The Regular Languages of First-Order Logic with One Alternation

Corentin Barloy
corentin.barloy@inria.fr
Univ. Lille, CNRS, INRIA, Centrale Lille, UMR 9189
CRIStAL
France

Michaël Cadilhac
michael@cadilhac.name
DePaul University
USA

Charles Paperman
charles.paperman@univ-lille.fr
Univ. Lille, CNRS, INRIA, Centrale Lille, UMR 9189
CRIStAL
France

Thomas Zeume
thomas.zeume@rub.de
Ruhr-Universität Bochum
Germany

ABSTRACT

The regular languages with a neutral letter expressible in first-order logic with one alternation are characterized. Specifically, it is shown that if an arbitrary Σ_2 formula defines a regular language with a neutral letter, then there is an equivalent Σ_2 formula that only uses the order predicate. This shows that the so-called Central Conjecture of Straubing holds for Σ_2 over languages with a neutral letter, the first progress on the Conjecture in more than 20 years. To show the characterization, lower bounds against polynomial-size depth-3 Boolean circuits with constant top fan-in are developed. The heart of the combinatorial argument resides in studying how positions within a language are determined from one another, a technique of independent interest.

KEYWORDS

automata theory, first-order logic, descriptive complexity, circuit complexity.

ACM Reference Format:
Corentin Barloy, Michaël Cadilhac, Charles Paperman, and Thomas Zeume. 2022. The Regular Languages of First-Order Logic with One Alternation. In *37th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS) (LICS '22), August 2–5, 2022, Haifa, Israel.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3531130.3533371

1 INTRODUCTION

Circuits and regular languages. Since the works of Barrington and Thérien [2, 5] in the early 1990s, regular languages have emerged as the backbone of small-depth circuit complexity. Despite being the most elementary class of languages, regular languages seem to embody the intrinsic power of circuit classes. Under a suitable notion of reduction, a lot of relevant circuit classes even admit *complete* regular languages (this is at the heart of Barrington's Theorem [2]). In addition, it seems that each natural restriction of small-depth circuits defines its *own* class of regular languages.

More precisely, consider the following families:

- AC^0_i is the class of Boolean circuits families of depth i and polynomial size,
- ACC^0_i is the same as AC^0_i but with additional modulo gates,
- TC^0_i is the same as AC^0_i but with additional *majority* gates (more than half of the inputs are 1).

The hierarchy in depth of AC^0 is known to be strict [27], but this is open for the other classes (for TC^0 this is known up to depth 3 [14]). It is however conjectured that each of these hierarchies is strict and that strictness for ACC^0 can always be witnessed by *regular* languages; in other words, as mentioned, ACC^0 is conjectured to have its *own* subset of regular languages.

Over the past 50 years, abundant literature has provided a sophisticated toolset to show separation (and sometimes decidability) of classes of *regular* languages. This toolset relies on algebraic objects that characterize the complexity of regular languages; this object satisfies some properties if and only if the language belongs to some algebraically defined class. It is thus tempting to, first, characterize the regular languages that belongs to a circuit class, and, second, extract from those descriptions *natural* circuits classes separators. This paper is focused on that first step, for a specific class of circuits (loosely speaking, AC^0_3).

Logic. Wishing to provide a guiding light, Straubing [31] presented in a succinct but beautiful way the links between circuit complexity and automata theory, and formulated a conjecture on… logics. Indeed, circuits themselves are ill-formed for statements of the form "a circuit family AC^0_i recognizes a regular language if and only if it has this specific shape." Logic came to the rescue by giving a descriptional tool for circuits. This started with the work of Barrington et al. [3] and Straubing [30] who showed that AC^0 ($= \bigcup_i \mathrm{AC}^0_i$) is equivalent to first-order logic. This means that for any AC^0 circuit family, there is a first-order formula, with quantifiers over positions, that recognizes the same language. For instance, over the alphabet $A = \{a, b, c\}$ the language $A^*ab^*aA^*$, which is in

LICS '22, August 2–5, 2022, Haifa, Israel

ACM ISBN 978-1-4503-9351-5/22/08…$15.00
https://doi.org/10.1145/3531130.3533371

AC^0_2, can be written as:

$$(\exists x, y)[x < y \wedge a(x) \wedge a(y) \wedge \quad \text{(find the 2 } a\text{s)}$$
$$(\forall z)[x < z < y \to b(z)]] \quad \text{(everything in between is a } b\text{)}$$

In this formula, we used the *numerical predicate* $<$; numerical predicates speak about the *numerical value* of positions but not their contents. The class of (languages recognized by) formulas FO[arb] is that of first-order formulas where we allow *any* numerical predicate (even undecidable ones!). The aforementioned characterization reads: $\mathrm{AC}^0 = \mathrm{FO[arb]}$. Extensions of this tight relationship between circuits and logics exist for ACC^0, TC^0, and other classes (see [31]).

Equipped with this, the characterization of the regular languages of AC^0 is given by this striking statement [30]:

$$\mathrm{FO[arb]} \cap \mathrm{Reg} = \mathrm{FO[reg]},$$

where regis the set of numerical predicates $<$, $+1$, and divisibility by constants. Straubing notes *"this phenomenon appears to be quite general"* and postulates that for well-behaved logics $\mathcal{L}$, it holds that

$$\mathcal{L}[\mathrm{arb}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{reg}].$$

This is known as the Straubing Property for $\mathcal{L}$ (simply *Central Conjecture* in [31]) and it is explicitly stated for the logics Σ_i. Straubing [31, p. 169] explains: *"This has the look of a very natural principle. It says, in effect, that the only numerical predicates we need in a sentence that defines a regular language are themselves recognized by finite automata."*

The Program for Separation in circuit complexity then becomes:

(1) Identify a logic that corresponds to the circuit class,
(2) Prove the Straubing Property for the class,
(3) Show separation over regular languages.

In this work, we apply this approach to the lower reaches of the AC^0_i hierarchy. Step 1 in the Program is covered by a result of [21]; we will be focusing on the subset Σ_i of FO of formulas with i quantifier alternations, starting with an existential one. For instance, the above formula is in $\Sigma_2[<, +1]$. The Straubing Properties for these logics are very much open: it is known to hold for Σ_1 [31] and its Boolean closure [21, 32], but no progress has been made on Straubing Properties since the end of the 1990s.

