all words u, v, w, w', we have that

$$\forall k\ \exists \ell \qquad \Big[w \lesssim_2^k u,\ w \lesssim_2^k v \text{ and } \mathsf{alph}(w') = \mathsf{alph}(w)\Big] \implies w^{2\ell} \lesssim_2^k u^\ell w' v^\ell. \tag{5}$$

This is why Σ_2-junctures are needed: in order to use this property, we need to have a *single word* w such that $w \lesssim_2^k u$ and $w \lesssim_2^k v$, and this information is not provided by Σ_2-chains alone. Once abstracted at the monoid level, Equation (5) yields an operation stating that whenever $(s, \mathcal{S})$ belongs to $\mathcal{J}_{2,2}[\alpha]$, then so does $(s, \mathcal{S})^\omega \cdot (1_M, \mathcal{T}) \cdot (s, \mathcal{S})^\omega$, where $\mathcal{T}$ is the set $\{t \mid \mathsf{alph}(t) = \mathsf{alph}(s)\}$. Note that this is also where we need α to be alphabet compatible.

Let us now formalize this procedure and generalize it to arbitrary length.

Algorithm. As we explained, our algorithm is a *least fixed point*. We start from a set of trivial Σ_2-junctures and saturate this set with two operations until stabilization. Denote by $n \geqslant 1$ the

common length of all chains in junctures we want to compute. We initialize our fixed-point algorithm with $\mathcal{D}_n \subseteq M \times 2^{M^{n-1}}$ defined by

$$\mathcal{D}_n = \left\{(\alpha(w), \{(\alpha(w), \ldots, \alpha(w))\}) \mid w \in A^*\right\}.$$

We now describe our fixed-point operation. To any set of junctures $\mathcal{R} \subseteq M \times 2^{M^{n-1}}$, we associate another subset $\mathrm{Sat}_n(\mathcal{R})$ of $M \times 2^{M^{n-1}}$ containing $\mathcal{R}$, defined as a lowest fixed point (with respect to inclusion). We will then prove that, for all n, one can extract from $\mathrm{Sat}_n(\mathcal{D}_n)$ the set $\mathcal{J}_{2,n}[\alpha]$ (and, therefore, also $C_{2,n}[\alpha]$ by Fact 5.5).

For length $n = 1$, we simply define Sat_1 as the identity, i.e., $\mathrm{Sat}_1(\mathcal{R}) = \mathcal{R}$. This is because, by definition, $\mathcal{J}_{2,1}[\alpha] = \mathcal{D}_1$. We now define Sat_n for a length $n \geqslant 2$. For $\mathcal{R} \subseteq M \times 2^{M^{n-1}}$, we define $\mathrm{Sat}_n(\mathcal{R})$ as the smallest subset of $M \times 2^{M^{n-1}}$ containing $\mathcal{R}$ and satisfying the three following closure properties:

(Op_1) $\downarrow \mathrm{Sat}_n(\mathcal{R}) \subseteq \mathrm{Sat}_n(\mathcal{R})$.
(Op_2) $\mathrm{Sat}_n(\mathcal{R}) \cdot \mathrm{Sat}_n(\mathcal{R}) \subseteq \mathrm{Sat}_n(\mathcal{R})$.
(Op_3) For all $(s, \mathcal{S}) \in \mathrm{Sat}_n(\mathcal{R})$, if $\mathcal{T} = \left\{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathrm{alph}(t_1) = \mathrm{alph}(s)\right\}$, then

$$(s, \mathcal{S})^\omega \cdot (1_M, \mathcal{T}) \cdot (s, \mathcal{S})^\omega \in \mathrm{Sat}_n(\mathcal{R}).$$

It is straightforward that $\mathrm{Sat}_n(\mathcal{R})$ can be effectively computed from $\mathcal{R}$ and $C_{2,n-1}[\alpha]$ using a smallest fixpoint algorithm. Note however that the definition of Sat_n is parametrized by the set $C_{2,n-1}[\alpha]$, *i.e.*, the set of Σ_2-chains of length $n-1$. This means that in order to compute Sat_n, we need to have previously computed the Σ_2-chains of length $n-1$. This set can be computed by the same algorithm at stage $n-1$: indeed, from its output $\mathrm{Sat}_{n-1}(\mathcal{D}_{n-1})$, one can compute the set of Σ_2-junctures of length $n-1$, and then by Fact 5.5, the set of all Σ_2-chains of length $n-1$.

This finishes the definition of the algorithm. Its soundness and completeness are stated in the following proposition.

Proposition 7.2. *Given $n \geqslant 1$ and $\ell_{2,n} = 9n|M|^2 \cdot 2^{|M|^{n-1}}$, we have that*

$$\mathcal{J}_{2,n}[\alpha] = \mathcal{J}_{2,n}^{\ell_{2,n}}[\alpha] = \mathrm{Sat}_n(\mathcal{D}_n). \tag{6}$$

Proposition 7.2 establishes both soundness and completeness of the algorithm:

— the inclusion $\mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}[\alpha]$ gives its soundness: $\mathrm{Sat}_n(\mathcal{D}_n)$ consists *only* of Σ_2-junctures of length n,
— the containment $\mathrm{Sat}_n(\mathcal{D}_n) \supseteq \mathcal{J}_{2,n}[\alpha]$ gives its completeness: the set $\mathrm{Sat}_n(\mathcal{D}_n)$ contains *all* Σ_2-junctures of length n.

It also establishes a bound $\ell_{2,n}$ on a sufficient quantifier rank, whose existence was already known from Lemma 5.6. This bound is a by-product of the proof of the algorithm. It is of particular interest for separation and Theorem 6.1. Indeed, one can prove that, for any two languages that are $\Sigma_2(<)$-separable and recognized by α, the separator can be chosen with quantifier rank $\ell_{2,2}$ (refer to Remark 6.1). From Theorem 6.1, we also get decidability of the separation problem for $\Sigma_2(<)$, as stated in the following corollary.

Corollary 7.3. *Given as input two regular languages L_1, L_2, it is decidable to test whether L_1 can be $\Sigma_2(<)$-separated (resp., $\Pi_2(<)$-separated) from L_2.*

Similarly, we get decidability of the membership problem for $\Sigma_3(<)$, $\Pi_3(<)$, and $\Delta_3(<)$ from Theorem 6.2.

Corollary 7.4. *Given as input a regular language L, the following problems are decidable:*

—*whether L is definable in* $\Sigma_3(<)$.
—*whether L is definable in* $\Pi_3(<)$.
—*whether L is definable in* $\Delta_3(<)$.

Moreover, we will see in Section 8 that an algorithm for the membership problem for $\mathcal{B}\Sigma_2(<)$ can also be obtained by relying on Proposition 7.2.

It now remains to prove Proposition 7.2, that it is the soundness and completeness of the algorithm. We devote the rest of Section 7 to this proof.

We proceed by induction on n. Observe that when $n = 1$, all three sets $\mathcal{J}_{2,n}[\alpha]$, $\mathcal{J}_{2,n}^{\ell_{2,n}}[\alpha]$, and $\mathrm{Sat}_n(\mathcal{D}_n)$ are, by definition, all equal to $\mathcal{D}_n$. Therefore, the result is immediate for $n = 1$.

Assume now that $n \geqslant 2$ and let $\ell_{2,n}$ and $\ell_{2,n-1}$ be defined as in Proposition 7.2. Our induction hypothesis implies the following fact.

Fact 7.5. *We have that* $\mathcal{J}_{2,n-1}[\alpha] = \mathcal{J}_{2,n-1}^{\ell_{2,n-1}}[\alpha]$. *In particular,* $C_{2,n-1}[\alpha] = C_{2,n-1}^{\ell_{2,n-1}}[\alpha]$.

We shall prove the following inclusions, which clearly entail Equation (6) and Proposition 7.2:

$$\mathcal{J}_{2,n}[\alpha] \subseteq \mathcal{J}_{2,n}^{\ell_{2,n}}[\alpha] \subseteq \mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}[\alpha].$$

That $\mathcal{J}_{2,n}[\alpha] \subseteq \mathcal{J}_{2,n}^{\ell_{2,n}}[\alpha]$ is immediate from Fact 5.6. Hence, two inclusions are left to prove:

—$\mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}[\alpha]$ (corresponding to soundness).
—$\mathcal{J}_{2,n}^{\ell_{2,n}}[\alpha] \subseteq \mathrm{Sat}_n(\mathcal{D}_n)$ (corresponding to completeness).

We give each proof its own section: soundness is shown in Section 7.2 and completeness in Section 7.3. Note that Fact 7.5 (i.e., induction on n) is used only for proving completeness.

7.2 Soundness of the Algorithm

In this section, we prove that $\mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}[\alpha]$. This is a consequence of the following proposition.

Proposition 7.6. *For all* $k \in \mathbb{N}$, $\mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}^k[\alpha]$.

Since, by definition, $\mathcal{J}_{2,n}[\alpha] = \bigcap_{k\in\mathbb{N}} \mathcal{J}_{2,n}^k[\alpha]$, it is immediate from Proposition 7.6 that $\mathrm{Sat}_n(\mathcal{D}_n) \subseteq \mathcal{J}_{2,n}[\alpha]$, which terminates the soundness proof.

Proof of Proposition 7.6. Let $k \in \mathbb{N}$. It is immediate from the definitions that $\mathcal{D}_n \subseteq \mathcal{J}_{2,n}^k[\alpha]$. Hence, by definition of Sat_n, it suffices to prove that $\mathcal{J}_{2,n}^k[\alpha]$ is closed under Operations (Op_1), (Op_2), and (Op_3), i.e., that

(1) $\downarrow \mathcal{J}_{2,n}^k[\alpha] \subseteq \mathcal{J}_{2,n}^k[\alpha]$.
(2) $\mathcal{J}_{2,n}^k[\alpha] \cdot \mathcal{J}_{2,n}^k[\alpha] \subseteq \mathcal{J}_{2,n}^k[\alpha]$.
(3) for all $(s, \mathcal{S}) \in \mathcal{J}_{2,n}^k[\alpha]$, if $\mathcal{T} = \{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathsf{alph}(t_1) = \mathsf{alph}(s)\}$, then

$$(s, \mathcal{S})^\omega \cdot (1_M, \mathcal{T}) \cdot (s, \mathcal{S})^\omega \in \mathcal{J}_{2,n}^k[\alpha].$$

Item (1) is exactly Fact 5.7: $\mathcal{J}_{2,n}^k[\alpha]$ is closed under subsets. That Item (2) holds follows from Fact 5.10: $\mathcal{J}_{2,n}^k[\alpha]$ is a submonoid of $M \times 2^{M^{n-1}}$. It remains to prove Item (3). For this, let $(s, \mathcal{S})$ and $\mathcal{T}$ be as in Item (3). Let $B = \mathsf{alph}(s)$. Let

$$(r, \mathcal{R}) = (s, \mathcal{S})^\omega \cdot (1_M, \mathcal{T}) \cdot (s, \mathcal{S})^\omega.$$

We have to prove that $\mathcal{R}$ belongs to $\mathcal{J}_{2,n}^k[\alpha]$.

Let $h = \omega \times 2^{2k}$, where $\omega = \omega(M \times 2^{M^{n-1}})$, so that, by definition, $(s,\mathcal{S})^{\omega} = (s,\mathcal{S})^{h} = (s,\mathcal{S})^{2h}$. Therefore:

$$(r,\mathcal{R}) = (s,\mathcal{S})^{h} \cdot (1_M,\mathcal{T}) \cdot (s,\mathcal{S})^{h}.$$

Since $(s,\mathcal{S}) \in \mathcal{J}^k_{2,n}[\alpha]$, there exists a k-witness $u \in A^*$ for $(s,\mathcal{S})$, i.e., such that

$$\alpha(u) = s,$$

and for every Σ_2-chain $(s_2,\ldots,s_n) \in \mathcal{S}$, there exist $u_2,\ldots,u_n \in A^*$ satisfying

$$\begin{cases} u \lesssim_2^k u_2 \lesssim_2^k \cdots \lesssim_2^k u_n, \\ \forall j,\ \alpha(u_j) = s_j. \end{cases} \tag{7}$$

Observe that $\alpha(u) = s$ implies that $\mathsf{alph}(u) = \mathsf{alph}(s) = B$. Let

$$w = u^{2h}$$

so that $\mathsf{alph}(w) = \mathsf{alph}(u) = B$ and $\alpha(w) = \alpha(u)^{2h} = s^{2h} = r$. We prove that $(r,\mathcal{R}) \in \mathcal{J}^k_{2,n}[\alpha]$ with w as k-witness. It suffices to show that, for any chain $(r_2,\ldots,r_n) \in \mathcal{R}$, there exist $w_2,\ldots,w_n \in A^*$ satisfying $w \lesssim_2^k w_2 \lesssim_2^k \cdots \lesssim_2^k w_n$ and such that $\alpha(w_j) = r_j$ for all j.

Let $(r_2,\ldots,r_n) \in \mathcal{R}$. By definition of $\mathcal{R}$, we have that $(r_2,\ldots,r_n) = (s_2' t_2 s_2'',\ldots,s_n' t_n s_n'')$ with $(s_2',\ldots,s_n')$,$(s_2'',\ldots,s_n'') \in \mathcal{S}^h$ and $(t_2,\ldots,t_n) \in \mathcal{T}$. Since $(s_2',\ldots,s_n') \in \mathcal{S}^h$, using h times Equation (7) and the fact that $\lesssim_2^k$ is a precongruence (Lemma 4.6), we obtain words $u_2',\ldots,u_n' \in A^*$ such that

$$\begin{cases} u^h \lesssim_2^k u_2' \lesssim_2^k \cdots \lesssim_2^k u_n' \\ \forall j,\ \alpha(u_j') = s_j'. \end{cases} \tag{8}$$

Similarly, since $(s_2'',\ldots,s_n'') \in \mathcal{S}^h$, we get that $u_2'',\ldots,u_n'' \in A^*$ such that

$$\begin{cases} u^h \lesssim_2^k u_2'' \lesssim_2^k \cdots \lesssim_2^k u_n'' \\ \forall j,\ \alpha(u_j'') = s_j''. \end{cases} \tag{9}$$

On the other hand, since $(t_2,\ldots,t_n) \in \mathcal{T}$, we obtain that $\mathsf{alph}(t_2) = B$ and $(t_2,\ldots,t_n) \in C_{2,n-1}[\alpha]$. Hence, we get words $v_2,\ldots,v_n \in A^*$, such that

$$\begin{cases} v_2 \lesssim_2^k \cdots \lesssim_2^k v_n \\ \forall j \geqslant 2,\ \alpha(v_j) = t_j. \end{cases} \tag{10}$$

Observe that this implies, in particular, that $\mathsf{alph}(v_2) = B$. For all $j \geqslant 2$, we let

$$w_j = u_j' v_j u_j''.$$

Note that, for all $j \geqslant 2$, $\alpha(w_j) = s_j' t_j s_j'' = r_j$. It remains to prove that $w \lesssim_2^k w_2 \lesssim_2^k \cdots \lesssim_2^k w_n$ to terminate the proof. That $w_2 \lesssim_2^k \cdots \lesssim_2^k w_n$ is immediate by Equations (8), (9), and (10), since $\lesssim_2^k$ is a precongruence (by Lemma 4.6 again). Since $w = u^{2h}$, the remaining inequality to prove is

$$u^h u^h \lesssim_2^k u_2' v_2 u_2''. \tag{11}$$

Since $\lesssim_2^k$ is a precongruence by Lemma 4.6, we know by Equations (8), (9), and (10) that $u^h v_2 u^h \lesssim_2^k u_2' v_2 u_2''$. Therefore, to establish Equation (11), it suffices to prove that

$$u^h u^h \lesssim_2^k u^h v_2 u^h. \tag{12}$$

Recall that, by definition, $\mathsf{alph}(v_2) = \mathsf{alph}(u) = B$. Therefore, it is straightforward that

$$v_2 \lesssim_1^k u^{2^k}. \tag{13}$$

Now, Equation (12) follows from Lemma 4.8 in view of the choice of $h = \omega \times 2^{2k}$ and of Equation (13). □

7.3 Completeness of the Algorithm

We prove that, for any $\ell \geqslant \ell_{2,n} = 9n|M|^2 \cdot 2^{|M|^{n-1}}$, we have that $\mathcal{J}^{\ell}_{2,n}[\alpha] \subseteq \mathrm{Sat}_n(\mathcal{D}_n)$. We denote by k_n the size of the set of junctures of length n, i.e.,

$$k_n = |M \times 2^{M^{n-1}}|.$$

In particular, this means that $\ell_{2,n} = 9n|M|k_n$. The proof is by induction and relies on Simon's Factorization Forests Theorem. To state the induction, we need more terminology.

Generated Junctures. Let $k \in \mathbb{N}$, $w \in A^*$. We let $g^k_n(w) \in M \times 2^{M^{n-1}}$ be the maximal juncture of $\mathcal{J}^k_{2,n}[\alpha]$ that has w as a k-witness. Formally, letting $g^k_n(w) = (\alpha(w), \mathcal{G})$, we have that $(t_2, \ldots, t_n) \in \mathcal{G}$ if and only if there exist $w_2, \ldots, w_n \in A^*$ satisfying

— for all j, $\alpha(w_j) = t_j$.
— $w \lesssim^k_2 w_2 \lesssim^k_2 \cdots \lesssim^k_2 w_n$.

By definition, any $g^k_n(w)$ is a $\Sigma_2[k]$-juncture of length n: $g^k_n(w) \in \mathcal{J}^k_{2,n}[\alpha]$. Moreover, by definition, we have that

$$\mathcal{J}^k_{2,n}[\alpha] = \downarrow \left\{ g^k_n(w) \mid w \in A^* \right\}.$$

We illustrate this definition with two lemmas that will be useful in the proof. The first states that $g^k_n(w)$ gets smaller as k gets larger.

Lemma 7.7. *Let $w \in A^*$, $n \in \mathbb{N}$ and $k < \ell$. We have that*

$$g^{\ell}_n(w) \subseteq g^k_n(w).$$

Proof. Immediate from the fact that if $k < \ell$, then for all u, v, $u \lesssim^{\ell}_2 v \Rightarrow u \lesssim^k_2 v$. □

Our second lemma is a decomposition result that we will use several times.