We will be mostly focusing on languages with a *neutral letter*, this means that the languages will admit a letter c that can be added or removed from words without impacting their membership in the language. This is usually not a restriction to the Program for Separation, since it is conjectured that circuit classes are separated by regular languages with neutral letters. See, in particular, the fascinating survey by Koucký [18]. Write Neut for the set of languages that have a neutral letter. The *Neutral* Straubing Property is that for a logic $\mathcal{L}$, it holds that:

$$\mathcal{L}[\mathrm{arb}] \cap \mathrm{Reg} \cap \mathrm{Neut} = \mathcal{L}[<] \cap \mathrm{Neut}.$$

Contributions. We show that Σ_2 has the Neutral Straubing Property and that $\Delta_2 = (\Sigma_2 \cap \Pi_2)$ has the Straubing Property, where Π_2 is defined as Σ_2 but with $\exists$ and $\forall$ swapped. Consequently, we exhibit some natural regular languages that separate AC^0_2 and AC^0_3.

Related work. In [13], the authors study the model of so-called programs over DA. This defines another subclass of AC^0, but it is not known to have any equivalent characterization in terms of constant depth circuits. It is in particular different from $\Sigma_2[\mathrm{arb}] \cap \Pi_2[\mathrm{arb}]$ or to the two-variable fragment of FO[arb] (follows from theorem 6 of [13]), classes that we will explore in Section 6. However, it is proven equivalent to a certain class of decision trees in [12]. They give a precise description of the regular languages computable with programs over DA, but their proof uses the algebraic structure of DA, which is not available in our setting.

The class $\Sigma_2[<]$ corresponds to the second level of the Straubing-Thérien hierarchy, an extensively studied hierarchy that is closely tied to the famous dot-depth hierarchy. A major open problem is whether the problem to decide if a given regular language belongs to a given level of this hierarchy is decidable. See the survey of Pin [22] for a modern account on this topic and the recent major progress of Place and Zeitoun [26].

Organization of the paper. We introduce circuits, logic, and a bit of algebra in Section 2. In Section 3, we introduce so-called *limits*, a classical tool in devising lower bounds against depth-3 circuits. In Section 4, we present a simple lower bound against a language in Σ_2; this serves as both a warm-up for the main proof and to identify the difficulties ahead. In Section 5, we prove our main result, that is, the Neutral Straubing Property for Σ_2. In Section 6, we derive some consequences of our main result, in particular the Straubing Property for Δ_2. We conclude in Section 7.

2 PRELIMINARIES

We assume familiarity with regular languages, logic, and circuits, although we strive to keep this presentation self-contained. We write Reg for the class of regular languages.

Words, languages, neutral letters. Following Lothaire [20], a word $u = a_1 a_2 \dots a_n$, with each a_i in an alphabet A, is a *subword* of a word v if v can be written as $v = v_0 a_1 v_1 a_2 \cdots a_n v_n$, with each v_i in A^*. We say that a language L *separates* X from Y if $X \subseteq L$ and $L \cap Y = \emptyset$. A language L has a *neutral letter* if there is a letter c such that $u \cdot c \cdot v \in L \Leftrightarrow u \cdot v \in L$ for all words u, v. We write Neut for the class of languages with a neutral letter.

Monoids, ordered monoids, morphisms. A *monoid* is a set equipped with a binary associative operation, denoted multiplicatively, with an identity element. An *idempotent* of M is an element $e \in M$ that satisfies $e^2 = e$. For an alphabet A, the set A^* is the free monoid generated by A, its identity element being the empty word. An *ordered monoid* is a monoid equipped with a partial order $\leq$ compatible with the product, i.e., $x \leq y$ implies $xz \leq yz$ and $zx \leq zy$ for any $x, y, z \in M$. Any monoid can be seen as an ordered monoid, using equality as order. An *upper set* of an ordered monoid M is a set S such that for any $x, y \in M$, if $x \in S$ and $x \leq y$ then $y \in S$. A morphism is a map $h\colon M \to N$ satisfying $h(ab) = h(a)h(b)$ and $h(1) = 1$, with $a, b \in M$ and 1 denoting the identity element of M and N.

Monoids as recognizers. An ordered monoid M *recognizes* a language $L \subseteq A^*$ if there is a morphism $h\colon A^* \to M$ and an upper set P of M such that $L = h^{-1}(P)$. The *ordered syntactic monoid* of L is the ordered monoid with the least number of elements that recognizes L; it is finite if and only if L is regular, in which case it is unique. The ordered syntactic monoid is usually defined as the quotient of

A^* by the so-called syntactic preorder induced by L: for two words x and y we have $x \leq_L y$ whenever:

$$\text{for every words } u \text{ and } v,\ uxv \in L \Rightarrow uyv \in L \ .$$

Both definitions are equivalent [24, Corollary 4.4].

Logic. We work with first-order logics recognizing languages. For instance, the formula over the alphabet $\{a, b\}$

$$(\forall x)(\exists y)[\mathrm{mod}_2(x) \vee ((y = x + 1) \wedge (a(x) \leftrightarrow b(y)))]$$

asserts that, in a given word w, for every position x there is another position y such that either x is divisible by 2 (mod_2) or y is just after x and w has different letters at x and y. The predicates mod_2 and $+1$ are examples of *numerical predicates*, i.e., they only speak about the numerical positions, not the contents of the input word. The predicates $a(\cdot)$ and $b(\cdot)$ are the *letter predicates*.

In this paper, first-order logics are specified by restricting two aspects:

- The number of quantifier alternations. We write Σ_2 for the subset of first-order formulas that can be written as $(\exists x_1, x_2, \ldots)(\forall y_1, y_2, \ldots)[\phi]$ with ϕ a quantifier-free formula, that is, Σ_2 is the set of formulas starting with an existential quantifier and alternating *once*. In Section 6.1, we will briefly mention Π_2 (defined as Σ_2 but with $\exists$ and $\forall$ swapped) and Δ_2, the set of formulas that are equivalent to *both* a Σ_2 and a Π_2 formula (this describes, a priori, fewer languages than Σ_2 or Π_2).
- The *numerical predicates* allowed. Except for a quick detour in Section 6.1, we will only be using two sets: $<$, that is, the sole order relation (e.g., $\Sigma_2[<]$) and arb the set of *all* predicates (e.g., $\Sigma_2[\mathrm{arb}]$). In the latter set, there would be predicates asserting that two positions are coprime or that a position encodes a halting Turing machine: there is no restriction whatsoever.

We associate two languages to a formula: the set of words that satisfy it and the same language enhanced with the empty word. [1] We commonly identify a class of formulas with the class of languages they recognize.

Languages of a syntactic ordered monoid. Let M be an ordered monoid. For any element $x \in M$, the *up-word problem* for M and x is the following language over M seen as an alphabet:

$$\{w \in M^* \mid w \text{ evaluates in } M \text{ to an element} \geq x\}.$$

In some precise sense, a regular language has the same complexity as the hardest of the up-word problems for its ordered syntactic monoid; in the case on $\Sigma_2[\mathrm{arb}]$, we can state:

Lemma 2.1 (From [24, Proof of Lemma 5.6]). *A regular language with a neutral letter is in $\Sigma_2[arb]$ if and only if all the up-word problems for its ordered syntactic monoid are in $\Sigma_2[arb]$.*

[1] The second language is a minor technicality needed to have a class that is closed under union (because every prenex normal form for a formula recognizing the empty word has to start with an universal quantifier).

Circuits. We will study languages computed by families of constant-depth, polynomial-size circuits consisting of unbounded fan-in $\wedge$- and $\vee$-gates. A circuit with n inputs $x_1, x_2, \ldots, x_n$ in some alphabet A can query whether any input contains or not any given letter in A. The *depth* of the circuit is the maximal number of gates appearing on a path from an input to the output. A circuit family is an infinite set $(C_n)_{n\geq 0}$ where the circuit C_n has n inputs and one output gate; a word w is deemed accepted if the circuit $C_{|w|}$ outputs 1 when w is placed as input. We identify classes of circuits with the class of languages they recognize.

For a circuit C we visualize the inputs on *top* and the one output gate at the *bottom*.[2] The *top fan-in* of C is the maximum fan-in of the gates that receive an input letter directly. We say that C is a $\exists\forall\exists$ circuit if it is layered with a bottom $\vee$-gate with $\wedge$-gates as inputs, each of these having $\vee$-gates as inputs.

We let Σ_2 be the class of families of $\exists\forall\exists$ circuits of polynomial-size and bounded top fan-in. Remark that Σ_2 is equivalent to disjunction of conjunction of Boolean circuit of constant-depth and bounded fan-in (NC^0). Indeed, we can transform NC^0 circuits into CNF of small size and merge their $\wedge$-gate with the previous layer.

In the literature, this circuit family is also called $\Sigma\Pi\Sigma(k)$ in [10] and $\Sigma_2^{\mathrm{poly},k}$ in [9], where k is the top fan-in. It is a subclass of AC^0_3, the class of polynomial-size, depth-3 circuits, a class we will discuss in Section 6.2. The name "Σ_2 circuit family" is all the more justified that:

Lemma 2.2 (From [21, Proposition 11]). *A language is recognized by a $\Sigma_2[arb]$ formula if and only if it is recognized by a Σ_2 circuit family.*

Note that the precise statement in [21] is about the Boolean closure of $\Sigma_2[\mathrm{arb}]$, but the very same proof can be applied by removing the complement. We will often exploit this equivalence without pointing at this lemma.

A decidable characterization of $\Sigma_2[<]$. Let $x, y \in M$; we say that y is a *subword* of x if there are two words $w_x, w_y \in M^*$ that evaluate, using M's product, to x and y, respectively, and w_y is a subword of w_x. The following is a characterization of the languages in $\Sigma_2[<]$ that is due to Pin and Weil [23], and we use a version proposed by Bojańczyk:

Theorem 2.3 (From [6]). *A regular language is in $\Sigma_2[<]$ if and only if its ordered syntactic monoid M is such that for any $x, y \in M$ such that x is an idempotent[3] and y a subword of x, it holds that*

$$x \leq xyx \ .$$

Equivalently, this could be worded over languages directly: a regular language L is in $\Sigma_2[<]$ if and only if for any three words w_1, w_2, w_3 that map to the same idempotent in M and v a subword of w_2, if a word $w_0 \cdot w_1 w_2 w_3 \cdot w_4$ is in L, for some words w_0, w_4, then so is $w_0 \cdot w_1 v w_3 \cdot w_4$. We will allude to this wording in some proofs in the Consequences section (Section 6.2).

[2] The literature has been flip-flopping between putting the inputs at the bottom or at the top, with a semblance of stability for "top" achieved in the late 90s. This explains that some references mention "bottom fan-in."

[3] This is usually written by letting x be any element, and considering x^ω, which is the unique idempotent that is a power of x; we simplify the presentation slightly by simply requiring x to be an idempotent.

3 LIMITS AND LOWER BOUNDS AGAINST Σ_2[arb]

We present a tool that has been used several times to show lower bounds against depth-3 circuits, in particular in [15]. The following definition is attributed to Sipser therein:

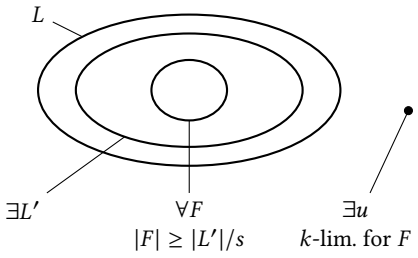
Definition 3.1 (From [28]). Let F be a set of words, all of same length n, and $k > 0$. A *k-limit* for F is a word u of length n such that for any set of k positions, a word in F matches u on all these positions. In symbols, u satisfies:

$$(\forall P \subseteq [n].|P| = k)(\exists v \in F)(\forall p \in P)\left[u_p = v_p\right].$$

Naturally, we will be interested in k-limits that fall outside of F, otherwise finding k-limits is trivial. In fact, we will consider sets F that are included in a subset of a target language, and find k-limits outside of the target language itself. We include a short proof of the following statement for completeness and because it makes the statement itself more readily understandable.

Lemma 3.2 (From [15, Lemma 2.2]). *Let L be a set of words all of same length n and C be a $\exists\forall\exists$ circuit that accepts at least all the words of L. Let k be the top fan-in of C and s its size.*

Assume there is a subset $L' \subseteq L$ such that for any $F \subseteq L'$ of size at least $|L'|/s$ there is a k-limit for F that does not belong to L. Then C accepts a word outside of L. The hypothesis can be represented graphically as:



Proof. At the bottom of C, we have an $\vee$-gate of fan-in at most s that receives the result of some $\wedge$-gates. By counting, one of these $\wedge$-gates should accept a subset F of L' of size at least $|L'|/s$; we will now focus on that gate. Let $u \notin L$ be the k-limit for F that exists by hypothesis. Consider an $\vee$-gate that feeds into the $\wedge$-gate under consideration. This $\vee$-gate checks the contents of a subset $P \subseteq [n]$ of at most k positions of the input. By hypothesis, there is a word v in F that matches u on all the positions in P, hence the $\vee$-gate cannot distinguish between u and v and must output 1 (true) on u as v must be accepted. This holds for all the $\vee$-gates feeding into the $\wedge$-gate under consideration, hence the $\wedge$-gate must accept u, and so does C. □