Lemma 7.8 (Decomposition Lemma). *Let $w, w' \in A^*$ and $k \geqslant 1$. Then,*

$$g^k_n(ww') \subseteq g^{k-1}_n(w) \cdot g^{k-1}_n(w').$$

Proof. Let $(r, \mathcal{R}) = g^k_n(ww')$, $(s, \mathcal{S}) = g^{k-1}_n(w)$, and $(t, \mathcal{T}) = g^{k-1}_n(w')$. By definition, $r = \alpha(ww')$, $s = \alpha(w)$, and $t = \alpha(w')$; hence, $r = st$. It remains to prove that $\mathcal{R} \subseteq \mathcal{S} \cdot \mathcal{T}$. Let $(r_2, \ldots, r_n) \in \mathcal{R}$. By definition, there exist $u_2, \ldots, u_n$ such that for all j, $\alpha(u_j) = r_j$, and

$$ww' \lesssim^k_2 u_2 \lesssim^k_2 \cdots \lesssim^k_2 u_n. \tag{14}$$

Using $(n-1)$ times a simple Ehrenfeucht-Fraïssé argument, one for each $\lesssim^k_2$ relation in Equation (14), we obtain that all words u_j can be decomposed as $u_j = v_j v'_j$ such that

$$\begin{array}{ccccccc} w & \lesssim^{k-1}_2 & v_2 & \lesssim^{k-1}_2 & \cdots & \lesssim^{k-1}_2 & v_n \\ w' & \lesssim^{k-1}_2 & v'_2 & \lesssim^{k-1}_2 & \cdots & \lesssim^{k-1}_2 & v'_n. \end{array}$$

For instance, v_2 and v'_2 are obtained by playing the k-round Ehrenfeucht-Fraïssé game over ww' and u_2, where the first move of Spoiler is to play in ww' on the first letter of w'. The answer of Duplicator in u_2 splits this word into two factors, v_2 and v'_2.

Now for all j, let $s_j = \alpha(v_j)$ and $t_j = \alpha(v'_j)$. By definition, we have that $(s_2, \ldots, s_n) \in \mathcal{S}$ and $(t_2, \ldots, t_n) \in \mathcal{T}$. Moreover, by definition,

$$(r_2, \ldots, r_n) = (s_2 t_2, \ldots, s_n t_n).$$

It follows that $(r_2, \ldots, r_n) \in \mathcal{S} \cdot \mathcal{T}$, which terminates the proof. □

We can now prove that $\mathcal{J}^{\ell}_{2,n}[\alpha] \subseteq \mathrm{Sat}_n(\mathcal{D}_n)$ when $\ell \geqslant \ell_{2,n}$ (recall that $\ell_{2,n} = 9n|M|k_n$, where $k_n = |M \times 2^{M^{n-1}}|$). This is a consequence of the next proposition.

Proposition 7.9. *Let $\ell \geqslant \ell_{2,n}$ and $w \in A^*$; then, $g^{\ell}_n(w) \in \mathrm{Sat}_n(\mathcal{D}_n)$.*

It is immediate from Proposition 7.9 that, for any $\ell \geqslant \ell_{2,n}$, $\left\{g^{\ell}_n(w) \mid w \in A^*\right\} \subseteq \mathrm{Sat}_n(\mathcal{D}_n)$. Since we know that $\mathcal{J}^{\ell}_{2,n}[\alpha] = \downarrow \left\{g^{\ell}_n(w) \mid w \in A^*\right\}$, we obtain that $\mathcal{J}^{\ell}_{2,n}[\alpha] \subseteq \downarrow \mathrm{Sat}_n(\mathcal{D}_n) = \mathrm{Sat}_n(\mathcal{D}_n)$, which yields completeness.

It therefore remains to prove Proposition 7.9. The proof is once again by induction on the height of the α-factorization forest of w. We state the induction in the following proposition. Recall again that $k_n = |M \times 2^{M^{n-1}}|$ is the size of the set of junctures of length n.

Proposition 7.10. *Let $h \geqslant 1$ and let $k \geqslant h \cdot 3k_n + \ell_{2,n-1}$. Then, for every $w \in A^*$ that admits an α-factorization forest of height at most h, we have that $g^{k}_n(w) \in \mathrm{Sat}_n(\mathcal{D}_n)$.*

Proposition 7.9 is a consequence of Proposition 7.10. This is because

—any $w \in A^*$ admits an α-factorization forest of height at most $3|M| - 1$, by Theorem 4.9, and
—one can verify that $\ell_{2,n} \geqslant (3|M| - 1) \cdot 3k_n + \ell_{2,n-1}$.

We now prove Proposition 7.10. Note that this is where we use Fact 7.5, i.e., induction on n.

As in the statement of Proposition 7.10, take $h \geqslant 1$, $k \geqslant h \cdot 3k_n + \ell_{2,n-1}$ and let $w \in A^*$ admitting an α-factorization forest of height at most h. We need to prove that $g^{k}_n(w) \in \mathrm{Sat}_n(\mathcal{D}_n)$. The proof is by induction on h.

If $h = 1$, then w admits an α-factorization forest that is a leaf. In that case, w is a single-letter word $a \in A$ or the empty word ε. Observe that $k \geqslant 2$. Therefore, one can check that the language $\{w\}$ is definable in $\Sigma_2(<)$; hence, $g^{k}_n(w) = (\alpha(w), \{(\alpha(w), \ldots, \alpha(w))\})$. It follows that $g^{k}_n(w) \in \mathcal{D}_n \subseteq \mathrm{Sat}_n(\mathcal{D}_n)$, which finishes the proof for this case.

Assume now that $h > 1$. If the α-factorization forest of w is again a leaf, we conclude as above. Otherwise, we apply induction to the factors given by this factorization forest. In particular, we will use Lemma 7.8 (the Decomposition Lemma) to decompose $g^{k}_n(w)$ according to this factorization forest. Then, once the factors have been treated by induction, we will use the operations in the definition of Sat_n to lift the result to the whole word w. We distinguish two cases depending on the nature of the topmost node in the α-factorization forest of w.

Case 1: The topmost node is a binary node. We use induction on h and Operation (Op_2) in the definition of Sat_n. By hypothesis, $w = w_1 w_2$ with w_1, w_2 words admitting α-factorization forests of respective heights $h_1, h_2 \leqslant h - 1$. Observe that

$$k - 1 \geqslant (h - 1) \times 3k_n + \ell_{2,n-1}.$$

Therefore, we can apply our induction hypothesis to w_1, w_2 and we obtain that $g^{k-1}_n(w_1) \in \mathrm{Sat}_n(\mathcal{D}_n)$ and $g^{k-1}_n(w_2) \in \mathrm{Sat}_n(\mathcal{D}_n)$. By Operation (Op_2) in the definition of Sat_n, it is immediate that $g^{k-1}_n(w_1) \cdot g^{k-1}_n(w_2) \in \mathrm{Sat}_n(\mathcal{D}_n)$. Moreover, by Lemma 7.8 (the Decomposition Lemma), $g^{k}_n(w) \subseteq g^{k-1}_n(w_1) \cdot g^{k-1}_n(w_2)$. It follows from Operation (Op_1) that $g^{k}_n(w) \in \downarrow \mathrm{Sat}_n(\mathcal{D}_n)$, which concludes this case.

Case 2: The topmost node is an idempotent node. This is the most involved case. We use induction on h and the two operations in the definition of Sat_n. Note that this is also where Fact 7.5 (i.e., induction on n in the general proof of Proposition 7.2) is used. We let

$$e = \alpha(w).$$

Observe that by hypothesis of this case, e is an idempotent. Let

$$B = \mathrm{alph}(w) = \mathrm{alph}(e).$$

Also, let

$$\mathcal{T} = \left\{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathrm{alph}(t_1) = B\right\}. \tag{15}$$

Since one can test the alphabet of a word in $\Sigma_2(<)$, all elements of any Σ_2-chain of $\mathcal{T}$ actually have alphabet B.

We begin by summarizing our hypothesis: w admits what we call an (e,p)-decomposition.

(e,p)-Decompositions. For the rest of the section, we let $p = (h-1) \times 3k_n + \ell_{2,n-1}$. Let $u \in A^*$. We say that u admits an (e,p)-*decomposition* $u_1, \ldots, u_m$ if

a) $u = u_1 \cdots u_m$,
b) for all j, $\alpha(u_j) = e$ and
c) for all j, $g_n^p(u_j) \in \mathrm{Sat}_n(\mathcal{D}_n)$.

Note that (b) means that $\alpha(u_j)$ is a constant idempotent. In particular, since α is alphabet compatible, this also implies that all factors u_i have the same alphabet as e, namely, B. Using Fact 7.5 (i.e., induction on n in the general proof of Proposition 7.2), we obtain the following fact.

Fact 7.11. *For any (e,p)-decomposition $u_1, \ldots, u_m$ of a word and for all $i \leqslant j$, we have that $g_n^p(u_i \cdots u_j) \subseteq (e, \mathcal{T})$, where $\mathcal{T}$ is defined by Equation (15).*

Proof. By definition, the "root" of the juncture $g_n^p(u_i \cdots u_j)$ is labeled by $\alpha(u_i \cdots u_j) = e$. Therefore, we may let $(e, \mathcal{T}') = g_n^p(u_i \cdots u_j)$. Since $p \geqslant \ell_{2,n-1}$, that $\mathcal{T}' \subseteq C_{2,n-1}[\alpha]$ follows from Lemma 7.7 and Fact 7.5. Let $(t_1, \ldots, t_{n-1}) \in \mathcal{T}'$; we have to prove that $\mathrm{alph}(t_1) = B$. This is because $t_1 = \alpha(v)$ for some word v satisfying $u_i \cdots u_j \lesssim_2^p v$. Since $p \geqslant 2$, it follows that $\mathrm{alph}(v) = \mathrm{alph}(u_i \cdots u_j) = B$, which terminates the proof. □

We now use the hypothesis of Case 2 to conclude that w admits an (e,p)-decomposition.

Fact 7.12. *The word w admits an (e,p)-decomposition.*

Proof. By hypothesis of Case 2, there exists a decomposition $w_1, \ldots, w_m$ of w that satisfies points (a) and (b). Moreover, for all j, w_j admits an α-factorization forest of height $h_j \leqslant h-1$. Therefore, point (c) is obtained by induction hypothesis on h. □

Recall that we want to prove that $g_n^k(w) \in \mathrm{Sat}_n(\mathcal{D}_n)$. In general, the number of factors m in the (e,p)-decomposition of w can be arbitrarily large. In particular, it is possible that $k - (m-1) < p$. This means that we cannot simply use Lemma 7.8 as we did in the previous case to conclude that $g_n^k(w) \subseteq g_n^p(w_1) \cdots g_n^p(w_m)$. However, we will partition $w_1, \ldots, w_m$ as a bounded number of subdecompositions that we can treat using the second operation in the definition of Sat_n. The partition is given by induction on a parameter of the (e,p)-decomposition $w_1, \ldots, w_m$, which we define now.

Index of an (e,p)-decomposition. Recall that $k_n = |M \times 2^{M^{n-1}}|$ and let $u \in A^*$ that admits an (e,p)-decomposition $u_1, \ldots, u_m$. Let $(f, \mathcal{F}) \in M \times 2^{M^{n-1}}$ be an idempotent and $j \leqslant m$; we say that $(f, \mathcal{F})$ can be *inserted* at position j if there exists $i \leqslant (k_n - 1)$ such that

$$g_n^p(u_{j-i}) \cdots g_n^p(u_j) \cdot (f, \mathcal{F}) = g_n^p(u_{j-i}) \cdots g_n^p(u_j).$$

The *index* of the (e,p)-decomposition $u_1, \ldots, u_m$ is the number of distinct idempotents $(f, \mathcal{F}) \in M \times 2^{|M|^{n-1}}$ that can be inserted at some position $j \leqslant m$. Observe that, by definition, the index of any (e,p)-decomposition is bounded by k_n.

Lemma 7.13. *Let $u \in A^*$ admitting an (e,p)-decomposition $u_1, \ldots, u_m$ of index g. Then, for all $\widehat{k}$ such that $\widehat{k} \geqslant g + 2k_n + p$, we have that $g_n^{\widehat{k}}(u) \in \mathrm{Sat}_n(\mathcal{D}_n)$.*

Before proving this lemma, we use it to conclude Case 2. We know that w admits an (e,p)-decomposition of index $g \leqslant k_n$. By definition, $k \geqslant 3k_n + p$; hence, it is immediate from Lemma 7.13 that $g_n^k(w) \in \mathrm{Sat}_n(\mathcal{D}_n)$. It now remains to prove Lemma 7.13.

Proof of Lemma 7.13. The proof goes by induction on the index g. When $m \leqslant k_n$, the result can be obtained from Lemma 7.8 by using $(m-1)$ times the argument that we used in Case 1. Assume now that $m > k_n$ and fix some number $\widehat{k} \geqslant g + 2k_n + p$. We have to show that $g_n^{\widehat{k}}(u) \in \mathrm{Sat}_n(\mathcal{D}_n)$. We rely on the following fact: □

Fact 7.14. *There exists a position $j \leqslant k_n$ and an idempotent $(e,\mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$ that can be inserted at position j.*

Proof. Since all $g_n^p(w_i)$ belong to the monoid $M \times 2^{M^{n-1}}$ whose size is k_n, it follows from the pigeonhole principle that there exist $j < j' \leqslant k_n + 1$ such that

$$g_n^p(w_1) \cdots g_n^p(w_j) = g_n^p(w_1) \cdots g_n^p(w_{j'}).$$

Hence, it suffices to take $(e,\mathcal{E}) = (g_n^p(w_{j+1}) \cdots g_n^p(w_{j'}))^\omega$. Note that $(e,\mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$ because of Item (c) in the definition of (e,p)-decompositions and Operation (Op_2) in the definition of Sat_n. □

Note that Fact 7.14 shows that if $g = 0$ (the base case of our induction), then we must have that $m \leqslant k_n$, a case that was already treated.

Denote by $j \leqslant k_n$ a position given by Fact 7.14, and let $\ell \leqslant m$ be the largest integer such that $(e,\mathcal{E})$ can be inserted at position ℓ. In particular, $j \leqslant \ell$. Using Lemma 7.8, we get that

$$g_n^{\widehat{k}}(u) \subseteq g_n^{\widehat{k}-1}(u_1 \cdots u_\ell) \cdot g_n^{\widehat{k}-1}(u_{\ell+1} \cdots u_m).$$

By definition, $u_{\ell+1}, \ldots, u_m$ is an (e,p)-decomposition and it has an index strictly smaller than that of $u_1, \ldots, u_m$ (by definition of ℓ, there is no position between $\ell+1$ and m at which $(e,\mathcal{E})$ can be inserted). Hence, it is immediate by induction hypothesis in Lemma 7.13 that

$$g_n^{\widehat{k}-1}(u_{\ell+1} \cdots u_m) \in \mathrm{Sat}_n(\mathcal{D}_n).$$

It now remains to prove that $g_n^{\widehat{k}-1}(u_1 \cdots u_\ell) \in \mathrm{Sat}_n(\mathcal{D}_n)$. The result will then follow from Operations (Op_1) and (Op_2) in the definition of Sat_n. We distinguish two cases depending on the distance between j and ℓ.

Case (a). Assume first that $\ell \leqslant j + k_n$. In that case, since $j \leqslant k_n$, we have that $\ell \leqslant 2k_n$. The result can then be obtained from Lemma 7.8 by using $\ell - 1$ times the same argument as the one that we used in Case 1.

Case (b). It remains to treat the case when $\ell > j + k_n$. This is where Operation (Op_3) in the definition of Sat_n is used. Consider the following:

$$\begin{array}{lcl} (e,\mathcal{R}) & = & g_n^p(u_1) \cdots g_n^p(u_j) \\ (e,\mathcal{T}') & = & g_n^p(u_{j+1} \cdots u_{j-k_n}) \cdot g_n^p(u_{\ell-(k_n-1)}) \cdots g_n^p(u_\ell). \end{array}$$

Note that we know from Item (c) in the definition of (e,p)-decompositions and Operation (Op_2) in the definition of Sat_n that $(e,\mathcal{R}) \in \mathrm{Sat}_n(\mathcal{D}_n)$. Moreover, using Fact 7.11, we obtain that $(e,\mathcal{T}') \subseteq (e,\mathcal{T})$.

Observe that $(\widehat{k}-1) - (j + k_n - 1) \geqslant p$. Hence, using $j + k_n - 1$ times the Decomposition Lemma (Lemma 7.8) and Lemma 7.7, we obtain that

$$g_n^{\widehat{k}-1}(u_1 \cdots u_\ell) \subseteq (e,\mathcal{R}) \cdot (e,\mathcal{T}').$$

By definition, $(e, \mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$ can be inserted at both positions j and ℓ. Hence, we have that

$$g_n^{\widehat{k-1}}(u_1 \cdots u_\ell) \subseteq (e, \mathcal{R}) \cdot (e, \mathcal{E}) \cdot (e, \mathcal{T}') \cdot (e, \mathcal{E}).$$

We now prove that $(e, \mathcal{E}) \cdot (e, \mathcal{T}') \cdot (e, \mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$. Since we already know that $(e, \mathcal{R}) \in \mathrm{Sat}_n(\mathcal{D}_n)$, it will then follow from Operations (Op_1) and (Op_2) that $g_n^{\widehat{k-1}}(u_1 \cdots u_\ell) \in \mathrm{Sat}_n(\mathcal{D}_n)$.

Since $(e, \mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$ and $\mathrm{alph}(e) = B$, it follows from Operation (Op_3) in the fixed-point procedure that:

$$(e, \mathcal{E}) \cdot (e, \mathcal{T}') \cdot (e, \mathcal{E}) \subseteq (e, \mathcal{E}) \cdot (e, \mathcal{T}) \cdot (e, \mathcal{E}) = (e, \mathcal{E}) \cdot (1_M, \mathcal{T}) \cdot (e, \mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n). \square$$

We conclude from Operation (Op_1) that $(e, \mathcal{E}) \cdot (e, \mathcal{T}') \cdot (e, \mathcal{E}) \in \mathrm{Sat}_n(\mathcal{D}_n)$, which terminates the proof.

8 DECIDABLE CHARACTERIZATION OF $\mathcal{B}\Sigma_2(<)$

In this section, we present our decidable characterization for $\mathcal{B}\Sigma_2(<)$. We already proved a (non-effective) characterization of $\mathcal{B}\Sigma_2(<)$ in Section 6 using the notion of *alternation*.