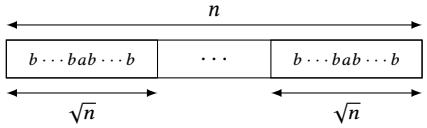
Corollary 3.3. *Let L be a language and write L_n for the subset of words of length n in L. Assume that for any $k, d \in \mathbb{N}$, there is an $n \geq 2$ and a subset $L' \subseteq L_n$ such that every subset $F \subseteq L'$ of size at least $|L'|/n^d$ admits a k-limit outside of L. Then L is not in Σ_2[arb].*

Proof. For a contradiction, assume there is a Σ_2 circuit family for L, with top fan-in k and size n^d for $n \geq 2$. Let n be the value provided by the hypothesis, then the circuit C for L_n satisfies the hypotheses of Lemma 3.2, hence C accepts a word outside of L, a contradiction. □

4 WARM-UP: $K = (ac^*b + c)^* \notin \Sigma_2$[arb]

In this section, we follow the approach of Håstad, Jukna, and Pudlák [15] to show the claim of the section title. We present it in a specific way that will help us stress the commonality and the differences of this approach with our main proof.

To show the claim of the section title, we consider a slightly different language. For any n that is a perfect square, we let Good_n be the set of words of length n over $\{a, b\}$ of the following shape:



In words, a word is in Good_n if it can be decomposed into $\sqrt{n}$ *blocks* of length $\sqrt{n}$, such that each of them has exactly one a. We let $\text{Good} = \bigcup_n \text{Good}_n$.

Lemma 4.1. *If K is in Σ_2[arb], then so is* Good.

Proof. This is easier to see on circuits, so assume there is a Σ_2 circuit family for K. For n a perfect square, we design a circuit for Good_n. On any input, we convert the b's to c's and insert a b every $\sqrt{n}$ positions; we call this the *expansion* of the input word. For instance, with $n = 9$, the input *abb bab bba* is expanded to *accb cacb ccab*. Clearly, if the input word is in Good_n, then its expansion is in K. Conversely, if a block of the input had two a's, the expansion will not add a b in between, so the expansion is not in K; similarly, if a block of the input contains only b's, it will be expanded to only c's sandwiched between two b's, and the expansion will not be in K (in the case where the block containing only b's is the first one, the expansion starts with $c \cdots cb$, again putting the expansion outside of K).

Thus a circuit for Good_n can be constructed by computing the expansion (this only requires wires and no gates), then feeding that expansion to a circuit for K. If the circuit family for K were in Σ_2, so would the circuit family for Good. □

We use Corollary 3.3 to show that $\text{Good} \notin \Sigma_2$[arb]. Let then $k, d \in \mathbb{N}$. The value of L' in Corollary 3.3 will simply be Good_n, and we show:

Lemma 4.2. *If n is large enough, any subset $F \subseteq \text{Good}_n$ with $|F| > k^{\sqrt{n}}$ has a k-limit outside of Good_n. This holds in particular if $|F| \geq |\text{Good}_n|/n^d$.*

Proof. We rely on the *Flower Lemma*, a combinatorial lemma that is a relaxation of the traditional Sunflower Lemma. We first need to introduce some vocabulary.

We consider families $\mathcal{F}$ containing sets of size s for some s. The *core* of the family is the set $Y = \bigcap_{S \in \mathcal{F}} S$. The *coreless* version of $\mathcal{F}$ is the family $\mathcal{F}_Y = \{S \setminus Y \mid S \in \mathcal{F}\}$. A set S *intersects* a family $\mathcal{F}$ if all the sets of $\mathcal{F}$ have a nonempty intersection with S. Finally, a *flower* with p petals is a family $\mathcal{F}$ of size p with core Y such that any set which intersects $\mathcal{F}_Y$ is of size at least p.

Lemma 4.3 (Flower Lemma [17, Lemma 6.4]). *Let $\mathcal{F}$ be a family containing sets of cardinality s and $p \geq 1$ be an integer. If $|\mathcal{F}| > (p-1)^s$, then there is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ that is a flower with p petals.*

To apply this lemma, consider the mapping τ from words in Good_n to $2^{[n]}$ that lists all the positions where a word has an a. For instance, with $n = 9$, $\tau(bba\,abb\,bab) = \{3, 4, 8\}$. For any word w in Good_n, $\tau(w)$ is of size $\sqrt{n}$. We let $\mathcal{F} = \{\tau(w) \mid w \in F\}$.

We now apply the lemma with $s = \sqrt{n}$ and $p = k + 1$. Since $|\mathcal{F}| = |F|$, we can apply the lemma on $\mathcal{F}$ and obtain a subfamily $\mathcal{F}'$ that is a flower with $k + 1$ petals. Let Y be its core. Consider the word u of length n over $\{a, b\}$ which has a's exactly at the positions in Y. Then:

- u is outside of Good_n. Indeed, $|Y| < \sqrt{n}$, since it is the intersection of distinct sets of size $\sqrt{n}$. Hence one of the blocks of u will contain only b's, putting it outside of Good_n.
- u is a k-limit. Let P be a set of k positions, we will find a word that is mapped to $\mathcal{F}'$ that matches u on P. If a position in P points to an a in u, then every word in $\mathcal{F}'$ has an a at that position (by construction, since this position would belong to the core Y). Therefore, as we are looking for a matching word in $\mathcal{F}'$, we can find a word that is matching on the b's only. So we assume that P contains only positions on which u is b. Since $|P|$ is k, it cannot intersect $\mathcal{F}'$, hence there is a set $S \in \mathcal{F}'$ such that $S \cap P = \emptyset$. The set S is thus $\tau(w)$ for a word $w \in F$ that has a b on all positions in P. This word w thus matches u on P, concluding the proof of the main statement.

The "in particular" part is implied by the fact that, for n large enough:

$$\frac{|\text{Good}_n|}{n^d} = \frac{\sqrt{n}^{\sqrt{n}}}{n^d} \geq k^{\sqrt{n}}.$$

□

Theorem 4.4. *The language $K = (ac^*b + c)^*$ is not in $\Sigma_2[arb]$.*

Proof. Corollary 3.3 applied on Good, using Lemma 4.2, implies that $\text{Good} \notin \Sigma_2[\text{arb}]$. Lemma 4.1 then asserts that K cannot be in $\Sigma_2[\text{arb}]$ either. □

5 THE REGULAR LANGUAGES WITH A NEUTRAL LETTER NOT IN $\Sigma_2[<]$ ARE NOT IN $\Sigma_2[\mathbf{arb}]$

The proof of the statement of the section title is along two main steps:

Section 5.1. We will start with a regular language with a neutral letter $L \notin \Sigma_2[<]$. Since it is not in $\Sigma_2[<]$, there are $x, y \in M$ that falsify the equations of Theorem 2.3. We use these witnesses to build a up-word problem T of the ordered syntactic monoid of L and show that it lies outside of $\Sigma_2[\text{arb}]$, implying that $L \notin \Sigma_2[\text{arb}]$ by Lemma 2.1.