Recall that a chain $(s_1, \ldots, s_n) \in M^*$ has *alternation* ℓ if there are exactly ℓ indices i such that $s_i \neq s_{i+1}$. Recall also that a set of chains $\mathcal{S}$ has *bounded alternation* if there exists a bound $\ell \in \mathbb{N}$ such that all chains in $\mathcal{S}$ have alternation at most ℓ. We know by Corollary 6.7 that a regular language L is definable in $\mathcal{B}\Sigma_i(<)$ if and only if $C_i[\alpha]$ has bounded alternation with α as the syntactic morphism of L.

In this section, we prove that a third equivalent (effective) criterion can be given in the special case $i = 2$. This criterion is presented as an equation that needs to be satisfied by the alphabet completion of the syntactic morphism of the language. This equation is parametrized by junctures of length 2 through a relation that we now define.

Alternation Schema. Let $\alpha : A^* \to M$ be an alphabet-compatible monoid morphism. An *alternation schema* for α is a triple $(s, s_1, s_2) \in M^3$ such that there exist $(r_1, \mathcal{R}_1), (r_2, \mathcal{R}_2), (e, \mathcal{E}) \in \mathcal{J}_{2,2}[\alpha]$ with $(e, \mathcal{E})$ *idempotent*, and such that

- $\mathrm{alph}(e) = \mathrm{alph}(s)$.
- $s = r_1 e r_2$.
- $s_1 \in \mathcal{R}_1 \cdot \mathcal{E}$.
- $s_2 \in \mathcal{E} \cdot \mathcal{R}_2$.

Observe that the set of all alternation schemas for α can be computed from $\mathcal{J}_{2,2}[\alpha]$.

The purpose of alternation schemas is to abstract over M a property of words relatively to $\Sigma_2(<)$: if (s, s_1, s_2) is an alternation schema then, for all $k \in \mathbb{N}$, there exist $w, w_1, w_2 \in A^*$, mapped, respectively, to s, s_1, s_2 under α and such that, for all $u \in \mathrm{alph}(s)^*$, $w \lesssim_2^k w_1 u w_2$ (see Lemma 8.4 below).

We now have all the terminology we need to state our decidable characterization of $\mathcal{B}\Sigma_2(<)$.

Theorem 8.1. *Let L be a regular language and let $\alpha : A^* \to M$ be the alphabet completion of its syntactic morphism. The three following properties are equivalent:*

(1) L is definable in $\mathcal{B}\Sigma_2(<)$.
(2) $C_2[\alpha]$ has bounded alternation.
(3) α satisfies the following equation:

$$(s_1 t_1)^\omega s (t_2 s_2)^\omega = (s_1 t_1)^\omega s_1 t s_2 (t_2 s_2)^\omega \quad \text{for } (s, s_1, s_2) \text{ and } (t, t_1, t_2) \text{ alternation schemas such that } alph(s) = alph(t). \tag{16}$$

We know from Proposition 7.2 that all Σ_2-chains of length 3 and all alternation schemas associated to α can be computed. Hence, the third item of Theorem 8.1 can be decided and we get the desired corollary.

Corollary 8.2. *Given as input a regular language L, it is decidable to test whether L is definable in $\mathcal{B}\Sigma_2(<)$.*

Note that the characterization of $\mathcal{B}\Sigma_2(<)$ that we present in Theorem 8.1 is different from the one presented in the conference version of this article [81]. The characterization of [81] was relying on three equations, Equation (16) and the following two equations, which are parametrized by Σ_2-chains of length 3.

$$\begin{array}{rcl} s_1^\omega s_3^\omega & = & s_1^\omega s_2 s_3^\omega \\ s_3^\omega s_1^\omega & = & s_3^\omega s_2 s_1^\omega \end{array} \quad \text{for all } (s_1, s_2, s_3) \in C_2[\alpha]. \tag{17}$$

It turns out that Equation (17) is actually a consequence of Equation (16). We state this property in the next lemma.

Lemma 8.3. *Let $\alpha : A^* \to M$ be an alphabet-compatible morphism into a finite monoid M. If α satisfies Equation (16), then α satisfies Equation (17) as well.*

Proof. Assume that α satisfies Equation (16) and let $(s_1, s_2, s_3) \in C_2[\alpha]$. We have to prove that $s_1^\omega s_3^\omega = s_1^\omega s_2 s_3^\omega$ and $s_3^\omega s_1^\omega = s_3^\omega s_2 s_1^\omega$. We prove only the first equality, since the other one is obtained symmetrically.

We claim that $(s_1^\omega, s_1^\omega, s_3^\omega)$ and $(s_1^\omega s_2, s_1^\omega, s_3^\omega)$ are alternation schemas. Assuming this claim, note that since (s_1, s_2, s_3) is a Σ_2-chain, we have in particular $\mathsf{alph}(s_1) = \mathsf{alph}(s_2) = \mathsf{alph}(s_3)$, whence $\mathsf{alph}(s_1^\omega) = \mathsf{alph}(s_1^\omega s_2)$. It follows from Equation (16) that

$$(s_1^\omega s_1^\omega)^\omega s_1^\omega (s_3^\omega s_3^\omega)^\omega = (s_1^\omega s_1^\omega)^\omega s_1^\omega s_1^\omega s_2 s_3^\omega (s_3^\omega s_3^\omega)^\omega.$$

This exactly says that $s_1^\omega s_3^\omega = s_1^\omega s_2 s_3^\omega$, as desired.

Let us now prove the claim. Observe that since $(s_1, s_2, s_3) \in C_2[\alpha]$, it is immediate from the definition of Σ_2-chains and Σ_2-junctures that $(s_1, \{s_1, s_3\}) \in \mathcal{J}_{2,2}[\alpha]$ and $(s_2, \{s_3\}) \in \mathcal{J}_{2,2}[\alpha]$. We begin by proving that $(s_1^\omega, s_1^\omega, s_3^\omega)$ is an alternation schema. Let $(e, \mathcal{E}) = (s_1, \{s_1, s_3\})^\omega \in \mathcal{J}_{2,2}[\alpha]$ and $(r_1, \mathcal{R}_1) = (r_2, \mathcal{R}_2) = (1_M, \{1_M\}) \in \mathcal{J}_{2,2}[\alpha]$. By definition, $s_1^\omega = e = r_1 e r_2$, $s_1^\omega \in \mathcal{E} = \mathcal{R}_1 \mathcal{E}$, and $s_3^\omega \in \mathcal{E} = \mathcal{E}\mathcal{R}_2$. It follows that $(s_1^\omega, s_1^\omega, s_3^\omega)$ is an alternation schema.

Finally, we prove that $(s_1^\omega s_2, s_1^\omega, s_3^\omega)$ is also an alternation schema. Again, let $(e, \mathcal{E}) = (s_1, \{s_1, s_3\})^\omega \in \mathcal{J}_{2,2}[\alpha]$. Recall our algorithm for computing $\mathcal{J}_{2,2}[\alpha]$ (see Section 7). Since $\mathsf{alph}(s_3) = \mathsf{alph}(s_1) = \mathsf{alph}(e)$, we know from Operation (Op_3) in the algorithm that

$$(e, \mathcal{E}) \cdot (1_M, (s_3)^{\omega-1}) \cdot (e, \mathcal{E}) \in \mathcal{J}_{2,2}[\alpha].$$

By closure under downset, it follows that $((s_1)^\omega, \{(s_3)^{2\omega-1}\}) \in \mathcal{J}_{2,2}[\alpha]$. We define $(r_1, \mathcal{R}_1) = (1_M, \{1_M\}) \in \mathcal{J}_{2,2}[\alpha]$ and $(r_2, \mathcal{R}_2) = ((s_1)^\omega, \{(s_3)^{2\omega-1}\}) \cdot (s_2, \{s_3\}) = ((s_1)^\omega s_2, \{(s_3)^\omega\}) \in \mathcal{J}_{2,2}[\alpha]$. By definition, $(s_1)^\omega s_2 = r_1 e r_2$, $(s_1)^\omega \in \mathcal{R}_1\mathcal{E}$, and $(s_3)^\omega \in \mathcal{E}\mathcal{R}_2$. Finally, $\mathsf{alph}(e) = \mathsf{alph}(s_1^\omega s_2)$. It follows that $(s_1^\omega s_2, s_1^\omega, s_3^\omega)$ is an alternation schema. □

Note that we still use Equation (17) in the proof of Theorem 8.1, as it will be more convenient to use it instead of Equation (16) in some places.

Another important remark is that there are similarities between Theorem 8.1 and a theorem of Bojańczyk and the first author [16] that states the decidable characterization of an entirely different formalism: Boolean combination of open sets of infinite trees. In [16] as well, the authors present a notion of "chains" tailored to their formalism (although they do not make the link with

separation). This is not surprising, as the notion of chain is quite generic for formalisms defined by Boolean combinations and what is specific is the algorithms computing them.

A more surprising fact is that our equations are very similar to the ones stated in [16]. Despite this fact, since the formalisms are of a different nature, the way the chains of [16] and the way our Σ_2-chains are constructed are completely independent. This also means that the proofs are also mostly independent. However, we do reuse several combinatorial arguments of [16] at the end of the proof. One could say that the proofs are both (very different) setups to apply similar combinatorial arguments in the end.

It now remains to prove Theorem 8.1. Observe that we already know from Corollary 6.7 and Lemma 7.1 that $1 \Leftrightarrow 2$. To conclude the proof, we shall show that $1 \Rightarrow 3$ and $3 \Rightarrow 2$. The direction $3 \Rightarrow 2$ is the most involved proof of this article. We devote three sections to this proof. In Section 9, we define a key object for this proof: *chain trees*. We then use this object to reduce the proof to two independent propositions that are then proved in Sections 10 and 11.

We finish this section with the much easier $1 \Rightarrow 3$ direction. Assume that L is a $\mathcal{B}\Sigma_2(<)$-definable language and let $\alpha : A^* \to M$ be the alphabet completion of its syntactic morphism. We prove that α satisfies Equation (16). This is an Ehrenfeucht-Fraïssé argument. We begin with a lemma on alternation schemas, which formalizes the property that we sketched above.

Lemma 8.4. *Assume that (s, s_1, s_2) is an alternation schema. Then, for all $k \in \mathbb{N}$, there exist $w, w_1, w_2 \in A^*$ such that:*

- *$\alpha(w) = s, \alpha(w_1) = s_1$ and $\alpha(w_2) = s_2$.*
- *for all $u \in alph(s)^*$, $w \lesssim_2^k w_1 u w_2$.*

Proof. This is proved using Lemma 4.8. Fix an alternation schema (s, s_1, s_2) and $k \in \mathbb{N}$. Let $(r_1, \mathcal{R}_1), (r_2, \mathcal{R}_2), (e, \mathcal{E}) \in \mathcal{J}_{2,2}[\alpha]$ be as in the definition of alternation schemas.

Let $h = 2^{2k}$. Since $(e, \mathcal{E})$ is idempotent, we have that $(e, \mathcal{E})^h = (e, \mathcal{E})$. By definition of Σ_2-junctures, we obtain words $v_1, v_1', v_2, v_2', x, x_1', x_2' \in A^*$ satisfying the following properties:

a) $\alpha(v_1) = r_1$, $\alpha(v_2) = r_2$, $\alpha(x) = e$, $\alpha(v_1' x_1') = s_1$, $\alpha(x_2' v_2') = s_2$.
b) $v_1 x^h \lesssim_2^k v_1' x_1'$ and $x^h v_2 \lesssim_2^k x_2' v_2'$.

Let $w = v_1 x^{2h} v_2$, $w_1 = v_1' x_1'$ and $w_2 = x_2' v_2'$. It follows from Item (a) that $\alpha(w) = r_1 e r_2 = s$, $\alpha(w_1) = s_1$ and $\alpha(w_2) = s_2$. Finally, since α is alphabet compatible, we have that $\mathrm{alph}(x) = \mathrm{alph}(e)$ and by definition of alternation schemas, $\mathrm{alph}(e) = \mathrm{alph}(s)$. Therefore, it is immediate using Ehrenfeucht-Fraïssé games that, for any word $u \in \mathrm{alph}(s)^*$, $u \lesssim_1^k x^h$. It then follows from Lemma 4.8 that $x^{2h} \lesssim_2^k x^h u x^h$, whence by Lemma 4.6, that $w \lesssim_2^k v_1 x^h u x^h v_2$. Finally, using Item (b), we conclude that $w \lesssim_2^k w_1 u w_2$. □

We can now use Lemma 8.4 to prove that α satisfies Equation (16). Let (s, s_1, s_2) and (t, t_1, t_2) be alternation schemas such that $\mathrm{alph}(s) = \mathrm{alph}(t)$. Let $w, w_1, w_2 \in A^*$ of images s, s_2, s_2 and $z, z_1, z_2 \in A^*$ of images t, t_1, t_2 satisfying the conditions of Lemma 8.4. We prove that, for any $u, v \in A^*$,

$$u[(z_1 w_1)^N z (w_2 z_2)^N] v \quad \cong_2^k \quad u[(z_1 w_1)^N z_1 w z_2 (w_2 z_2)^N] v, \tag{18}$$

where again $N = 2^k \omega$. By definition of the alphabet completion of the syntactic monoid, of the alphabetic conditions and since L is defined by a $\mathcal{B}\Sigma_2(<)$ formula of rank k, Equation (16) will follow. Since $\mathrm{alph}(s) = \mathrm{alph}(t)$, the words w, w_1, w_2 and z, z_1, z_2 given by Lemma 8.4 satisfy

$$z \lesssim_2^k z_1 w z_2, \tag{19}$$

$$w \lesssim_2^k w_1 z w_2. \tag{20}$$

Using Lemma 4.6, we may multiply Equation (19) by $u(z_1w_1)^N$ on the left and by $(w_2z_2)^N v$ on the right:

$$u(z_1w_1)^N z(w_2z_2)^N v \quad \lesssim_2^k \quad u(z_1w_1)^N z_1wz_2(w_2z_2)^N v.$$

For the converse direction, from Lemma 4.7, we have that $(z_1w_1)^N \lesssim_2^k (z_1w_1)^{N-1}$ and $(w_2z_2)^N \lesssim_2^k (w_2z_2)^{N-1}$. Using Equation (20) and Lemma 4.6 again, we conclude that

$$u(z_1w_1)^N z_1wz_2(w_2z_2)^N v \quad \lesssim_2^k \quad u(z_1w_1)^{N-1} z_1(w_1zw_2)z_2(w_2z_2)^{N-1} v,$$

i.e.,

$$u(z_1w_1)^N z_1wz_2(w_2z_2)^N v \quad \lesssim_2^k \quad u(z_1w_1)^N z(w_2z_2)^N v.$$

9 PROOF OF THEOREM 8.1: CHAIN TREES

In this section, we begin the proof of the difficult direction of Theorem 8.1. Given an alphabet-compatible morphism $\alpha : A^* \to M$, we prove that if Equation (16) is satisfied, then $C_2[\alpha]$ has bounded alternation. More precisely, we prove the contrapositive: if $C_2[\alpha]$ has unbounded alternation, then the equation is not satisfied.

To prove this, we rely on a new notion: "*chain trees*." Chain trees are a means to analyze how Σ_2-chains with high alternation are built. In particular, we will use them at the end of the section to decide which equation is contradicted. Intuitively, a chain tree is associated to a single Σ_2-chain and represents a computation of our least fixed-point algorithm of Section 7 that generates this Σ_2-chain.

As we explained in the previous section, one can find connections between our proof and that of the decidable characterization of Boolean combination of open sets of trees [16]. In [16] as well, the authors consider a notion of "chains," which corresponds to open sets of trees and analyze how they are built. This is achieved with an object called a "Strategy Tree." Though strategy trees and chain trees share the same purpose, i.e., analyzing how chains are built, there is no connection between the notions themselves since they deal with completely different objects.

We organize the section in two subsections. First, we define the general notion of chain trees. Then, we use chain trees to reduce the proof of Theorem 8.1 to two independent propositions (we will then prove these two propositions in Sections 10 and 11).

9.1 Definition

Let $\alpha : A^* \to M$ be an alphabet-compatible morphism into a finite monoid M. We associate to α a set $\mathbb{T}_C[\alpha]$ of *chain trees*. As we explained, a chain tree is associated to a single Σ_2-chain for α and represents a way to compute this Σ_2-chain using our least fixed-point algorithm. However, recall that this algorithm does not work directly with chains but rather with the more general notion of junctures. For this reason, we actually define two notions:

(1) The set $\mathbb{T}_{\mathcal{J}}[\alpha]$ of *juncture trees* associated to α. Each tree in $\mathbb{T}_{\mathcal{J}}[\alpha]$ represents an actual computation of the least fixed-point algorithm. Hence, we can associate the result of this computation to the tree: this Σ_2-juncture is called the *juncture value* of the tree.
(2) The set $\mathbb{T}_C[\alpha]$ of *chain trees*. Each tree in $\mathbb{T}_C[\alpha]$ instantiates a juncture tree of $\mathbb{T}_{\mathcal{J}}[\alpha]$ and is associated to a specific Σ_2-chain that belongs to its juncture value. This Σ_2-chain is called the *chain value* of the chain tree.

Juncture Trees. For any $n \geqslant 1$, a *juncture tree T of level n* for α is an ordered unranked tree that may have four types of nodes: product nodes, operation nodes, initial leaves, and ports. To each node that is not a port, we associate a *juncture value*, $\mathrm{val}_{\mathcal{J}}(x) \in M \times 2^{M^{n-1}}$, by induction on the structure of the tree. Intuitively, each type of node corresponds to a stage of the least fixed-point

algorithm while constructing the juncture value of the tree. We now give a precise definition of each type of node.