To show T out of $\Sigma_2[\text{arb}]$, we identify (Section 5.1.1) a subset of well-behaved words of T, and make some simple syntactical changes (in Section 5.1.2) on them so that they look like words in Good, in a similar fashion as the "expansions" of Lemma 4.1. The argument used in Lemma 4.1 then needs to be refined, as we do not have that any word outside of Good comes from a word outside of T. We will define a set Bad of words that look like words in Good except for *one* block that contains only b's; Lemma 4.1 is then worded as: if K is in $\Sigma_2[\text{arb}]$, then there is a $\Sigma_2[\text{arb}]$ language that separates Good from Bad (Lemma 5.1).

Section 5.2. We show that no language of $\Sigma_2[\text{arb}]$ can separate Good from Bad. We thus need to provide a statement in the spirit of Lemma 4.2. We first write good and bad words in a succinct ("packed") way, as words in $[\sqrt{n}]^{\sqrt{n}}$, the i-th letter being some value v if the original word had the a of its i-th block in position v (Section 5.2.1). We then translate the notion of k-limit to packed words (Lemma 5.3). Finally, we provide a measure of how diverse a set of (packed) good words is (Definition 5.4), and show that such a set is either not diverse and small (Lemma 5.5), or diverse and admits a k-limit (Lemma 5.6). Our term for "not diverse" will be *tangled*, referring to the fact that there is a strong correlation between the contents of positions within words.

5.1 If a language not in $\Sigma_2[<]$ is in $\Sigma_2[\mathbf{arb}]$, we can separate Good from Bad with a language in $\Sigma_2[\mathbf{arb}]$

5.1.1 Target up-word problem and some of its words. For the rest of this section, let $L \subseteq A^*$ be a regular language with a neutral letter that lies outside of $\Sigma_2[<]$ and let M be its ordered syntactic monoid. Since L is not in $\Sigma_2[<]$, there are elements $x, y \in M$ such that $x \not\leq xyx$ with x an idempotent and y a subword of x. Let T be the up-word problem of M for x. Clearly, any word of M^* that evaluates to xyx does *not* belong to T.

By hypothesis, y is thus also a subword of x; this provides us with words that evaluate to x and y of the shape:

$$x_1 y_1 \dots x_t y_t \text{ evaluates to } x, \quad y_1 \dots y_t \text{ evaluates to } y,$$

where each x_i and y_i are letters in M. (Note that we can use the identity element of M as needed to fill up and ensure we have as many x_i's as y_i's.)

Let $n \in \mathbb{N}$ be a perfect square (whose value is meant to be taken large enough later). We define $\sqrt{n} + 1$ words of length $t\sqrt{n} + t$ over M:

- For $i \in [\sqrt{n}]$,
$$x^{(i)} = \left(1^{i-1} x_1 1^{\sqrt{n}-i} \cdot y_1\right) \cdots \left(1^{i-1} x_t 1^{\sqrt{n}-i} \cdot y_t\right).$$
Here $1 \in M$ is the neutral element of M. Note that these words evaluate to x.
- Additionally, we consider the word $1^{\sqrt{n}} y_1 \cdots 1^{\sqrt{n}} y_t$, which evaluates to y, and we simply write y for it. Note that y can be obtained by replacing all the letters x_j by 1 from any word $x^{(i)}$.

Call T-*good* a concatenation of $\sqrt{n}$ words of the form $x^{(i)}$ sandwiched between two words $x^{(1)}$ (the 1 is arbitrary), and T-*bad* a word obtained by changing, in a T-good word, *exactly one* of the words $x^{(i)}$ to y (but for the $x^{(1)}$ at the beginning and end). By construction, any T-good word evaluates to x in M, so belongs to T, while any T-bad word evaluates to xyx, hence does not belong to T. Note that if we had switched *two* blocks of a T-good word to y, we would not be able to say whether it belonged to T or not.

5.1.2 T-good, T-bad to Good *and* Bad. The T-good and T-bad words contain a lot of redundant information, for instance $x^{(i)}$

is of length $t\sqrt{n}+t$, while all the information it really contains is $i \in [\sqrt{n}]$. Recall the set Good_n of Section 4 which contains all words of length n over $\{a,b\}$ that can be divided into $\sqrt{n}$ blocks of length $\sqrt{n}$, each containing a single a. Again, we let Good be all such words, of any perfect square length. Define similarly Bad_n as the set of words that are like Good_n except for *one* block which has only b's, and let $\mathrm{Bad} = \bigcup_n \mathrm{Bad}_{n^2}$.

In the next lemma, we show that we can modify, using only wires in a circuit, words over $\{a,b\}$ so that if they are in Good they become T-good, and if they are in Bad they become T-bad. This modification is simple enough that we can take a Σ_2 circuit family for T, apply the modification at the top of each circuit, and still have a circuit family in Σ_2; the resulting circuit family separates Good from Bad:

Lemma 5.1. *If $T \in \Sigma_2[arb]$, then there is a $\Sigma_2[arb]$ language that separates* Good *from* Bad.

Proof. As in Lemma 4.1, this is easier seen on circuits: we design a circuit for inputs of length n over $\{a,b\}$ that separates Good from Bad.

Consider the first block of $\sqrt{n}$ letters of the input. We replicate it t times, with the i-th replication changing b's to 1 and a's to x_i. We then concatenate these and add y_i between the i-th and $(i+1)$-th replication. For instance, $b^7ab^{\sqrt{n}-8}$ would turn into:

$$\left(1^7x_11^{\sqrt{n}-8}\cdot y_1\right)\cdots\left(1^7x_t1^{\sqrt{n}-8}\cdot y_t\right) = x^{(8)}.$$

In particular, if the block were all b's, we would obtain the word y, which has no letter x_j.

We can do this to each block of $\sqrt{n}$ letters, concatenate the resulting words, then add the word $x^{(1)}$ at the beginning and the end. Note that these operations can be done with only wires, with no gates involved.

If the input word is in Good, then the word produced is T-good, hence in T. If it was in Bad, then the resulting word would be T-bad, hence would lie outside of T. This shows that the desired circuit can be constructed using the above wiring followed by the circuit for T for inputs of length $(t\sqrt{n}+t)(2+\sqrt{n})$. Since t is a constant and depends solely on L, the resulting circuit is of polynomial size and of the correct shape. □

5.2 No language in $\Sigma_2[\mathbf{arb}]$ separates Good from Bad

Note that this section is independent from the previous one. We will now rely on Corollary 3.3 to show that any Σ_2[arb] language L that accepts all of Good must accept a word in Bad. To apply Corollary 3.3, *from this point onward* we let $k, d \in \mathbb{N}$, and set n to be a large enough value that depends only on k and d. The role of L' in the statement of Corollary 3.3 will be played by Good_n and we will build k-limits belonging to Bad_n, which we call *bad k-limits*. The reader may check that the statements of the forthcoming Lemma 5.5 and Lemma 5.6 conclude the proof.

5.2.1 Packed words. Words in Good_n and Bad_n can be described by the position of the letter a in each block of size $\sqrt{n}$. We make this explicit, by seeing $[\sqrt{n},\bot] = [\sqrt{n}] \cup \{\bot\}$ as an alphabet, and working with words in $[\sqrt{n},\bot]^{\sqrt{n}}$. We call these words *packed* and will use Greek letters λ, μ, ν for them; we also call the letter at some position in packed words its *contents* at this position, only to stress that we are working with packed words. We define the natural functions to pack and unpack words:

- unpack: $[\sqrt{n},\bot] \to \{a,b\}^{\sqrt{n}}$ maps i to $b^{i-1}ab^{\sqrt{n}-i}$ and $\bot$ to $b^{\sqrt{n}}$. This extends naturally to *words* over $[\sqrt{n},\bot]$.
- pack: $\{a,b\}^* \to [\sqrt{n},\bot]^*$ is the inverse of unpack. We will use that function on sets of words too, with the natural meaning.

Example 5.2. With $n = 9$, $\mathrm{pack}(abb\,bbb\,bab) = 1\bot2$, and $\mathrm{unpack}(31\bot) = bba\,abb\,bbb$.

We can now rephrase the notion of k-limit using packed words:

Lemma 5.3. *Let $F \subseteq \mathrm{Good}_n$ and define $\Phi = pack(F)$. If μ is a packed word that has the following properties, then unpack(μ) is a bad k-limit for F:*

1. *There is a word $\nu \in \Phi$ that differs on a single position i with μ, at which μ has contents $\bot$:*
$$\mu_i = \bot \wedge (\forall j \neq i)[\nu_j = \mu_j].$$
2. *For every set $C \subseteq [\sqrt{n}]$ of contents that contains ν_i and every set $P \subseteq [\sqrt{n}] \setminus \{i\}$ of positions such that $|C| + |P| = k$, there is a word $\lambda \in \Phi$ whose contents at position i is not in C and that matches ν on P:*
$$\lambda_i \notin C \wedge (\forall p \in P)[\lambda_p = \nu_p].$$

Proof. Write u for unpack(μ). That $u \in \mathrm{Bad}$ is immediate from Property 1: u is but a word v of Good in which one block was set to all b's.

We now show that u is a k-limit. Let T be a set of k positions, we split T into two sets:

- T' is the set of positions p that do not belong to the i-th block of u, that is, they do not satisfy $\lceil p/\sqrt{n}\rceil = i$. We let P be each of the elements of T' divided by $\sqrt{n}$, that is, for any $p \in T'$ we add $\lceil p/\sqrt{n}\rceil$ to P.
- T'' is the set of positions that *do* fall in the i-th block. Note that u only has b's at the positions of T''. We let C be that set, modulo $\sqrt{n}$, that is, for any $p \in T''$, we add $p \bmod \sqrt{n}$ to C or $\sqrt{n}$ if this value is 0.

First, if $\mu_i \notin C$, then T indicates positions of u that have the same letters as in $\mathrm{unpack}(\nu) \in F$, so a word of F matches u over T, as required. We thus assume next that $\mu_i \in C$.

Let $\lambda \in \Phi$ be the word given by Property 2 for C and P, we claim that $w = \mathrm{unpack}(\lambda)$ matches u on the positions of T, concluding the proof.

First note that the i-th block of w has its a in a position that is not in T'', hence w matches u on T''. Consider next any position $p \in T'$ and write j for the block in which p falls (i.e., $j = \lceil p/\sqrt{n}\rceil$). Since $\mu_j = \lambda_j$ by hypothesis, the j-th block of u and w are the same, hence $u_p = w_p$. □

5.2.2 Tangled sets of good words are small, nontangled ones have a bad k-limit. Consider an $F \subseteq \mathrm{Good}_n$. To find a bad k-limit for F, we need a lot of diversity in F; see in particular Prop. 2 of Lemma 5.3. Hence having some given contents at a given position in a word

of Φ should not force too many other positions to have a specific value. We make this notion formal:

Definition 5.4. Let $F \subseteq \text{Good}_n$ and $\Phi = \text{pack}(F)$. The *entailment relation* of Φ is relating sets of pairs (i, c) of position/contents in words of Φ. Let us say that a word and a pair position/contents (i, c) *agree* if the contents at position i of the word is c, and that a word and a set of such pairs agree if the word agrees with *each* pair. We say that a set of pairs is an i-set if all its pairs have i as position.

A set S of pairs position/contents *entails* an i-set D if all words in Φ that agree with S also agree with some pair of D (in which case the pair is unique); additionally, the position i should not appear in S:

$$\begin{aligned}&(\nexists c)[(i,c) \in S] \wedge \\ &(\forall \mu \in \Phi)[(\forall (j,d) \in S)[\mu_j = d] \rightarrow \\ &\qquad (\exists (i,c) \in D)[\mu_i = c]].\end{aligned}$$

The set F is said to be *k-tangled* if for any word $\mu \in \Phi$ and any position $i \in [\sqrt{n}]$, there is an i-set of pairs of size $\leq k$ that contains (i, μ_i) and that is entailed by a set of size k that agrees with μ. In other words, every position of μ is entailed by a subset of its positions. We drop the k in k-tangled if it is clear from context.

Lemma 5.5. *Let $F \subseteq \text{Good}_n$. If F is k-tangled, then $|F| < \sqrt{n}^{2k\sqrt{n}/(2k+1)}$. In particular, $|F| < |\text{Good}_n|/n^d$.*

Proof. Assume F is k-tangled and let $\Phi = \text{pack}(F)$. We show that every word in Φ can be fully described in Φ by fully specifying a portion $k/(k+1)$ of its positions and encoding the contents of each of the other $1/(k+1)$ positions with elements from $[k]$. That is, if two words in Φ have the same such description, they are the same, hence Φ cannot be larger than the number of such descriptions. We first show this property, then derive the numerical implication on $|F|$. We moreover make the technical assumption that $\sqrt{n}$ is a multiple of $k+1$ to not bother with integrability issues.