- *Initial Leaves.* An initial leaf x is labeled with a constant Σ_2-chain $(s, \ldots, s) \in C_{2,n}[\alpha]$. We let $\mathrm{val}_{\mathcal{J}}(x) = (s, \{(s, \ldots, s)\}) \in \mathcal{J}_{2,n}[\alpha]$. Initial leaves correspond to the set $\mathcal{D}_n$ of trivial Σ_2-junctures, which serves to initialize the least fixed-point algorithm when it starts.
- *Ports.* A port is an unlabeled leaf whose parent has to be an operation node. In particular, a port may never be the root of the tree. Ports have no juncture value and are simply placeholders that get replaced by true leaves when the juncture tree is instantiated into a chain tree (see below).
- *Product Nodes.* A product node x is unlabeled. It has exactly two children, x_1 and x_2, which may be of any node type except "port." We let $\mathrm{val}_{\mathcal{J}}(x) = \mathrm{val}_{\mathcal{J}}(x_1) \cdot \mathrm{val}_{\mathcal{J}}(x_2)$. Product nodes correspond to Operation (Op_2) in the fixed-point algorithm.
- *Operation Nodes.* An operation node x is unlabeled and has exactly 3 children x_1, x_2, and x_3 from left to right. The middle child x_2 has to be a port. The left and right children x_1 and x_3 may be of any node type except "port." However, the trees rooted in x_1 and x_3 must be *identical*. Moreover, we require $\mathrm{val}_{\mathcal{J}}(x_1) = \mathrm{val}_{\mathcal{J}}(x_3)$ to be an *idempotent* $(e, \mathcal{E})$ of $M \times 2^{M^{n-1}}$. Finally, we let the juncture value of the operation node x as $\mathrm{val}_{\mathcal{J}}(x) = (e, \mathcal{E}) \cdot (1_M, \mathcal{T}) \cdot (e, \mathcal{E})$ with $\mathcal{T} = \{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathrm{alph}(t_1) = \mathrm{alph}(e)\}$. Operation nodes and ports correspond to Operation (Op_3) in the fixed-point algorithm.

We denote by $\mathbb{T}_{\mathcal{J}}[\alpha]$ the set of juncture trees that can be associated to α. If $T \in \mathbb{T}_{\mathcal{J}}[\alpha]$, we denote by $\mathrm{val}_{\mathcal{J}}(T)$ the juncture value of the root of T. An example of juncture tree is given in Figure 6. The following proposition is essentially an alternate statement of Proposition 7.2.

Proposition 9.1. *Let $n \geqslant 1$. Then,*

$$\begin{aligned} \mathcal{J}_{2,n}[\alpha] &= \downarrow\left\{val_{\mathcal{J}}(T) \mid T \in \mathbb{T}_{\mathcal{J}}[\alpha] \text{ with level } n\right\}, \\ \mathcal{J}_{2}[\alpha] &= \downarrow\left\{val_{\mathcal{J}}(T) \mid T \in \mathbb{T}_{\mathcal{J}}[\alpha]\right\}. \end{aligned}$$

Proof. Immediate from Proposition 7.2. □

Chain Trees. Chain trees are obtained by instantiating juncture trees. Let T be a juncture tree and let n be its level. An *instantiation* of T is a new tree T' which is obtained from T by replacing all ports with new *operation leaves*.

An operation leaf x is labeled with a chain of length n. Moreover, this chain has to satisfy an additional condition with respect to its parent. Observe first that since ports carry no information in juncture trees, $\mathrm{val}_{\mathcal{J}}(t)$ remains well defined for any node t of T' that is not a new operation leaf. Since x replaces a port, its parent z has to be an operation node. We ask the label of x to be chosen in $\mathrm{val}_{\mathcal{J}}(z)$, i.e., in the juncture $(e, \mathcal{E}) \cdot (1_M, \mathcal{T}) \cdot (e, \mathcal{E})$, where $(e, \mathcal{E})$ is the (idempotent) juncture value shared by the left and right children of z and $\mathcal{T} = \left\{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathrm{alph}(t_1) = \mathrm{alph}(e)\right\}$. Note that by Proposition 9.1, this means that the label of x belongs to $C_{2,n}[\alpha]$.

For every juncture tree T, we denote by $\|T\|$ the set of instantiations of T (see Figure 7 for an example). The set $\mathbb{T}_C[\alpha]$ of *chain trees* associated to α is the set $\bigcup_{T \in \mathbb{T}_{\mathcal{J}}[\alpha]} \|T\|$. Finally, if $R \in \mathbb{T}_C[\alpha]$ is of level n, to every node x of R, we associate a second value $\mathrm{val}_C(x) \in M^n$, called the *chain value* of x. If x is an initial (resp., operation) leaf, $\mathrm{val}_C(x)$ is simply the label of x. If x is a product node with children x_1 and x_2, then $\mathrm{val}_C(x) = \mathrm{val}_C(x_1) \cdot \mathrm{val}_C(x_2)$. Finally, if x is an operation node with children x_1, x_2, and x_3, then $\mathrm{val}_C(x) = \mathrm{val}_C(x_1) \cdot \mathrm{val}_C(x_2) \cdot \mathrm{val}_C(x_3)$. We let $\mathrm{val}_C(R)$ be the chain value of the root of R. The following facts are immediate from the definitions.

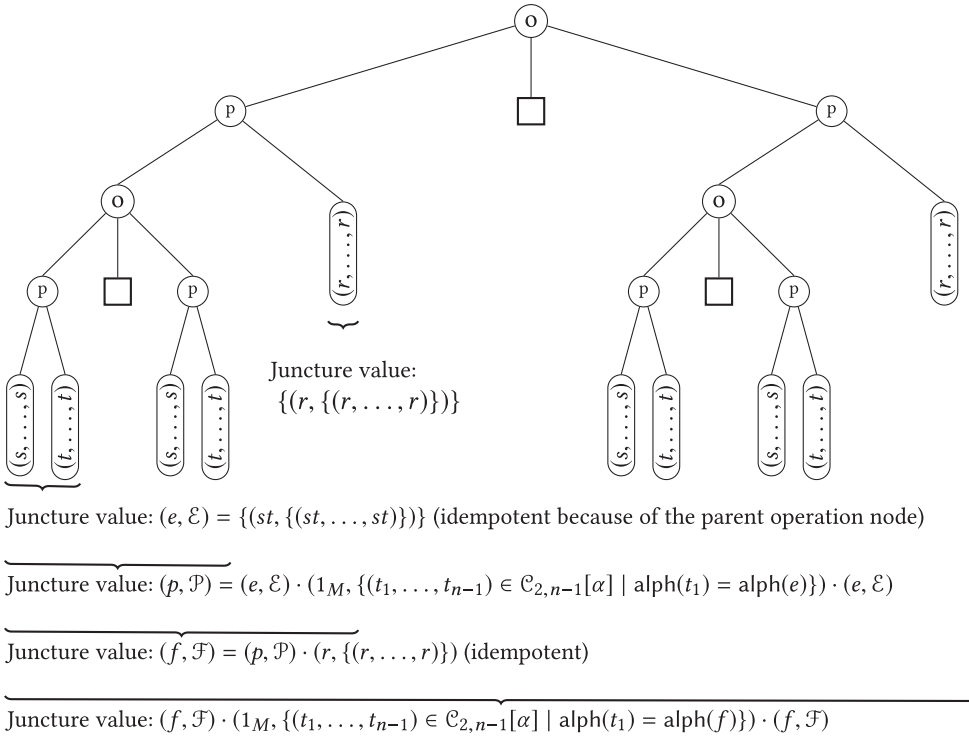


Fig. 6. An example of juncture tree of level n.

Fact 9.2. *Let $T \in \mathbb{T}_C[\alpha]$ and let $x_1, \ldots, x_m$ be its leaves listed from left to right. Then, $val_C(T) = val_C(x_1) \cdots val_C(x_m)$.*

Fact 9.3. *Let $T \in \mathbb{T}_C[\alpha]$ of level n and let x be a node of T. Then, $val_C(x) \in C_{2,n}[\alpha]$.*

We now prove that the definition of chain trees matches our purpose, i.e., that the set of Σ_2-chains is exactly the set of values of trees in $\mathbb{T}_C[\alpha]$. This is a corollary of the following proposition.

Proposition 9.4. *Let $T \in \mathbb{T}_{\mathcal{J}}[\alpha]$. Then,*

$$val_{\mathcal{J}}(T) = \left\{ val_C(T') \mid T' \in \llbracket T \rrbracket \right\}. \tag{21}$$

Proof. Before proving the statement, note that $\mathrm{val}_{\mathcal{J}}(T)$ is a juncture, while the right member of Equation (21) is a set of chains. To simplify the notation, we identify in this proof the juncture $\mathrm{val}_{\mathcal{J}}(T)$ in Equation (21) with the set of chains that it contains, i.e., $\{(s, \bar{s}) \mid \{(s, \{\bar{s}\})\} \subseteq \mathrm{val}_{\mathcal{J}}(T)\}$.

We proceed by induction on the structure of T (which is shared with any chain tree $T' \in \llbracket T \rrbracket$). If T is a single initial leaf, then $\llbracket T \rrbracket = \{T\}$ since there is no port in T to replace, and the result is by definition. Otherwise, let x be the root of T.

If x is a product node, then let T_1 and T_2 be the subtrees rooted at its children. By induction hypothesis, we have that $\mathrm{val}_{\mathcal{J}}(T_1) = \{\mathrm{val}_C(T_1') \mid T_1' \in \llbracket T_1 \rrbracket\}$ and $\mathrm{val}_{\mathcal{J}}(T_2) = \{\mathrm{val}_C(T_2') \mid T_2' \in \llbracket T_2 \rrbracket\}$.

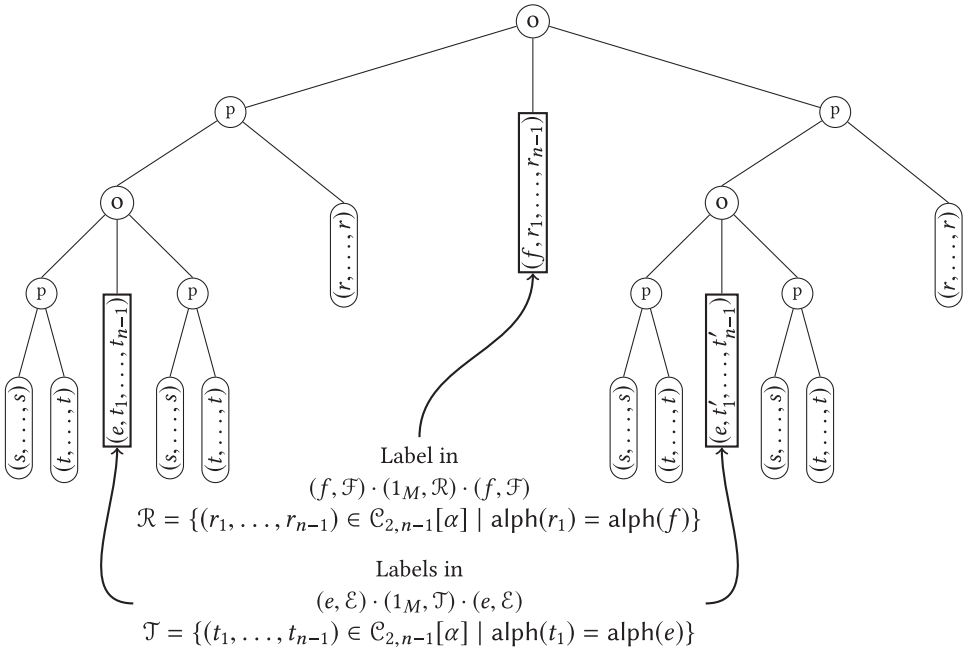


Fig. 7. An instantiation of the juncture tree in Figure 6.

By definition, $\mathrm{val}_{\mathcal{J}}(T)=\mathrm{val}_{\mathcal{J}}(T_1)\cdot\mathrm{val}_{\mathcal{J}}(T_2)$ and

$$\left\{\mathrm{val}_C(T')\mid T'\in[\![T]\!]\right\}=\left\{\mathrm{val}_C(T'_1)\cdot\mathrm{val}_C(T'_2)\mid T'_1\in[\![T_1]\!]\text{ and }T'_2\in[\![T_2]\!]\right\},$$

which terminates this case.

If x is an operation node, let R be the single juncture tree that is rooted in both its left and right children and let $(e,\mathcal{E})=\mathrm{val}_{\mathcal{J}}(R)$. By definition, $\mathrm{val}_{\mathcal{J}}(T)=(e,\mathcal{E})\cdot(1_M,\mathcal{T})\cdot(e,\mathcal{E})$ with $\mathcal{T}=\{(t_1,\ldots,t_{n-1})\in C_{2,n-1}[\alpha]\mid \mathsf{alph}(t_1)=\mathsf{alph}(e)\}$. Moreover, since $(e,\mathcal{E})$ is idempotent by definition, we have that

$$\mathrm{val}_{\mathcal{J}}(T)=(e,\mathcal{E})\cdot(e,\mathcal{E})\cdot(1_M,\mathcal{T})\cdot(e,\mathcal{E})\cdot(e,\mathcal{E}).$$

This terminates the proof since the set of values that can be given to an operation leaf replacing the port child of x is exactly $(e,\mathcal{E})\cdot(1_M,T)\cdot(e,\mathcal{E})$ and, by induction hypothesis, $(e,\mathcal{E})=\mathrm{val}_{\mathcal{J}}(R)=\{\mathrm{val}_C(R')\mid R'\in[\![R]\!]\}$. □

The following corollary states that the set of Σ_2-chains is exactly the set of chain values of chain trees and is immediate from Proposition 9.1 and Proposition 9.4.

Corollary 9.5. *Let $B\subseteq A$, $n\in\mathbb{N}$. Then,*

$$C_{2,n}[\alpha]=\left\{val_C(T)\mid T\in\mathbb{T}_C[\alpha]\text{ with level }n\right\},$$
$$C_2[\alpha]\quad=\left\{val_C(T)\mid T\in\mathbb{T}_C[\alpha]\right\}.$$

Alternation and Recursive Alternation of a Chain Tree. The *alternation* of a chain tree is the alternation of its chain value. We say that a set of chain trees $\mathbb{S}$ has *unbounded alternation* if the set $\{\mathrm{val}_C(T)\mid T\in\mathbb{S}\}$ has unbounded alternation. Note that, by Proposition 9.4, $C_2[\alpha]$ has unbounded alternation if and only if $\mathbb{T}_C[\alpha]$ has unbounded alternation.

In the proof, we will be interested in another property of chain trees: *recursive alternation*. Recursive alternation corresponds to the maximal alternation of labels at operation leaves in the tree.

More precisely, if T is a chain tree, its *recursive alternation* is the largest integer j such that there exists an *operation leaf* in T whose label has alternation j. An important idea in the proof is to separate the case when we can find a set of chain trees with unbounded alternation but bounded recursive alternation from the converse one. This is what we do in the following section.

9.2 Applying Chain Trees to Theorem 8.1

We prove that $3 \Rightarrow 2$ in Theorem 8.1. Let $\alpha : A^* \to M$ be an alphabet-compatible morphism into a finite monoid M. We have to prove that if α satisfies Equation (16), then $C_2[\alpha]$ has bounded alternation.

The proof is by contrapositive. We assume that $C_2[\alpha]$ has unbounded alternation and prove that α does not satisfy the equation. Using chain trees, we separate the argument into two independent cases. These two cases are stated in the following propositions.

Proposition 9.6. *Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation but bounded recursive alternation. Then, α does not satisfy both equations in Equation (17).*

Proposition 9.7. *Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation and that all such sets have unbounded recursive alternation. Then, α does not satisfy Equation (16).*

Proposition 9.6 and Proposition 9.7 are both involved and proved in Section 11 and Section 10, respectively. We finish this section by using them to conclude the proof of Theorem 8.1.

By hypothesis, $C_2[\alpha]$ has unbounded alternation. Hence, it follows from Corollary 9.5 that $\mathbb{T}_C[\alpha]$ also has unbounded alternation. Therefore, there exists at least one set of chain trees $\mathbb{S}$ with unbounded alternation. If $\mathbb{S}$ can be chosen with bounded recursive alternation, it follows from Proposition 9.6 that there is a contradiction to one of the equations in Equation (17) and therefore to Equation (16) by Lemma 8.3. Otherwise, there is a contradiction to Equation (16) by Proposition 9.7, which terminates the proof.

10 PROOF OF PROPOSITION 9.7

In this section, we prove Proposition 9.7. Recall that we have fixed an alphabet-compatible morphism $\alpha : A^* \to M$ into a finite monoid M. Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation and that all such sets have unbounded recursive alternation. We need to prove that α does not satisfy Equation (16).

We rely on a new object that is specific to this case, the *chain graph*. A chain graph describes a construction process for a subset of the set of Σ_2-chains for α. While this subset may not be the whole set of Σ_2-chains for α, we will prove that, under the hypothesis of Proposition 9.7, it is sufficient to derive a contradiction to Equation (16).

The Chain Graph. We define a directed graph $G[\alpha] = (V, E)$ whose edges are unlabeled ($E \subseteq V \times V$). We call $G[\alpha]$ the *chain graph* of α. The set V of nodes of $G[\alpha]$ is the set $V = M^2 \times M$. We now define the set E of edges of $G[\alpha]$. Let $((p_1, p_2), s)$ and $((q_1, q_2), t)$ be two nodes of $G[\alpha]$; then, E contains an edge from $((p_1, p_2), s)$ to $((q_1, q_2), t)$ if there exist $s_1, s_2 \in M$ such that $(s, s_1, s_2) \in M^3$ is an alternation schema, $p_1 s_1 = q_1$, and $s_2 p_2 = q_2$. Observe that this definition does not depend on t.

Define the *value* of a node $((p_1, p_2), s)$ as $p_1 s p_2$ and its *alphabet* as $\mathrm{alph}(s)$ (recall that α is alphabet compatible).

We say that $G[\alpha]$ is *recursive* if it contains a cycle such that

a) all nodes in the cycle have the same alphabet, and
b) the cycle contains two nodes with different values.

Such a cycle is called *productive*. We now prove Proposition 9.7 as a consequence of the two following propositions.

Proposition 10.1. *Assume that $G[\alpha]$ is recursive. Then, α does not satisfy Equation (16).*

Proposition 10.2. *Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation and that all such sets have unbounded recursive alternation. Then, $G[\alpha]$ is recursive.*

Observe that Proposition 9.7 is an immediate consequence of Propositions 10.1 and 10.2. Before proving them, note that the notion of chain graph is inspired from the notion of strategy graph in [16]. This is because both notions are designed to derive contradictions to similar equations. However, our proof remains fairly different from the one of [16]. The reason for this is that the main difficulty here is proving Proposition 10.2, i.e., going from chain trees (which are unique to our setting) to a recursive chain graph. On the contrary, the much simpler proof of Proposition 10.1 is similar to the corresponding one in [16].

10.1 Proof of Proposition 10.1

Assume that $G[\alpha]$ is recursive. By definition, we get a productive cycle in the graph $G[\alpha]$. We first prove that we may assume this cycle to consist exactly of two nodes.

Lemma 10.3. *If $G[\alpha]$ is recursive, it has a productive cycle with exactly two nodes.*

Proof. Since $G[\alpha]$ is recursive, by definition it contains a productive cycle, i.e., a cycle whose nodes all share the same alphabet and contain two nodes with different values. In particular, the number n of nodes in the cycle is at least 2. If $n = 2$, the lemma is immediate. Assume that $n \geqslant 3$; we prove that $G[\alpha]$ must contain a productive cycle of length $n - 1$. The lemma will then follow by induction.

To construct such a productive cycle of length $n - 1$, it suffices to show that one can replace any two consecutive nodes

$$((u_1, u_2), r) \to ((p_1, p_2), s)$$

in the cycle by a single one having the same value as $((p_1, p_2), s)$. Indeed, since the cycle is of length at least 3, there exists such an edge, where $((p_1, p_2), s)$ is one of the two nodes having distinct values and the other one is not $((u_1, u_2), r)$, meaning that the resulting shortened cycle will still exhibit two nodes with distinct values.

Pick such an edge in the cycle; by definition, there exists an alternation schema (r, r_1, r_2) such that $u_1 r_1 = p_1$ and $r_2 u_2 = p_2$. Consider the node $((u_1, u_2), r_1 s r_2)$.

- By definition of an alternation schema and of a productive cycle, $\mathrm{alph}(r_1 s r_2) = \mathrm{alph}(rs) = \mathrm{alph}(s)$; hence, the node $((u_1, u_2), r_1 s r_2)$ has the same alphabet as all nodes in the cycle.
- Its value is $u_1(r_1 s r_2)u_2 = p_1 s p_2$; hence, $((u_1, u_2), r_1 s r_2)$ has the same value as $((p_1, p_2), s)$.
- By definition of the graph, any node having an outgoing edge to $((u_1, u_2), r)$ also has an outgoing edge to $((u_1, u_2), r_1 s r_2)$.
- It remains to show that if there is an edge $((p_1, p_2), s) \to ((q_1, q_2), t)$, then there is also an edge $((u_1, u_2), r_1 s r_2) \to ((q_1, q_2), t)$.

Let (s, s_1, s_2) be an alternation schema such that $p_1 s_1 = q_1$ and $s_2 p_2 = q_2$ (such an alternation schema exists by definition of the edges). One can verify that $(r_1 s r_2, r_1 s_1, s_2 r_2)$ is an alternation schema as well. Moreover, $u_1 r_1 s_1 = p_1 s_1 = q_1$ and $s_2 r_2 u_2 = s_2 p_2 = q_2$, which proves that there is an edge from $((u_1, u_2), r_1 s r_2)$ to $((q_1, q_2), t)$. □

We now conclude the proof of Proposition 10.1: we have to show that α fails Equation (16). Let $((p_1, p_2), s)$ and $((q_1, q_2), t)$ be two nodes forming a productive cycle of length 2, as defined in Lemma 10.3. We get alternation schemas (s, s_1, s_2) and (t, t_1, t_2) such that

(1) $p_1sp_2 \neq q_1tq_2$.
(2) $\mathrm{alph}(s) = \mathrm{alph}(t)$.
(3) $p_1s_1 = q_1$ and $q_1t_1 = p_1$; hence, $p_1 = p_1(s_1t_1)^\omega$.
(4) $s_2p_2 = q_2$ and $t_2q_2 = p_2$; hence, $p_2 = (t_2s_2)^\omega p_2$.

By combining Items (3) and (4), we obtain that

$$\begin{aligned} p_1sp_2 &= p_1(s_1t_1)^\omega s(t_2s_2)^\omega p_2 \\ q_1tq_2 &= p_1(s_1t_1)^\omega s_1ts_2(t_2s_2)^\omega p_2. \end{aligned}$$

Hence, since $\mathrm{alph}(s) = \mathrm{alph}(t)$, Equation (16) would require that $p_1sp_2 = q_1tq_2$, which contradicts Item (1). We conclude that Equation (16) is not satisfied by α.

10.2 Proof of Proposition 10.2

In the remainder of the section, we assume that α satisfies the hypothesis of Proposition 10.2. We prove that $G[\alpha]$ is recursive by constructing a productive cycle.

We say that a node $((p_1, p_2), s)$ of $G[\alpha]$ is *alternating* if for all n, there exists $(s_1, \ldots, s_n) \in C_{2,n}[\alpha]$ such that $s_1 = s$ and the chain $(p_1s_1p_2, \ldots, p_1s_np_2)$ has alternation $n - 1$.

Lemma 10.4. *$G[\alpha]$ contains at least one alternating node.*

Proof. By hypothesis, $C_2[\alpha]$ has unbounded alternation. It follows that there exists a least one $s \in M$ such that there are Σ_2-chains with arbitrary high alternation and s as first element. By definition, the node $((1_M, 1_M), s)$ is then alternating. □

For the remainder of the proof, we define B as a minimal alphabet such that there exists an alternating node $((p_1, p_2), s)$ in $G[\alpha]$ with $\mathrm{alph}(s) = B$. By this, we mean that the only $C \subseteq B$ such that there exists an alternating node $((q_1, q_2), t)$ in $G[\alpha]$ with $\mathrm{alph}(t) = C$ is B itself.

Lemma 10.5. *Let $((p_1, p_2), s)$ be an alternating node of $G[\alpha]$ with $\mathrm{alph}(s) = B$. Then, there exists an alternating node $((q_1, q_2), t)$ such that*

(1) $\mathrm{alph}(t) = B$.
(2) there exists an edge from $((p_1, p_2), s)$ to $((q_1, q_2), t)$.
(3) $p_1sp_2 \neq q_1tq_2$.

By definition, $G[\alpha]$ has finitely many nodes. Therefore, since by Lemma 10.4 there exists at least one alternating node, it is immediate from Lemma 10.5 that $G[\alpha]$ must contain a cycle witnessing that $G[\alpha]$ is recursive. This terminates the proof of Proposition 10.2. It remains to prove Lemma 10.5. We present the proof in the next section.

10.3 Proof of Lemma 10.5

Let $((p_1, p_2), s)$ be an alternating node of $G[\alpha]$ with $\mathrm{alph}(s) = B$. We need to construct a node $((q_1, q_2), t)$ satisfying the conditions of the lemma (i.e., a successor of $((p_1, p_2), s)$ with a different value and the same minimal alphabet B). Since $((p_1, p_2), s)$ is alternating, there exists a set of Σ_2-chains $\mathcal{S}$ such that, for every chain $(s_1, \ldots, s_n)$ of $\mathcal{S}$, we have that $s = s_1$ and $(p_1s_1p_2, \ldots, p_1s_np_2)$ has alternation $n - 1$. By Corollary 9.5, this yields a set of chain trees $\mathbb{S}$ such that $\mathcal{S} = \{\mathrm{val}_C(T) \mid T \in \mathbb{S}\}$. By construction, $\mathbb{S}$ has unbounded alternation and, hence, unbounded recursive alternation by hypothesis in Proposition 10.2.

We now proceed in two steps. First, we use $\mathbb{S}$ to construct a new set of chain trees $\mathbb{U}$ and that satisfies an additional property that we call *local optimality*. We then choose a tree $T \in \mathbb{U}$ with large enough recursive alternation and use it to construct the desired node $((q_1, q_2), t)$.

Construction of $\mathbb{U}$: Local Optimality. Let us first define local optimality. Note that the definition depends on the pair (p_1, p_2). Let T be a chain tree, x be any operation node in T and $(t_1, \ldots, t_n) = \mathrm{val}_C(x)$. We say that x is *locally optimal* if for all $i < n$, either $t_i = t_{i+1}$ or the chain tree T_i obtained from T by replacing the label of x by $(t_1, \ldots, t_{i-1}, t_i, t_i, t_{i+2}, \ldots, t_n)$ satisfies

$$(p_1, \ldots, p_1) \cdot \mathrm{val}_C(T) \cdot (p_2, \ldots, p_2) \neq (p_1, \ldots, p_1) \cdot \mathrm{val}_C(T_i) \cdot (p_2, \ldots, p_2).$$

Intuitively, this means that for all i, alternating from t_i to t_{i+1} in the label of x is necessary to maintain the value of the tree (in the context determined by $(p_1, \ldots, p_1)$ and $(p_2, \ldots, p_2)$). We say that a chain tree T is *locally optimal* if **all of its operation leaves** are locally optimal.

Lemma 10.6. *There exists a set of locally optimal chain trees $\mathbb{U}$ such that for any $(u_1, \ldots, u_n) \in \{val_C(T) \mid T \in \mathbb{U}\}$, we have $s = u_1$ and $(p_1u_1p_2, \ldots, p_1u_np_2)$ has alternation $n - 1$.*

Proof. From any chain tree T, we construct a new chain tree T' such that

(1) $(p_1, \ldots, p_1) \cdot \mathrm{val}_C(T) \cdot (p_2, \ldots, p_2) = (p_1, \ldots, p_1) \cdot \mathrm{val}_C(T') \cdot (p_2, \ldots, p_2)$.
(2) $\mathrm{val}_C(T)$ and $\mathrm{val}_C(T')$ have the same first element.
(3) T' is locally optimal.

It then suffices to let $\mathbb{U}$ be the set of all trees T' constructed in this way from trees T of $\mathbb{S}$.

Let T be any chain tree of level n. For all $i < n$, define the *i-alternation* of T as the number of operation leaves x in T such that $\mathrm{val}_C(x) = (t_1, \ldots, t_n)$ with $t_i \neq t_{i+1}$. Finally, define the *index of* T as the sequence of size $n - 1$ of its i-alternations, ordered increasingly with respect to values of i. Note that the lexicographic ordering on this set of sequences of fixed length is well founded.

Assume that T is not locally optimal. We explain how to construct a new chain tree T' satisfying (1), (2) and

(3') T' has strictly smaller index than T.

It then suffices to iteratively apply this construction starting from T until we get the desired locally optimal tree (which must eventually happen since the ordering on indices of chain trees of level n is well founded). The construction of T' is as follows. Since T is not locally optimal, there exists an operation leaf x of T that is not locally optimal. Let $(t_1, \ldots, t_n) = \mathrm{val}_C(x)$. By hypothesis, there exists $i < n$ such that $t_i \neq t_{i+1}$ and the chain tree T' obtained by replacing the label of x by $(t_1, \ldots, t_{i-1}, t_i, t_i, t_{i+2}, \ldots, t_n)$ satisfies 1. Since this replacement does not modify the first component of any node, Property 2 is satisfied as well. Finally, by definition, for any $j < i$, T, T' have the same j-alternation and T' has strictly smaller i-alternation. It follows that T' has a strictly smaller index than T, which terminates the proof. □

For the remainder of the section, we assume that $\mathbb{U}$ is fixed as the set of locally optimal chain trees of Lemma 10.6. Observe that, by definition, $\mathbb{U}$ has unbounded alternation. Hence, by the hypothesis in Proposition 10.2, it has unbounded recursive alternation as well.

Construction of the node $((q_1, q_2), t)$. We choose a tree $T \in \mathbb{U}$. The choice is based on the following lemma.

Lemma 10.7. *There exists an integer k such that, for all $t_1, t_2 \in M$,*

$$(t_1, t_2)^k \in C_2[\alpha] \Rightarrow (t_1, t_2)^* \subseteq C_2[\alpha].$$

Proof. If for all $t_1, t_2 \in M$, we have that $(t_1, t_2)^* \subseteq C_2[\alpha]$, we choose $k = 1$. Otherwise, since $C_2[\alpha]$ is closed under subwords (Fact 5.2), if $(t_1, t_2)^k \notin C_2[\alpha]$, then, for all $j \geqslant k$, we have that $(t_1, t_2)^j \notin C_2[\alpha]$ as well. Therefore, one can define k as the largest integer such that there exist $t_1, t_2 \in M$ with $(t_1, t_2)^{k-1} \in C_2[\alpha]$ but $(t_1, t_2)^k \notin C_2[\alpha]$ (with the convention that $(t_1, t_2)^0 \in C_2[\alpha]$). □

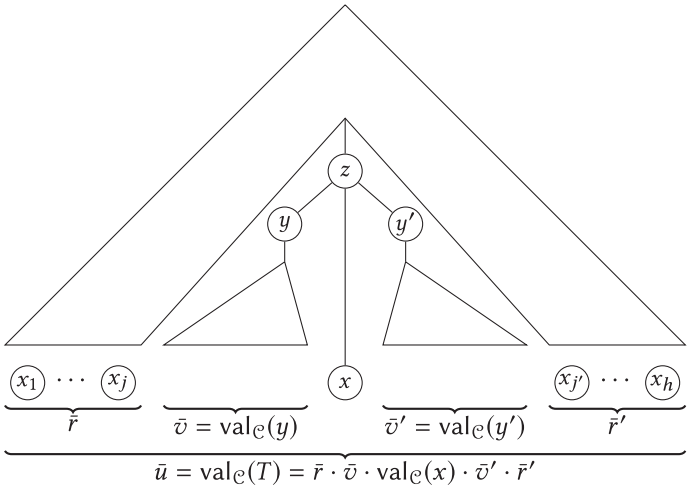


Fig. 8. The chain tree T.

Let $m = |M|^2 \times k$, with k defined as in Lemma 10.7. Since $\mathbb{U}$ has unbounded recursive alternation, there exists $T \in \mathbb{U}$ with recursive alternation m. Let n be the level of T.

We now use T to construct the desired node $((q_1, q_2), t)$ in $G[\alpha]$ fulfilling all properties of Lemma 10.5. We begin by summarizing all hypotheses that we have on T (these hypotheses are also represented in Figure 8). Let $\bar{u} = (u_1, \ldots, u_n) = \mathrm{val}_C(T)$ and recall that, by choice of T in $\mathbb{U}$, we have that $u_1 = s$. Let $x_1, \ldots, x_h$ be the leaves of T (from left to right). Recall that, by Fact 9.2, $\mathrm{val}_C(T) = \mathrm{val}_C(x_1) \cdots \mathrm{val}_C(x_h)$.

By definition of recursive alternation, T must contain an operation leaf $x \in \{x_1, \ldots, x_h\}$ whose label $\mathrm{val}_C(x)$ has alternation m. By definition of chain trees, x is the middle child of an operation node z. We let y, y' be the left and right children of this node. Finally, we let $j, j' \leqslant h$ such that x_{j+1} is the leftmost leaf that is a descendant of y and $x'_{j'-1}$ is the rightmost leaf that is a descendant of y' (see Figure 8). We now define the following chains:

$$\begin{array}{lcll} \bar{t} & = & (t_1, \ldots, t_n) & = \mathrm{val}_C(x) \\ \bar{v} & = & (v_1, \ldots, v_n) & = \mathrm{val}_C(y) \\ \bar{v}' & = & (v'_1, \ldots, v'_n) & = \mathrm{val}_C(y') \\ \bar{r} & = & (r_1, \ldots, r_n) & = \mathrm{val}_C(x_1) \cdots \mathrm{val}_C(x_j) \\ \bar{r}' & = & (r'_1, \ldots, r'_n) & = \mathrm{val}_C(x_{j'}) \cdots \mathrm{val}_C(x_h) \end{array}$$

By definition, we have $\mathrm{val}_C(T) = \bar{r} \cdot \bar{v} \cdot \bar{t} \cdot \bar{v}' \cdot \bar{r}'$. Since x is an operation node, there exists an idempotent $(e, \mathcal{E}) \in \mathcal{J}_{2,n}[\alpha]$ such that

- $\mathrm{val}_{\mathcal{J}}(y) = \mathrm{val}_{\mathcal{J}}(y') = (e, \mathcal{E})$.
- $\bar{v}, \bar{v}' \in (e, \mathcal{E})$.
- $\bar{t} \in (e, \mathcal{E}) \cdot (1_M, \mathcal{T}) \cdot (e, \mathcal{E})$ with $\mathcal{T} = \left\{(t_1, \ldots, t_{n-1}) \in C_{2,n-1}[\alpha] \mid \mathrm{alph}(t_1) = \mathrm{alph}(e)\right\}$.

By choice of x, $\bar{t} = (t_1, \ldots, t_n) = \mathrm{val}_C(x)$ has alternation $m = |M|^2 \cdot k$. It follows from the pigeonhole principle that there exist i such that $t_i \neq t_{i+1}$ and a set $I \subseteq \{1, \ldots, n-1\}$ of size at least k such that for all $j \in I$, $t_j = t_i$ and $t_{j+1} = t_{i+1}$. Note that this implies that the chain $(t_i, t_{i+1})^k$ is a subword of $(t_1, \ldots, t_n)$ and, therefore, is a Σ_2-chain. By choice of k (see Lemma 10.7), it follows that $(t_i, t_{i+1})^* \subseteq C_2[\alpha]$.

Recall that T is locally optimal since it belongs to $\mathbb{U}$. By definition of local optimality, changing t_{i+1} to t_i in the label $\mathrm{val}_C(x)$ of the operation node x changes the value $\mathrm{val}_C(T)$, hence, its $(i+1)$th component. We therefore obtain the following fact.

Fact 10.8. $p_1 r_{i+1} v_{i+1} t_i v'_{i+1} r'_{i+1} p_2 \neq p_1 r_{i+1} v_{i+1} t_{i+1} v'_{i+1} r'_{i+1} p_2$.

We now define the node $((q_1, q_2), t)$. We let

$$q_1 = p_1 r_{i+1} v_{i+1} \text{ and } q_2 = v'_{i+1} r'_{i+1} p_2.$$

It is immediate from Fact 10.8 that either $q_1 t_i q_2 \neq p_1 s p_2$ or $q_1 t_{i+1} q_2 \neq p_1 s p_2$. In the first case, we let $t = t_i$; in the second case, we let $t = t_{i+1}$. Note that since $(t_i, t_{i+1})^* \subseteq C_2[\alpha]$ and $q_1 t_i q_2 \neq q_1 t_{i+1} q_2$, the node $((q_1, q_2), t)$ is alternating by definition.

It remains to prove that

— $\mathsf{alph}(t) = \mathsf{alph}(s)$ and that
— there is an edge $((p_1, p_2), s) \to ((q_1, q_2), t)$ in $G[\alpha]$.

For the proof, we assume that $t = t_i$ (the case $t = t_{i+1}$ is similar).