Let $\mu \in \Phi$, we construct iteratively a set K of positions that we will fully specify and a set K^+ of positions that are restricted when setting the positions in K.

First consider the pair $(1, \mu_1)$. Since F is tangled, there is an 1-set containing $(1, \mu_1)$, entailed by a set S that agrees with μ. We add to K the positions of S and to K^+ the positions of S *and* position 1.

We now iterate this process: Take a pair (i, μ_i) such that $i \notin K^+$. There is an i-set containing (i, μ_i) that is entailed by a set S that agrees with μ. Let S' be the set of positions of S that are not in K^+. We add S' to K, and $S' \cup \{i\}$ to K^+. Note that the size increase for K^+ is one more than that for K. We continue iterating until all positions appear in K^+.

We now bound the size of K at the end of the computation. For each iteration, in the worst case, we need to add k positions to K to obtain $k+1$ new positions in K^+ (this is the worst case in the sense that this is the worst ratio of the number of positions we need to pick in K to the number of positions that are put in K^+). In that case, after s steps, we have $|K| = sk$ and $|K^+| = sk + s$. Thus when $|K^+| = \sqrt{n}$, that is, when no more iterations are possible, we have:

$$sk + s = \sqrt{n} \Rightarrow s = \frac{\sqrt{n}}{k+1}.$$

This shows that $|K| \leq k\sqrt{n}/(k+1)$.

We now turn to describing the word μ using K. We first provide all the contents of μ at positions in K; call Z the set of pairs position/contents of μ that correspond to positions in K. We mark the positions of K as *specified*, and carry on to specify the other positions in a deterministic fashion.

We first fix an arbitrary order on sets of pairs of position/contents. We iterate through all the subsets of Z of size k, in order. For each such subset S, we consider, in order again, the subsets D that are entailed by S. Assume D is an i-set; if position i is already specified, we do nothing, otherwise, we describe which element of D is (i, μ_i) using an integer in $[k]$, and mark i as specified. We proceed until all the subsets of Z have been seen, at which point, by construction of K, all the positions will have been specified. As claimed, given Z and the description of which elements in sets D correspond to the correct contents, we can reconstruct μ.

Summing up, to fully describe μ, we had to specify the positions of K (one of $\binom{\sqrt{n}}{k\sqrt{n}/(k+1)}$ possible choices), their contents (one of $\sqrt{n}^{k\sqrt{n}/(k+1)}$ possible choices), and for each position not specified by K, we needed to provide an integer in $[k]$ (one of $k^{\sqrt{n}/(k+1)}$ possible choices). This shows that:

$$\begin{aligned}|F| &\leq \binom{\sqrt{n}}{k\sqrt{n}/(k+1)} \cdot \sqrt{n}^{k\sqrt{n}/(k+1)} \cdot k^{\sqrt{n}/(k+1)} \\ &< 2^{\sqrt{n}} \cdot 2^{(\sqrt{n}/(k+1))(k \log \sqrt{n})} \cdot 2^{(\sqrt{n}/(k+1)) \log k} \\ &= 2^{(\sqrt{n}/(k+1))((k+1)+k \log \sqrt{n} + \log k)} \\ &\leq 2^{(\sqrt{n}/(k+1))((k+\frac{1}{3k}) \log \sqrt{n})} \qquad \text{(n large enough)} \\ &= \sqrt{n}^{(k+\frac{1}{3k})\sqrt{n}/(k+1)} \leq \sqrt{n}^{2k\sqrt{n}/(2k+1)}.\end{aligned}$$

The "in particular" part is a consequence of the fact that, for n large enough:

$$\frac{|\text{Good}_n|}{n^d} = \frac{\sqrt{n}^{\sqrt{n}}}{n^d} \geq \sqrt{n}^{2k\sqrt{n}/(2k+1)}.$$

□

Lemma 5.6. *Let $F \subseteq \text{Good}_n$. If F is not k-tangled, then F has a bad k-limit.*

Proof. Write $\Phi = \text{pack}(F)$. That F is not tangled means that there is a word $\nu \in \Phi$ and a position i such that for any set of pairs position/contents S of size k and any i-set D of size at most k that contains (i, ν_i), S does not entail D. We define μ to be the word ν but with μ_i set to $\perp$. We show that unpack(μ) is a bad k-limit using Lemma 5.3. Property 1 therein is true by construction, so we need only show Property 2.

Let $C \subseteq [\sqrt{n}]$ with $\mu_i \in C$ and $P \subseteq [\sqrt{n}] \setminus \{i\}$ with $|C| + |P| = k$. We add some more arbitrary positions in P so that $|P| = k$, avoiding i. Define:

$$S = \{(p, \mu_p) \mid p \in P\},\ D = \{(i, c) \mid c \in C\}.$$

By hypothesis, since $(i, \mu_i) \in D$, S does not entail D. This means that there is a word $\lambda \in \Phi$ such that S and λ agree, but $\lambda_i \notin C$. This is the word needed for Property 2 of Lemma 5.3, concluding the proof. □

Corollary 5.7. *No $\Sigma_2[arb]$ language can separate* Good *from* Bad.

Proof. We apply Corollary 3.3 on any language L that separates Good from Bad. We let $k, d \in \mathbb{N}$, and n large enough; L' in the statement of Corollary 3.3 is set to Good_n. We are then given a set F of size at least $|\text{Good}_n|/n^d$ and Lemma 5.5 shows that F is not tangled. Lemma 5.6 then implies that F has a bad k-limit, which is therefore not in L. Corollary 3.3 concludes that F is not in $\Sigma_2[\text{arb}]$, showing the statement. □

Theorem 5.8 (Neutral Straubing Property for Σ_2).

$$\Sigma_2[arb] \cap Reg \cap Neut \subseteq \Sigma_2[<].$$

Proof. Let $L \notin \Sigma_2[<]$ regular with a neutral letter and T be the language defined in Section 5.1.1. Corollary 5.7 and Lemma 5.1 imply that T cannot be in $\Sigma_2[\text{arb}]$, and in turn, Lemma 2.1 shows that L cannot be in $\Sigma_2[\text{arb}]$. □

6 CONSEQUENCES

6.1 Life without neutral letters

The *regular numerical predicates*, denoted reg, are the numerical predicates $+1$, $<$, and for any $p > 0$, mod_p which is true of a position if it is divisible by p. The term "regular" stems from the fact that these are the properties on numerical positions that automata can express. It is in particular possible to express all unary predicates of the form "the position is congruent to r modulo p" for every integer p and $r < p$ (see the proof of theorem III.2.1 in [31]). Recall that the Straubing Property for a logic $\mathcal{L}$ expresses that $\mathcal{L}[\text{arb}] \cap \text{Reg} = \mathcal{L}[\text{reg}]$.