Observe that in the Σ_2-chain, $\mathrm{val}_C(T) = (u_1, \ldots, u_n)$, $u_1 = s$ and $u_i = p_1 r_i v_i t_i v'_i r'_i p_2$. Since $(u_1, \ldots, u_n)$ is a Σ_2-chain, one can verify that all of its elements have the same alphabet; hence, $\mathsf{alph}(u_i) = \mathsf{alph}(s) = B$ and $\mathsf{alph}(t) \subseteq B$. Now, recall that B was chosen as a minimal alphabet such that there is an alternating node $((q_1, q_2), t)$ with $\mathsf{alph}(t) = B$. Hence, since $((q_1, q_2), t)$ is alternating and $\mathsf{alph}(t) \subseteq B$, we have that $\mathsf{alph}(t) = B = \mathsf{alph}(s)$.

Finally, we need to prove that there is an edge from $((p_1, p_2), s)$ to $((q_1, q_2), t)$, i.e., to find $s_1, s_2 \in M$ such that (s, s_1, s_2) is an alternation schema and $p_1 s_1 = q_1$ and $s_2 p_2 = q_2$. We define $s_1 = r_{i+1} v_{i+1}$ and $s_2 = v'_{i+1} r'_{i+1}$. That $p_1 s_1 = q_1$ and $s_2 p_2 = q_2$ is immediate by definition of q_1 and q_2. It remains to prove that (s, s_1, s_2) is an alternation schema.

Recall that $\bar{v}, \bar{v}' \in (e, \mathcal{E})$ with $(e, \mathcal{E}) \in \mathcal{J}_{2,n}[\alpha]$. Define $\mathcal{F} \subseteq M$ as the set containing all elements that are at component i of some chain in $\mathcal{E}$. In particular, $v_{i+1}, v'_{i+1} \in \mathcal{F}$. It is immediate from Fact 5.8 (closure of junctures under subwords) that $(e, \mathcal{F}) \in \mathcal{J}_{2,2}[\alpha]$. Moreover, the idempotency of $(e, \mathcal{E})$ entails that $(e, \mathcal{F})$ is also idempotent. By Fact 5.2 (closure of chains under subwords), we have that $(r_1, r_{i+1}) \in C_2[\alpha]$ and $(r'_1, r'_{i+1}) \in C_2[\alpha]$. Hence, we have that $(r_1, \{r_{i+1}\}) \in \mathcal{J}_{2,2}[\alpha]$ and $(r'_1, \{r'_{i+1}\}) \in \mathcal{J}_{2,2}[\alpha]$. We conclude that $s = u_1 = r_1 e r'_1, s_1 \in \{r_{i+1}\} \cdot \mathcal{F}$ and $s_2 \in \mathcal{F} \cdot \{r'_{i+1}\}$. Moreover, by definition, $\mathsf{alph}(e) = \mathsf{alph}(t) = B = \mathsf{alph}(s)$: we conclude that (s, s_1, s_2) is an alternation schema, which terminates the proof. □

11 PROOF OF PROPOSITION 9.6

In this section, we prove Proposition 9.6. Recall that we have fixed a morphism $\alpha : A^* \to M$ into a finite monoid M. Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation but bounded recursive alternation. We need to prove that α does not satisfy one of the equations in Equation (17). As for the previous section, we will use a new object that is specific to this case: *chain matrices*.

Chain Matrices. Let $n \in \mathbb{N}$. A *chain matrix of length n* is a rectangular matrix with n columns and whose rows belong to $C_{2,n}[\alpha]$. If $\mathscr{M}$ is a chain matrix, we will denote by $\mathscr{M}_{i,j}$ the entry at row i (starting from the top) and column j (starting from the left) in $\mathscr{M}$. If $\mathscr{M}$ is a chain matrix of length n and with m rows, we call the chain $\big((\mathscr{M}_{1,1} \cdots \mathscr{M}_{m,1}), \ldots, (\mathscr{M}_{1,n} \cdots \mathscr{M}_{m,n})\big)$, the *value* of $\mathscr{M}$. Since $C_{2,n}[\alpha]$ is a monoid by Fact 5.4, the value of a chain matrix is a Σ_2-chain. We give an example with 3 rows in Figure 9.

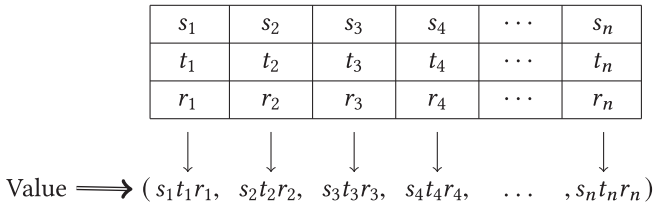


Fig. 9. Value of a chain matrix with 3 rows.

Given a chain matrix $\mathscr{M}$, the *alternation* of $\mathscr{M}$ is the alternation of its value. Finally, the *local alternation* of a chain matrix, $\mathscr{M}$, is the largest integer m such that $\mathscr{M}$ has a row with alternation m. We now prove the following two propositions.

Proposition 11.1. *Assume that there exists a set of chain trees $\mathbb{S} \subseteq \mathbb{T}_C[\alpha]$ with unbounded alternation and recursive alternation bounded by $K \in \mathbb{N}$. Then, there exist chain matrices with arbitrarily large alternation and local alternation bounded by K.*

Proposition 11.2. *Assume that there exist chain matrices with arbitrarily large alternation and local alternation bounded by $K \in \mathbb{N}$. Then, α does not satisfy Equation (17).*

Proposition 9.6 is an immediate consequence of Proposition 11.1 and 11.2. Note that chain matrices are reused from [16] (in which they are called "strategy matrices"). Moreover, in this case, going from chain trees to chains matrices (i.e., proving Proposition 11.1) is simple and the main difficulty is proving Proposition 11.2. This means that while our presentation is different from that of [16], the fundamental arguments themselves are essentially the same. We give a full proof for the sake of completeness. We begin by proving Proposition 11.1.

Proof of Proposition 11.1. We prove that, for all $n \in \mathbb{N}$, there exists a chain matrix $\mathscr{M}$ of alternation n and local alternation bounded by K. By definition of $\mathbb{S}$, there exists a tree $T \in \mathbb{S}$ whose value has alternation n and has recursive alternation bounded by K. Let $x_1, \ldots, x_m$ denote the leaves of T, listed from left to right. By Fact 9.2, $\mathrm{val}_C(T) = \mathrm{val}_C(x_1) \cdots \mathrm{val}_C(x_m)$. Observe that, by definition, for all i, $\mathrm{val}_C(x_i)$ has alternation bounded by K. Therefore, it suffices to pick $\mathscr{M}$ as the $m \times n$ matrix, where row i is filled with $\mathrm{val}_C(x_i)$. □

It now remains to prove Proposition 11.2. We proceed as follows: assuming that there exists a chain matrix $\mathscr{M}$ with local alternation bounded by K and very large alternation, we refine $\mathscr{M}$ in several steps to ultimately obtain a chain matrix of a special kind that we call a *contradiction matrix*. There are two types of contradiction matrices, *increasing* and *decreasing*. Both are chain matrices of length 6 with the following entries:

u_1	v_1	f	f	f	f
e	e	u_2	v_2	f	f
e	e	e	e	u_3	v_3

Increasing Contradiction Matrix

f	f	f	f	u_3	v_3
f	f	u_2	v_2	e	e
u_1	v_1	e	e	e	e

Decreasing Contradiction Matrix

where e, f are idempotents and $fu_2e \neq fv_2e$. As the name suggests, the existence of a contradiction matrix contradicts Equation (17). This is what we state in the following lemma.

Lemma 11.3. *If there exists a contradiction matrix, then α does not satisfy Equation (17).*

Proof. Assume that we have an increasing contradiction matrix (the other case is treated in a symmetrical way). Since $fu_2e \neq fv_2e$, either $fu_2e \neq fe$ or $fv_2e \neq fe$. By symmetry, assume that

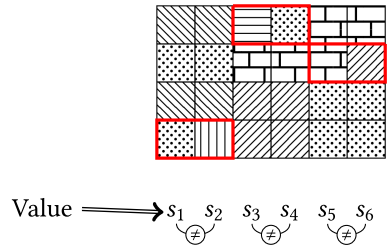


Fig. 10. A tame chain matrix of length 6.

it is the former. Since e, f are idempotents, this means that $f^\omega u_2 e^\omega \neq f^\omega e^\omega$. However, by definition of chain matrices, $(e, u_2, v_2, f) \in C_2[\alpha]$ and, therefore, $(e, u_2, f) \in C_2[\alpha]$, which contradicts the second equation in Equation (17). Note that we used only one-half of Equation (17); the other half is used in the decreasing case. □

By Lemma 11.3, it suffices to prove the existence of a contradiction matrix to conclude the proof of Proposition 11.2. This is what we do in the remainder of this section. By hypothesis, we know that there exist chain matrices with arbitrarily large alternation and local alternation bounded by $K \in \mathbb{N}$. For the remainder of the section, we assume that this hypothesis holds. We use several steps to prove that we can choose our chain matrices with increasingly strong properties until we get a contradiction matrix. We use two intermediary types of matrices, which we call *Tame Chain Matrices* and *Monotonous Chain Matrices*. We divide the proof in three sections, one for each step.

11.1 Tame Chain Matrices

Let $\mathscr{M}$ be a chain matrix of *even length* 2ℓ and let $j \leqslant \ell$. The *set of alternating rows for* j, denoted by $\mathrm{alt}(\mathscr{M}, j)$, is the set $\{i \mid \mathscr{M}_{i,2j-1} \neq \mathscr{M}_{i,2j}\}$. Let $(s_1, \ldots, s_{2\ell})$ be the value of $\mathscr{M}$. We say that $\mathscr{M}$ is *tame* if

a) for all $j \leqslant \ell$, $s_{2j-1} \neq s_{2j}$,
b) for all $j \leqslant \ell$, $\mathrm{alt}(\mathscr{M}, j)$ is a singleton, and
c) if $j \neq j'$, then $\mathrm{alt}(\mathscr{M}, j) \neq \mathrm{alt}(\mathscr{M}, j')$.

We represent a tame chain matrix of length 6 in Figure 10. Observe that the definition considers only the relationship between odd columns and the next even column. Moreover, observe that a tame chain matrix of length 2ℓ has, by definition, alternation at least ℓ.

Lemma 11.4. *There exist tame chain matrices of arbitrarily large length.*

Proof. Let $n \in \mathbb{N}$. We explain how to construct a tame chain matrix of length $2n$. By hypothesis, there exists a chain matrix $\mathscr{M}$ with local alternation at most K and alternation greater than $2nK$. Let m be the number of rows of $\mathscr{M}$. We explain how to modify $\mathscr{M}$ to obtain a matrix satisfying (a, (b and (c. Recall that Σ_2-chains are closed under subwords; therefore, removing columns from $\mathscr{M}$ yields a chain matrix. Since $\mathscr{M}$ has alternation greater than $2nK$, it is simple to see that by removing columns, one can obtain a chain matrix of length $2nK$ that satisfies (a). We denote this matrix by $\mathscr{N}$. We now proceed in two steps. First, we modify the entries in $\mathscr{N}$ to get a matrix $\mathscr{P}$ of length $2nK$, satisfying both (a) and (b). Then, we use our bound on local alternation to remove columns and enforce (c) in the resulting matrix.

Construction of $\mathscr{P}$. Let $j \leqslant nK$ such that $\mathrm{alt}(\mathscr{N}, j)$ is of size at least 2. We modify the matrix to reduce the size of $\mathrm{alt}(\mathscr{N}, j)$ while preserving (a). One can then repeat the operation to get the

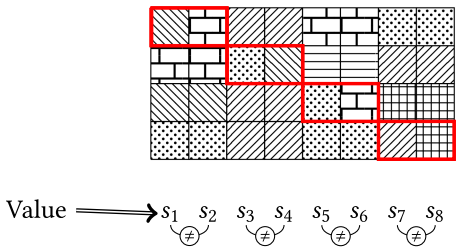


Fig. 11. A monotonous chain matrix (increasing).

desired matrix. Let $i \in \mathrm{alt}(\mathscr{N}, j)$. Let $s_1 = \mathscr{N}_{1,2j-1} \cdots \mathscr{N}_{i-1,2j-1}$ and $s_2 = \mathscr{N}_{i+1,2j-1} \cdots \mathscr{N}_{m,2j-1}$. We distinguish two cases.

First, if $s_1\mathscr{N}_{i,2j-1}s_2 \neq s_1\mathscr{N}_{i,2j}s_2$, then, for all $i' \neq i$, we replace entry $\mathscr{N}_{i',2j}$ with entry $\mathscr{N}_{i',2j-1}$. One can verify that this yields a chain matrix of length $2nK$, local alternation bounded by K. Moreover, it still satisfies (a), since $s_1\mathscr{N}_{i,2j-1}s_2 \neq s_1\mathscr{N}_{i,2j}s_2$. Finally, $\mathrm{alt}(\mathscr{N}, j)$ is now a singleton, namely, $\{i\}$.

In the second case, we have that $s_1\mathscr{N}_{i,2j-1}s_2 = s_1\mathscr{N}_{i,2j}s_2$. In that case, we replace $\mathscr{N}_{i,2j-1}$ with $\mathscr{N}_{i,2j}$. One can verify that this yields a chain matrix of length $2nK$, local alternation bounded by K. Moreover, it still satisfies (a) since we did not change the value of the matrix. Finally, the size of $\mathrm{alt}(\mathscr{N}, j)$ has decreased by 1.

Construction of the tame matrix. We now have a chain matrix $\mathscr{P}$ of length $2nK$, with local alternation bounded by K and satisfying both (a) and (b). Since (a) and (b) are satisfied, for all $j \leqslant nK$, there exists exactly one row i such that $\mathscr{N}_{i,2j-1} \neq \mathscr{N}_{i,2j}$. Moreover, since each row has alternation at most K, a single row i has this property for at most K indices j. Therefore, it suffices to remove at most $n(K-1)$ pairs of odd-even columns to get a matrix that satisfies (c). Since the original matrix had length $2nK$, this leaves a matrix of length at least $2n$, as desired. □

11.2 Monotonous Chain Matrices

Let $\mathscr{M}$ be a tame chain matrix of length $2n$ and let $x_1, \ldots, x_n$ be integers such that, for all j, $\mathrm{alt}(\mathscr{M}, j) = \{x_j\}$. We say that $\mathscr{M}$ is a *monotonous chain matrix* if it has exactly n rows and $1 = x_1 < x_2 < \cdots < x_n = n$ (in which case the matrix is said to be *increasing*) or $n = x_1 > x_2 > \cdots > x_n = 1$ (in which case we say the matrix is *decreasing*). We give a representation of the increasing case in Figure 11.

Lemma 11.5. *There exist monotonous chain matrices of arbitrarily large length.*

Proof. Let $n \in \mathbb{N}$. We explain how to construct a monotonous chain matrix of length $2n$. By Lemma 11.4, there exists a tame chain matrix $\mathscr{M}$ of length $2n^2$. Let $x_1, \ldots, x_{n^2}$ be the indices such that, for all j, $\mathrm{alt}(\mathscr{M}, j) = \{x_j\}$. Note that, by tameness, $x_j \neq x_{j'}$ for $j \neq j'$. Since the sequence $x_1, \ldots, x_{n^2}$ is of length n^2, we can extract, using the Erdös-Szekeres theorem, a monotonous sequence of length n, $x_{j_1} < \cdots < x_{j_n}$ or $x_{j_1} > \cdots > x_{j_n}$ with $j_1 < \cdots < j_n$. By symmetry, we assume that it is the former and construct an increasing chain matrix of length n.

Let $\mathscr{P}$ be the matrix of length $2n$ obtained from $\mathscr{M}$ by keeping only the pairs of columns $2j-1, 2j$ for $j \in \{j_1, \ldots, j_n\}$. Let $x'_1, \ldots, x'_n$ be the indices such that, for all j, $\mathrm{alt}(\mathscr{P}, j) = \{x'_j\}$. By definition, $x'_1 < \cdots < x'_n$. We now want $\mathscr{P}$ to have exactly n rows. Note that the rows whose indices do not belong to $\{x'_1, \cdots, x'_n\}$ are constant chains. We simply merge these rows with others. For example, if row i is labeled with the constant chain $(s, \ldots, s)$, let $(s_1, \ldots, s_{2n})$ be the label of

row $i+1$. We remove row i and replace row $i+1$ by the Σ_2-chain $(ss_1, \ldots, ss_{2n})$. Repeating the operation yields the desired increasing monotonous chain matrix. □

11.3 Construction of the Contradiction Matrix

We can now use Lemma 11.5 to construct a contradiction matrix and end the proof of Proposition 9.6. We state this in the following proposition.

Proposition 11.6. *There exists a contradiction matrix.*

The remainder of this section is devoted to the proof of Proposition 11.6. The result follows from a Ramsey argument. We use Lemma 11.5 to choose a monotonous matrix of sufficiently large length. Then, we use Ramsey's Theorem (for hypergraphs with edges of size 3) to extract the desired contradiction matrix.

We first define the length of the monotonous chain matrix that we need to pick. By Ramsey's Theorem, for every $m \in \mathbb{N}$, there exists a number $\varphi(m)$ such that, for any complete 3-hypergraph with hyperedges colored over the monoid M, there exists a complete sub-hypergraph of size m in which all edges share the same color. We choose $n = \varphi(\varphi(4)+1)$. By Lemma 11.5, there exists a monotonous chain matrix $\mathscr{M}$ of length $2n$. Since it is monotonous, $\mathscr{M}$ has n rows.

By symmetry, we assume that $\mathscr{M}$ is increasing and use it to construct an increasing contradiction matrix. We use our choice of n to extract a contradiction matrix from $\mathscr{M}$. We proceed in two steps, using Ramsey's Theorem each time. In the first step, we treat all entries above the diagonal in $\mathscr{M}$ and in the second step all entries below the diagonal. We state the first step in the next lemma.

Lemma 11.7. *There exists an increasing monotonous matrix $\mathscr{N}$ of length $2 \cdot \varphi(4)$ such that all cells above the diagonal contain the same idempotent $f \in M$.*

Proof. This is proved by applying Ramsey's Theorem to $\mathscr{M}$. Consider the complete 3-hypergraph whose nodes are $\{0, \ldots, n\}$. We label the hyperedge $\{i_1, i_2, i_3\}$, where $i_1 < i_2 < i_3$ by the value obtained by multiplying in the monoid M the cells that appear in rows $i_1+1, \ldots, i_2$ in column $2i_3 - 1$. Observe that since $i_1 < i_2 < i_3$, by monotonicity, these entries are the same as in column $2i_3$. More formally, the label of the hyperedge $\{i_1, i_2, i_3\}$ with $i_1 < i_2 < i_3$ is, therefore,

$$\mathscr{M}_{i_1+1,2i_3-1} \cdots \mathscr{M}_{i_2,2i_3-1} = \mathscr{M}_{i_1+1,2i_3} \cdots \mathscr{M}_{i_2,2i_3}.$$

By choice of n, we can apply Ramsey's Theorem to this coloring. We get a subset of $\varphi(4)+1$ vertices, say, $K = \{k_1, \ldots, k_{\varphi(4)+1}\} \subseteq \{0, \ldots, n\}$, such that all hyperedges connecting nodes in K have the same color, say, $f \in M$. For $i_1 < i_2 < i_3 < i_4$ in K, note that the color of the hyperedge $\{i_1, i_3, i_4\}$ is, by definition, the product of the colors of the hyperedges $\{i_1, i_2, i_4\}$ and $\{i_2, i_3, i_4\}$. Therefore, the common color f needs to be an idempotent: $f = ff$. We now extract the desired matrix $\mathscr{N}$ from $\mathscr{M}$ according to the subset K. The main idea is that the new row i in $\mathscr{N}$ will be the merging of rows $k_i + 1$ to k_{i+1} in $\mathscr{M}$ and the new pair of columns $2j-1, 2j$ will correspond to the pair $2k_{j+1} - 1, 2k_{j+1}$ in $\mathscr{M}$.

We first merge rows. For all $i \geqslant 1$, we "merge" all rows from $k_i + 1$ to k_{i+1} into a single row. More precisely, this means that we replace the rows $k_i + 1$ to k_{i+1} by a single row containing the Σ_2-chain

$$(\mathscr{M}_{k_i+1,1} \cdots \mathscr{M}_{k_{i+1},1}, \ldots, \mathscr{M}_{k_i+1,2n} \cdots \mathscr{M}_{k_{i+1},2n}).$$

Moreover, we remove the top and bottom rows, i.e., rows 1 to k_1 and rows $k_{\varphi(4)+1} + 1$ to $\varphi(4)+1$. Then, we remove all columns from 1 to $2k_2 - 2$, all columns from $2k_{\varphi(4)+1} + 1$ to $2n$, and for all $i \geqslant 2$, all columns from $2k_i + 1$ to $2k_{i+1} - 2$. One can verify that these two operations applied together preserve monotonicity. Observe that the resulting matrix $\mathscr{N}$ has exactly $2 \times \varphi(4)$ columns.

Moreover, the cell $i, 2j$ in the new matrix contains entry $\mathscr{M}_{k_i+1,2k_{j+1}} \cdots \mathscr{M}_{k_{i+1},2k_{j+1}}$. In particular, if $j > i$, by definition of the set K, this entry is f, which means that $\mathscr{N}$ satisfies the conditions of the lemma. □

It remains to apply Ramsey's Theorem a second time to the matrix $\mathscr{N}$ obtained from Lemma 11.7 to treat the cells below the diagonal and get the contradiction matrix. We state this in the following last lemma.

Lemma 11.8. *There exists an increasing monotonous matrix $\mathscr{P}$ of length 6 such that all cells above the diagonal contain the same idempotent $f \in M$ and all cells below the diagonal contain the same idempotent $e \in M$ (i.e., $\mathscr{P}$ is an increasing contradiction matrix).*

Proof. The argument is identical to the one of Lemma 11.7. This time, we apply it to the matrix $\mathscr{N}$ of length $2 \times \varphi(4)$ for the cells below the diagonal. The monochromatic set given by Ramsey's theorem is of size 4 this time, which, with the above construction, will leave a matrix with 3 rows and 6 columns. □

12 ADDING SUCCESSOR: THE ENRICHED HIERARCHY

All decidability results that we have proved so far are for fragments of the order hierarchy. In this section, we transfer these results to the enriched hierarchy. More precisely, we present algorithms for the following problems:

— the separation problem for $\Sigma_2(<, +1, min, max)$ and $\Pi_2(<, +1, min, max)$.
— the membership problem for $\Sigma_3(<, +1, min, max)$.
— the membership problem for $\mathcal{B}\Sigma_2(<, +1, min, max)$.

For each problem, we actually present a reduction to the same problem for the corresponding fragment in the order hierarchy; decidability then follows from the results of the previous sections. The transfer results are not new and were initially presented in [75, 102] for the membership problem and in [4, 96] and [83] for the separation problem (unlike the former, this latter work also copes with classes not closed under complement and, therefore, can be applied to $\Sigma_i(<)$). In this section, we state only the reductions and refer the reader to these papers for proofs.

Note that the reductions that we use are all taken from [83]. In particular, for membership, while the underlying ideas remain similar to that of [75, 102], the reduction itself is fairly different from the original one.

We divide the section into two parts. In the first part, we define the main tool used in the reductions: *the morphism of well-formed words*. In the second part, we present the reductions themselves.

12.1 Morphism of Well-Formed Words

Fix a morphism $\alpha : A^* \to M$ into a finite monoid M. We define $E \subseteq M$ as the set of idempotents of $\alpha(A^+)$, i.e., E is the set of idempotents of M that are images of a nonempty word. We define a new alphabet $\mathbb{A}_\alpha$, called *alphabet of well-formed words of α*, as follows:

$$\mathbb{A}_\alpha = \begin{array}{cl} & M \\ \cup & M \times E \\ \cup & E \times M \\ \cup & E \times M \times E \end{array}$$

We will not be interested in all words in $\mathbb{A}_\alpha^*$, but only in those that are well formed. A word $\mathbb{w} \in \mathbb{A}_\alpha^*$ is said to be *well formed* if one of the two following properties hold:

— $\mathbb{w} = \varepsilon$ or is a single-letter word $s \in M$.
— $\mathbb{w} = (s_1, f_1)(e_2, s_2, f_2)(e_3, s_3, f_3) \cdots (e_n, s_n) \in (S \times E) \cdot (E \times S \times E)^* \cdot (E \times S)$ and for all $1 \leqslant i \leqslant n-1$, we have that $f_i = e_{i+1}$.

The following fact is immediate.

Fact 12.1. *The set of well-formed words of $\mathbb{A}^*_\alpha$ is a regular language.*

Observe that one can define a monoid morphism $\beta : \mathbb{A}^*_\alpha \to M$ by letting $\beta(s) = s$ for all $s \in M$, $\beta((e,s)) = es$ for all $(e,s) \in E \times M$, $\beta((s,e)) = se$ for all $(s,e) \in M \times E$, and $\beta((e,s,f)) = esf$ for all $(e,s,f) \in E \times M \times E$. We call β the *morphism of well-formed words associated to α*.

Associated language of well-formed words. To any language $L \subseteq A^*$ that is recognized by α, one can associate a language of well-formed words $\mathbb{L} \subseteq \mathbb{A}^*_\alpha$ (depending on α):

$$\mathbb{L} = \left\{ \mathbb{w} \in \mathbb{A}^*_\alpha \mid \mathbb{w} \text{ is well formed and } \beta(\mathbb{w}) \in \alpha(L) \right\}.$$

By definition, the language $\mathbb{L} \subseteq \mathbb{A}^*_\alpha$ is the intersection of the language of well-formed words with $\beta^{-1}(\alpha(L))$. Therefore, it is immediate by Fact 12.1 that it is regular, more precisely:

Fact 12.2. *Let $L \subseteq A^*$ that is recognized by α. Then, the associated language of well-formed words $\mathbb{L} \subseteq \mathbb{A}^*_\alpha$ is a regular language, and one can compute it from α.*

12.2 Reductions

We can now state the reductions; we begin with the separation result.

Theorem 12.3 (Place and Zeitoun [83]). *Let L_0, L_1 be regular languages and let $\alpha : A^* \to M$ be a morphism into a finite monoid M that recognizes both L_0 and L_1. Finally, let $\mathbb{L}_0$ and $\mathbb{L}_1$ be the languages of well-formed words associated to L_0 and L_1.*
For all $i \geqslant 1$, L_0 is $\Sigma_i(<, +1, min, max)$-separable (resp., $\mathcal{B}\Sigma_i(<, +1, min, max)$-separable) from L_1 if and only if $\mathbb{L}_0$ is $\Sigma_i(<)$-separable (resp., $\mathcal{B}\Sigma_i(<)$-separable) from $\mathbb{L}_1$.

Theorem 12.3 reduces $\Sigma_i(<, +1, min, max)$-separability (resp., $\mathcal{B}\Sigma_i(<, +1, min, max)$-separability) to $\Sigma_i(<)$-separability (resp., $\mathcal{B}\Sigma_i(<)$-separability). Since we already know that $\Sigma_2(<)$-separability is decidable (see Corollary 7.3), we get the following corollary:

Corollary 12.4. *Given as input two regular languages L_1, L_2 it is decidable to test whether L_1 can be $\Sigma_2(<, +1, min, max)$-separated (resp., $\Pi_2(<, +1, min, max)$-separated) from L_2.*

This terminates our separation results. We now state the membership reduction.

Theorem 12.5 (Place and Zeitoun [83]). *Let L be a regular language and let $\alpha : A^* \to M$ be a morphism into a finite monoid M that recognizes L. Finally, let $\mathbb{L}$ be the language of well-formed words associated to L.*
For all $i \geqslant 3$, L is $\Sigma_i(<, +1, min, max)$-definable if and only if $\mathbb{L}$ is $\Sigma_i(<)$-definable.
For all $i \geqslant 2$, L is $\mathcal{B}\Sigma_i(<, +1, min, max)$-definable if and only if $\mathbb{L}$ is $\mathcal{B}\Sigma_i(<)$-definable.

Observe that, in contrast to the separation reduction, the membership reduction does not work for lower levels in the hierarchy. For example, it does not work for $\mathcal{B}\Sigma_1(<)$ and $\Sigma_2(<)$. This is essentially because these logics are not powerful enough to express that a word in $\mathbb{A}^*_\alpha$ is well formed (this is only possible for logics including and above $\Pi_2(<)$).
By combining Theorem 12.5 with Corollaries 7.4 and 8.2, we get the desired corollary.

Corollary 12.6. *Given as input a regular language L, the following problems are decidable:*

—*whether L is definable in* $\mathcal{B}\Sigma_2(<, +1, min, max)$.
—*whether L is definable in* $\Delta_3(<, +1, min, max)$.
—*whether L is definable in* $\Sigma_3(<, +1, min, max)$.
—*whether L is definable in* $\Pi_3(<, +1, min, max)$.

13 CONCLUSION

We solved the separation problem for $\Sigma_2(<)$ using the new notion of Σ_2-chains, and we used our solution to prove decidable characterizations for $\mathcal{B}\Sigma_2(<)$, $\Delta_3(<)$, $\Sigma_3(<)$, and $\Pi_3(<)$. The main open problem in this field remains to lift up these results to higher levels in the hierarchy. In particular, we proved that, for any positive integer i, generalizing our separation solution to $\Sigma_i(<)$ (i.e., being able to compute the Σ_i-chains of length 2) would yield a decidable characterization for $\Sigma_{i+1}(<)$, $\Pi_{i+1}(<)$, and $\Delta_{i+1}(<)$.

Our algorithm for computing Σ_2-chains cannot be directly generalized for higher levels. An obvious reason for this is the fact that it considers Σ_2-chains parametrized by subalphabets. This parameter is designed to take care of the alternation between levels 1 and 2, but is not adequate for higher levels. However, this problem has been circumvented for the next level: a new algorithm to compute $\Sigma_3(<)$-chains has been designed and proved in [78]. This requires introducing hybrid objects capturing even more information than $\Sigma_3(<)$-chains and $\Sigma_3(<)$-junctures, and which are amenable to a recursive computation. Yet, this difficulty is unlikely to be the only problem. In particular, we do have an algorithm that avoids using the alphabet, but it remains difficult to generalize. We leave the presentation of this alternate algorithm for further work.

Another orthogonal research direction is to solve separation for $\mathcal{B}\Sigma_i(<)$ levels. The idea of exploiting the knowledge on some class to solve separation for the Boolean algebra that it generates is actually more meaningful for other classes than levels of the alternation hierarchy. Indeed, one can generalize the relationship between $\Sigma_i(<)$-chains with unbounded alternation and separation for $\mathcal{B}\Sigma_i(<)$ (as stated in Theorem 6.6) by replacing the class $\Sigma_i(<)$ with any lattice $\mathcal{L}$ of regular languages. Otherwise stated, one can generalize the definitions to make generic the link between $\mathcal{L}$-chains with unbounded alternation and separation by languages of $\mathcal{B}\mathcal{L}$, the Boolean algebra generated by the lattice $\mathcal{L}$. Even for $\mathcal{B}\Sigma_2(<)$, the problem of determining, for two given elements s_1, s_2 of the monoid under consideration, whether the set of chains $(s_1, s_2)^*$ only consists of $\Sigma_i(<)$-chains is still wide open. Solving it may provide intuition for upper levels, but probably requires new concepts.

REFERENCES

[1] Douglas Albert, Robert Baldinger, and John Rhodes. 1992. Undecidability of the identity problem for finite semigroups. *The Journal of Symbolic Logic* 57, 1 (1992), 179—192.
[2] Jorge Almeida. 1991. Implicit operations on finite J-trivial semigroups and a conjecture of I. Simon. *Journal of Pure and Applied Algebra* 69, 3 (1991), 205–218.
[3] Jorge Almeida. 1995. *Finite Semigroups and Universal Algebra*. World Scientific, Singapore.
[4] Jorge Almeida. 1999. Some algorithmic problems for pseudovarieties. *Publicationes Mathematicae Debrecen* 54 (1999), 531–552.
[5] Jorge Almeida and Ondrej Klíma. 2009. A counterexample to a conjecture concerning concatenation hierarchies. *Information Processing Letters* 110, 1 (2009), 4–7.
[6] Jorge Almeida and Ondrej Klíma. 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 (2010), 41–58.
[7] Jorge Almeida and Marc Zeitoun. 1997. The pseudovariety J is hyperdecidable. *RAIRO Informatique Théorique et Applications* 31, 5 (1997), 457–482.
[8] Mustapha Arfi. 1987. Polynomial operations on rational languages. In *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS'87), Lecture Notes in Computer Science*, Franz-Josef Brandenburg, Guy Vidal-Naquet, and Martin Wirsing (Eds.), Vol. 247. Springer, Berlin, 198–206.

[9] Mustapha Arfi. 1991. Opérations polynomiales et hiérarchies de concaténation. *Theoretical Computer Science* 91, 1 (1991), 71–84.

[10] Karl Auinger. 2010. On the decidability of membership in the global of a monoid pseudovariety. *International Journal of Algebra and Computation* 20, 2 (2010), 181–188.

[11] Bernhard Banaschewski. 1983. The Birkhoff theorem for varieties of finite algebras. *Algebra Universalis* 17, 1 (1983), 360–368.

[12] Danièle Beauquier and Jean-Éric Pin. 1989. Factors of words. In *Proceedings of the 16th International Colloquium on Automata, Languages, and Programming (ICALP'89)*. Springer, Berlin, 63–79.

[13] Danièle Beauquier and Jean-Éric Pin. 1991. Languages and scanners. *Theoretical Computer Science* 84, 1 (1991), 3–21.

[14] Mikolaj Bojańczyk. 2007. A new algorithm for testing if a regular language is locally threshold testable. *Information Processing Letters* 104, 3 (2007), 91–94.

[15] Mikołaj Bojańczyk. 2009. Factorization forests. In *Proceedings of the 13th International Conference on Developments in Language Theory (DLT'09), Lecture Notes in Computer Science*, Volker Diekert and Dirk Nowotka (Eds.), Vol. 5583. Springer, Berlin, 1–17.

[16] Mikołaj Bojańczyk and Thomas Place. 2012. Regular languages of infinite trees that are Boolean combinations of open sets. In *Proceedings of the 39th International Colloquium on Automata, Languages and Programming (ICALP'12), Lecture Notes in Computer Science*, Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer (Eds.), Vol. 7392. Springer, Berlin, 104–115.

[17] Janusz A. Brzozowski. 1976. Hierarchies of aperiodic languages. *ITA* 10, 2 (1976), 33–49.

[18] Janusz A. Brzozowski and Rina S. Cohen. 1971. Dot-depth of star-free events. *Journal of Computer and System Sciences* 5, 1 (1971), 1–16.

[19] Janusz A. Brzozowski and Robert Knast. 1978. The dot-depth hierarchy of star-free languages is infinite. *Journal of Computer and System Sciences* 16, 1 (1978), 37–55.

[20] Janusz A. Brzozowski and Imre Simon. 1971. Characterizations of locally testable events. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (SWAT'71)*. IEEE, East Lansing, MI, 166–176.

[21] Janusz A. Brzozowski and Imre Simon. 1973. Characterizations of locally testable events. *Discrete Mathematics* 4, 3 (1973), 243–271.

[22] Julius R. Büchi. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly* 6, 1-6 (1960), 66–92.

[23] Sang Cho and Dung T. Huynh. 1991. Finite-automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science* 88, 1 (1991), 99–116.

[24] Thomas Colcombet. 2010. Factorization forests for infinite words and applications to countable scattered linear orderings. *Theoretical Computer Science* 411, 4-5 (2010), 751–764.

[25] Thomas Colcombet. 2011. Green's relations and their use in automata theory. In *Proceedings of the 5th International Conference on Language and Automata Theory and Applications (LATA'11), Lecture Notes in Computer Science*, Adrian-Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide (Eds.), Vol. 6638. Springer, Berlin, 1–21.

[26] Thomas Colcombet. 2019. The factorisation forest theorem. In *Handbook of Automata Theory*, Jean-Éric Pin (Ed.). Vol. I: Theoretical Foundations. Eur. Math. Soc., Zürich. To appear.

[27] David Cowan. 1993. Inverse monoids of dot-depth two. *International Journal of Algebra and Computation* 03, 04 (1993), 411–424.

[28] Wojciech Czerwiński, Wim Martens, and Tomáš Masopust. 2013. Efficient separability of regular languages by subsequences and suffixes. In *Proceedings of the 40th International Colloquium on Automata, Languages, and Programming, (ICALP'13), Lecture Notes in Computer Science*, Fedor V. Fomin, Rusins Freivalds, Marta Z. Kwiatkowska, and David Peleg (Eds.), Vol. 7966. Springer, Berlin, 150–161.

[29] Volker Diekert and Paul Gastin. 2008. First-order definable languages. In *Logic and Automata: History and Perspectives*, Jörg Flum, Erich Grädel, and Thomas Wilke (Eds.). Texts in Logic and Games, Vol. 2. Amsterdam University Press, Amsterdam, the Netherlands, 261–306.

[30] Samuel Eilenberg. 1976. *Automata, Languages, and Machines*. Vol. B. Academic Press, Orlando, FL.

[31] Calvin C. Elgot. 1961. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society* 98, 1 (1961), 21–51.

[32] Christian Glaßer and Heinz Schmitz. 2000. Languages of dot-depth 3/2. In *Proceedings of the 17th Annual Symposium on Theoretical Aspects of Computer Science (STACS'00), Lecture Notes in Computer Science*, Horst Reichel and Sophie Tison (Eds.), Vol. 1770. Springer, Berlin, 555–566.

[33] Christian Glaßer and Heinz Schmitz. 2008. Languages of Dot-Depth 3/2. *Theory of Computing Systems* 42, 2 (2008), 256–286.

[34] Kosaburo Hashiguchi. 1983. Representation theorems on regular languages. *Journal of Computer and System Sciences* 27, 1 (1983), 101–115.

[35] Karsten Henckell. 1988. Pointlike sets: The finest aperiodic cover of a finite semigroup. *Journal of Pure and Applied Algebra* 55 (1988), 85–126.
[36] Karsten Henckell and Jean-Éric Pin. 2000. Ordered monoids and J-trivial monoids. In *Algorithmic Problems in Groups and Semigroups*, Jean-Camille Birget, Stuart Margolis, John Meakin, and Mark Sapir (Eds.). Birkhäuser, Boston, MA, 121–137.
[37] Karsten Henckell, John Rhodes, and Benjamin Steinberg. 2010. Aperiodic pointlikes and beyond. *International Journal of Algebra and Computation* 20, 2 (2010), 287–305.
[38] Peter M. Higgins. 1997. A proof of Simon's theorem on piecewise testable languages. *Theoretical Computer Science* 178, 1-2 (1997), 257–264.
[39] Peter M. Higgins. 2000. A new proof of Schützenberger's theorem. *International Journal of Algebra and Computation* 10, 02 (2000), 217–220.
[40] John M. Howie. 1991. *Automata and Languages*. Clarendon Press, Oxford.
[41] Neil Immerman. 1999. *Descriptive Complexity*. Springer, Berlin.
[42] Prateek Karandikar, Manfred Kufleitner, and Philippe Schnoebelen. 2015. On the index of Simon's congruence for piecewise testability. *Information Processing Letters* 115, 4 (2015), 515–519.
[43] Ondřej Klíma. 2011. Piecewise testable languages via combinatorics on words. *Discrete Mathematics* 311, 20 (2011), 2124–2127.
[44] Ondřej Klíma and Libor Polák. 2013. Alternative automata characterization of piecewise testable languages. In *Developments in Language Theory*. Springer, Berlin, 289–300.
[45] Robert Knast. 1983. A semigroup characterization of dot-depth one languages. *RAIRO — Theoretical Informatics and Applications* 17, 4 (1983), 321–330.
[46] Robert Knast. 1983. Some theorems on graph congruences. *RAIRO —Theoretical Informatics and Applications* 17, 4 (1983), 331–342.
[47] Manfred Kufleitner. 2008. The height of factorization forests. In *Proceedings of the 33rd International Symposium on Mathematical Foundations of Computer Science (MFCS'08), Lecture Notes in Computer Science*, Edward Ochmanski and Jerzy Tyszkiewicz (Eds.), Vol. 5162. Springer, Berlin, 443–454.
[48] Gérard Lallement. 1979. *Semigroups and Combinatorial Applications*. John Wiley & Sons, New York, NY.
[49] Leonid Libkin. 2004. *Elements of Finite Model Theory*. Springer, Berlin.
[50] Cláudio L. Lucchesi, Imre Simon, Istvan Simon, Janos Simon, and Tomasz Kowaltowski. 1979. *Aspectos Teóricos da Computação*. IMPA, São Paulo. Retrieved February 16, 2019 from http://www.impa.br/opencms/pt/biblioteca/cbm/11CBM/11_CBM_77_04.pdf.
[51] Stuart W. Margolis and Jean-Éric Pin. 1984. Power monoids and finite J-trivial monoids. *Semigroup Forum* 29 (1984), 99–108.
[52] Stuart W. Margolis and Jean-Éric Pin. 1985. Products of group languages. In *Proceedings of the 5th International Symposium on Fundamentals of Computation Theory (FCT'85), Lecture Notes in Computer Science*, Lothar Budach (Ed.), Vol. 199. Springer, Berlin, 285–299.
[53] Robert McNaughton. 1974. Algebraic decision procedures for local testability. *Mathematical Systems Theory* 8, 1 (1974), 60–76.
[54] Robert McNaughton and Seymour A. Papert. 1971. *Counter-Free Automata*. MIT Press, Cambridge, MA.
[55] Albert R. Meyer. 1969. A note on star-free events. *J. ACM* 16, 2 (1969), 220–225.
[56] Anil Nerode. 1958. Linear automaton transformations. *Proc. Amer. Math. Soc.* 9, 4 (1958), 541–544.
[57] Dominique Perrin. 1990. Finite automata. In *Formal Models and Semantics*. Elsevier, Amsterdam, 1–57.
[58] Dominique Perrin and Jean-Éric Pin. 1986. First-order logic and star-free sets. *Journal of Computer and System Sciences* 32, 3 (1986), 393–406.
[59] Jean-Éric Pin. 1984. *Variétés de Langages Formels*. Masson, Paris. English translation: 1986, Varieties of formal languages, Plenum, New York, NY.
[60] Jean-Éric Pin. 1995. Finite semigroups and recognizable languages: An introduction. In *Semigroups, Formal Languages and Groups*. Springer, Berlin, 1–32.
[61] Jean-Éric Pin. 1995. A variety theorem without complementation. *Russian Mathematics (Izvestiya Vuzov. Matematika)* 39 (1995), 74–83.
[62] Jean-Éric Pin. 1996. The expressive power of existential first order sentences of Büchi's sequential calculus. In *Proceedings of the 23rd International Colloquium on Automata, Languages, and Programming (ICALP'96)*. Springer, Berlin, 300–311.
[63] Jean-Éric Pin. 1997. Syntactic semigroups. In *Handbook of Formal Languages*. Springer, Berlin, 679–746.
[64] Jean-Éric Pin. 1998. Bridges for concatenation hierarchies. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming (ICALP'98), Lecture Notes in Computer Science*, Kim Guldstrand Larsen, Sven Skyum, and Glynn Winskel (Eds.), Vol. 1443. Springer, Berlin, 431–442.

[65] Jean-Éric Pin. 2005. Expressive power of existential first-order sentences of Büchi's sequential calculus. *Discrete Mathematics* 291, 1-3 (2005), 155–174.
[66] Jean-Éric Pin. 2011. Theme and variations on the concatenation product. In *Proceedings of the 4th International Conference on Algebraic Informatics (CAI'11), Lecture Notes in Computer Science*, Franz Winkler (Ed.), Vol. 6742. Springer, Berlin, 44–64.
[67] Jean-Éric Pin. 2013. An explicit formula for the intersection of two polynomials of regular languages. In *Developments in Language Theory*. Springer, Berlin, 31–45.
[68] Jean-Éric Pin. 2017. The dot-depth hierarchy, 45 years later. In *The Role of Theory in Computer Science, Essays Dedicated to Janusz Brzozowski*, Stavros Konstantinidis, Nelma Moreira, Rogério Reis, and Jeffrey Shallit (Eds.). World Scientific, Singapore, 177–202.
[69] Jean-Éric Pin. 2018. Mathematical Foundations of Automata Theory. (2018). https://www.irif.fr/~jep/MPRI/MPRI.html.
[70] Jean-Éric Pin and Howard Straubing. 1981. Monoids of upper triangular Boolean matrices. In *Semigroups. Structure and Universal Algebraic Problems*, S. Schwarz, G. Pollák, and O. Steinfeld (Eds.). Colloquia Mathematica Societatis Janos Bolyal, Vol. 39. North-Holland, Szeged, Hungary, 259–272.
[71] Jean-Éric Pin and Pascal Weil. 1995. Polynomial closure and unambiguous product. In *Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming (ICALP'95)*. Springer, Berlin, 348–359.
[72] Jean-Éric Pin and Pascal Weil. 1996. Profinite semigroups, Mal'cev products and identities. *Journal of Algebra* 182, 3 (1996), 604–626.
[73] Jean-Éric Pin and Pascal Weil. 1996. A Reiterman theorem for pseudovarieties of finite first-order structures. *Algebra Universalis* 35, 4 (1996), 577–595.
[74] Jean-Éric Pin and Pascal Weil. 2001. A conjecture on the concatenation product. *RAIRO Informatique Théorique* 35, 6 (2001), 597–618.
[75] Jean-Éric Pin and Pascal Weil. 2002. The wreath product principle for ordered semigroups. *Communications in Algebra* 30 (2002), 5677–5713.
[76] Jean-Éric Pin and Pascal Weil. 1997. Polynomial closure and unambiguous product. *Theory of Computing Systems* 30, 4 (1997), 383–422.
[77] Nicholas Pippenger. 1997. *Theories of Computability*. Cambridge University Press, Cambridge, UK.
[78] Thomas Place. 2015. Separating regular languages with two quantifier alternations. In *Proceedings of the 30th Annual ACM/IEEE Symposium on Logic in Computer Science(LICS'15)*. IEEE, Kyoto, Japan, 202–213.
[79] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. 2013. Separating regular languages by locally testable and locally threshold testable languages. In *Proceedings of the 33rd IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'13), Leibniz International Proceedings in Informatics*, Anil Seth and Nisheeth K. Vishnoi (Eds.), Vol. 24. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 363–375.
[80] Thomas Place, Lorijn van Rooijen, and Marc Zeitoun. 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), Lecture Notes in Computer Science*, Krishnendu Chatterjee and Jirí Sgall (Eds.), Vol. 8087. Springer, Berlin, 729–740.
[81] Thomas Place and Marc Zeitoun. 2014. 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), Lecture Notes in Computer Science*, Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias (Eds.), Vol. 8572. Springer, Berlin, 342–353.
[82] Thomas Place and Marc Zeitoun. 2014. Separating regular languages with first-order logic. In *Proceedings of the Joint Meeting of the 23rd EACSL Annual Conference on Computer Science Logic and the 29th Annual ACM/IEEE Symposium on Logic in Computer Science (CSL-LICS'14)*, Thomas A. Henzinger and Dale Miller (Eds.). ACM, New York, NY, 75:1–75:10.
[83] Thomas Place and Marc Zeitoun. 2015. Separation and the successor relation. In *Proceedings of the 32nd Annual Conference on Theoretical Aspects of Computer Science (STACS'15), Leibniz International Proceedings in Informatics*, Ernst W. Mayr and Nicolas Ollinger (Eds.), Vol. 30. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 662–675.
[84] Thomas Place and Marc Zeitoun. 2015. The tale of the quantifier alternation hierarchy of first-order logic over words. *SIGLOG News* 2, 3 (2015), 4–17. http://siglog.hosting.acm.org/wp-content/uploads/2015/10/siglog_news_5.pdf.
[85] Thomas Place and Marc Zeitoun. 2016. Separating regular languages with first-order logic. *Logical Methods in Computer Science* 12, 1 (2016), 1–30.
[86] Klaus Reinhardt. 2002. The complexity of translating logic to finite automata. In *Automata, Logics, and Infinite Games: A Guide to Current Research [outcome of a Dagstuhl seminar, February 2001], Lecture Notes in Computer Science*, Erich Grädel, Wolfgang Thomas, and Thomas Wilke (Eds.), Vol. 2500. Springer, Berlin, 231–238.

[87] Jan Reiterman. 1982. The Birkhoff theorem for finite algebras. *Algebra Universalis* 14, 1 (1982), 1–10.
[88] John Rhodes. 1999. Undecidability, automata, and pseudovarities of finite semigroups. *International Journal of Algebra and Computation* 9, 3-4 (1999), 455–474.
[89] Jacques Sakarovitch and Imre Simon. 1997. Subwords. In *Combinatorics on Words, Lothaire*. Cambridge University Press, Cambridge, UK. 105–144.
[90] Marcel Paul Schützenberger. 1955-1956. Une théorie algébrique du codage. *Séminaire Dubreil. Algèbre et Théorie Des Nombres* 9 (1955-1956), 1–24. http://eudml.org/doc/111094.
[91] Marcel Paul Schützenberger. 1965. On finite monoids having only trivial subgroups. *Information and Control* 8, 2 (1965), 190–194.
[92] Marcel Paul Schützenberger. 1976. Sur le produit de concaténation non ambigu. *Semigroup Forum* 13 (1976), 47–75.
[93] Imre Simon. 1972. *Hierarchies of events of dot-depth one*. Ph.D. Dissertation. University of Waterloo, Waterloo, Ontario.
[94] Imre Simon. 1975. Piecewise testable events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, Helmut Brakhage (Ed.). Springer, Berlin, 214–222.
[95] Imre Simon. 1990. Factorization forests of finite height. *Theoretical Computer Science* 72, 1 (1990), 65–94.
[96] Benjamin Steinberg. 2001. A delay theorem for pointlikes. *Semigroup Forum* 63, 3 (2001), 281–304.
[97] Jacques Stern. 1985. Characterizations of some classes of regular events. *Theoretical Computer Science* 35 (1985), 17–42.
[98] Jacques Stern. 1985. Complexity of some problems from the theory of automata. *Information and Control* 66, 3 (1985), 163–176.
[99] Larry J. Stockmeyer. 1974. *The complexity of decision problems in automata theory and logic*. Ph.D. Dissertation. Massachusetts Institute of Technology, Cambridge, MA. http://opac.inria.fr/record=b1000295.
[100] Larry J. Stockmeyer and Albert R. Meyer. 1973. Word problems requiring exponential time (preliminary report). In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing (STOC'73)*, Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp, and H. Raymond Strong (Eds.). ACM, New York, NY, 1–9.
[101] Howard Straubing. 1981. A generalization of the Schützenberger product of finite monoids. *Theoretical Computer Science* 13, 2 (1981), 137–150.
[102] Howard Straubing. 1985. Finite semigroup varieties of the form V ∗ D. *Journal of Pure and Applied Algebra* 36 (1985), 53–94.
[103] Howard Straubing. 1986. Semigroups and languages of dot-depth 2. In *Proceedings of the 13th International Colloquium on Automata, Languages, and Programming (ICALP'86), Lecture Notes in Computer Science*, Laurent Kott (Ed.), Vol. 226. Springer, Berlin, 416–423.
[104] Howard Straubing. 1988. Semigroups and languages of dot-depth two. *Theoretical Computer Science* 58, 1-3 (1988), 361–378.
[105] Howard Straubing. 1994. *Finite Automata, Formal Logic and Circuit Complexity*. Birkhauser, Basel, Switzerland.
[106] Howard Straubing and Denis Thérien. 1988. Partially ordered finite monoids and a theorem of I. Simon. *Journal of Algebra* 119, 2 (1988), 393–399.
[107] Howard Straubing and Pascal Weil. 1992. On a conjecture concerning dot-depth two languages. *Theoretical Computer Science* 104, 2 (1992), 161–183.
[108] Pascal Tesson and Denis Thérien. 2002. Diamonds are forever: The variety DA. In *Semigroups, Algorithms, Automata and Languages*, Gracinda M. S. Gomes, Jean Eric Pin, and Pedro V. Silva (Eds.). World Scientific, Singapore, 475–500.
[109] Denis Thérien. 1981. Classification of finite monoids: The language approach. *Theoretical Computer Science* 14, 2 (1981), 195–208.
[110] Denis Thérien. 2011. The power of diversity. In *Descriptional Complexity of Formal Systems*, Lecture Notes in Computer Science, Markus Holzer, Martin Kutrib, and Giovanni Pighizzini (Eds.), Vol. 6808. Springer, Berlin, 43–54.
[111] Denis Thérien and Alex Weiss. 1985. Graph congruences and wreath products. *Journal of Pure and Applied Algebra* 36 (1985), 205–215.
[112] Denis Thérien and Thomas Wilke. 1998. Over words, two variables are as powerful as one quantifier alternation. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, Jeffrey Scott Vitter (Ed.). ACM, New York, NY, 234–240.
[113] Wolfgang Thomas. 1982. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences* 25, 3 (1982), 360–376.
[114] Wolfgang Thomas. 1984. An application of the Ehrenfeucht-Fraissé game in formal language theory. *Mémoires de la Société Mathématique de France* 16 (1984), 11–21.
[115] Wolfgang Thomas. 1987. A concatenation game and the dot-depth hierarchy. In *Computation Theory and Logic*. Springer, Berlin, 415–426.

[116] Wolfgang Thomas. 1997. Languages, automata, and logic. In *Handbook of Formal Languages*. Springer, Berlin.
[117] Bret Tilson. 1987. Categories as algebra: An essential ingredient in the theory of monoids. *Journal of Pure and Applied Algebra* 48, 1–2 (1987), 83–198.
[118] Avraham N. Trahtman. 2001. An algorithm to verify local threshold testability of deterministic finite automata. In *Proceedings of the 4th International Workshop on Implementing Automata, Automata Implementation, Lecture Notes in Computer Science*, Oliver Boldt and Helmut Jürgensen (Eds.),Vol. 2214. Springer, Berlin, 164–173.
[119] Avraham N. Trahtman. 2001. Piecewise and local threshold testability of DFA. In *Proceedings of the 13th International Symposium on Fundamentals of Computation Theory (FCT'01), Lecture Notes in Computer Science*, Rusins Freivalds (Ed.), Vol. 2138. Springer, London, 347–358.
[120] Boris A. Trakhtenbrot. 1961. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk SSSR* 149 (1961), 326–329. In Russian.
[121] Pascal Weil. 1989. Concatenation product: A survey. In *Formal Properties of Finite Automata and Applications*. Lecture Notes in Computer Science, Jean-Éric Pin, Vol. 386. Springer, Berlin, 120–137.
[122] Pascal Weil. 1989. Inverse monoids of dot-depth two. *Theoretical Computer Science* 66, 3 (1989), 233–245.
[123] Thomas Wilke. 1999. Classifying discrete temporal properties. In *Proceedings of the 16th Annual Conference on Theoretical Aspects of Computer Science (STACS'99), Lecture Notes in Computer Science*, Christoph Meinel and Sophie Tison (Eds.), Vol. 1563. Springer, Berlin, 32–46.

Received February 2016; revised October 2018; accepted January 2019