The Straubing Property does not immediately imply the Neutral Straubing Property; for this, one would need in addition that $\mathcal{L}[\text{reg}] \cap \text{Neut} \subseteq \mathcal{L}[<]$. This latter property is called the *Crane Beach Property* of $\mathcal{L}[\text{reg}]$, and also stems from the natural idea that if a language has a neutral letter, then numerical predicates do not provide any useful information. Albeit natural, this property is false for FO[arb] [4], but Theorem 5.8 shows that $\Sigma_2[\text{reg}]$ does have the Crane Beach Property.

Relying on Theorem 4.4 and some results from [8], we can show:

Theorem 6.1 (Straubing Property of Δ_2).

$$\Delta_2[arb] \cap Reg = \Delta_2[reg].$$

Proof. The proof structure is as follows: We first show that $\Delta_2[\text{arb}] \cap \text{Reg}$ has some closure properties, so that it is a so-called *lm-variety*. We then show that $\Delta_2[\text{reg}]$ recognizes precisely all the regular languages definable with a first-order formula with *two* variables that uses regas numerical predicates; that class of languages is denoted $\text{FO}^2[\text{reg}]$. We then rely on the following lemma, where $K = (ac^*b + c)^*$:

Lemma 6.2 (From [8, Lemma 8]). *If an lm-variety of regular languages $\mathcal{V}$ satisfies:*

$$FO^2[reg] \subseteq \mathcal{V} \subseteq FO[arb] \text{ and } K \notin \mathcal{V}$$

then $\mathcal{V} = FO^2[reg]$.

Since $\Delta_2[\text{arb}] \cap \text{Reg}$ satisfies the hypotheses, it is equal to $\text{FO}^2[\text{reg}] = \Delta_2[\text{reg}]$, concluding the proof.

$\Delta_2[\text{arb}] \cap \text{Reg}$ *is an lm-variety.* We ought to first define *lm-variety*. If for a morphism $h\colon A^* \to B^*$ there is a k such that $h(A) \subseteq B^k$, we call h an *lm-morphism*, where lm stands for *length-multiplying*. Given a language L and a letter a, the *left quotient* of L by a is the set $a^{-1}L = \{v \mid av \in L\}$. The *right quotient* La^{-1} is defined symmetrically. An *lm-variety of languages* is a set of languages closed under the Boolean operations, quotient, and inverse lm-morphisms.

Since Reg is an lm-variety of languages, it is sufficient to show that $\Delta_2[\text{arb}]$ is too; this is not hard:

- Boolean operations: Both $\Sigma_2[\text{arb}]$ and $\Pi_2[\text{arb}]$ formulas are closed under Boolean OR and AND, hence the classes of languages they recognize are closed under union and intersection, and so is $\Delta_2[\text{arb}]$. Also, since the negation of a $\Sigma_2[\text{arb}]$ formula is a $\Pi_2[\text{arb}]$ formula, and vice versa, $\Delta_2[\text{arb}]$ is closed under complement.
- Quotient: We show that $\Sigma_2[\text{arb}]$ is closed under quotient; the proof is the same for $\Pi_2[\text{arb}]$, and this implies that $\Delta_2[\text{arb}]$ is also closed under quotient. Let $L \in \Sigma_2[\text{arb}]$ and a be a letter. Consider the circuit for the words of length n in L. We can hardwire the first letter to a; the resulting circuit has $n-1$ inputs, and recognizes a word w if and only if $aw \in L$. The family thus obtained recognizes $a^{-1}L$. The argument for right quotient is similar.
- lm-morphisms: Again, we show this holds for $\Sigma_2[\text{arb}]$, the proof for $\Pi_2[\text{arb}]$ being similar, and these two facts imply closure under lm-morphisms of $\Delta_2[\text{arb}]$. Let $L \in \Sigma_2[\text{arb}]$ over the alphabet B and h be an lm-morphism such that $h(A) \subseteq B^k$ for some k. Consider the circuit for the words of L of length kn for some n. Given a word in A^n, we can use NC^0-circuits (see section 2) to map each input letter $a \in A$ to $h(a)$, and we can feed the resulting word to the circuit for L. A word $w \in A^n$ is thus accepted if and only if $h(w) \in L$, hence the circuit family thus defined recognizes $h^{-1}(L)$.

$\Delta_2[\text{reg}]$ *and* **$\text{FO}^2[\text{reg}]$** *recognize the same languages.* We show the inclusion from left to right, the converse being similar. We rely on the fact that $\text{FO}^2[<, +1] = \Delta_2[<, +1]$, a result due to Thérien and Wilke [34, Theorem 7]. The rest of our proof is fairly simple: we put the information given by the mod_p predicates within the alphabet, show that this information is easily checked with one universal quantifier if we have regpredicates, and that if the modular predicates are put within the alphabet, the only required predicates to express our $\Delta_2[\text{reg}]$ formula are $<$ and $+1$. We then rely on the equivalence of FO^2 and Δ_2 over these predicates to conclude.

Formally, let $\phi \in \Delta_2[\text{reg}]$ be a formula over the alphabet A. Let P be the set of moduli used in ϕ, that is, the predicate mod_p appears in ϕ if and only if $p \in P$. The *P-annotation* of a word $w \in A^*$ is the word in $(A \times 2^P)^*$ that indicates, for each position i, the set of moduli in P that divide i. In other words, the P-annotation of $w = w_1 w_2 \cdots w_n$, with each $w_i \in A$, is the word of length n whose i-th letter is:

$$\begin{pmatrix} w_i \\ \{p \in P \mid p \text{ divides } i\} \end{pmatrix} \in (A \times 2^P).$$

Let $\mathcal{W}$ be the set of words in $(A \times 2^P)^*$ that are P-annotations. Then:

- There is a $\phi' \in \Delta_2[<,+1]$ such that the words of $\mathcal{W}$ that satisfy ϕ' are precisely the P-annotations of words that satisfy ϕ. The formula ϕ' is simply the formula ϕ in which each predicate $\mathrm{mod}_p(x)$ is replaced with the property "p belongs to the 2^P part of the letter at position x," which can be written as a simple disjunction. If the input word is a P-annotation, $\mathrm{mod}_p(x)$ is indeed equivalent to that property.
  Since $\mathrm{FO}^2[<,+1] = \Delta_2[<,+1]$, there is a formula ψ' of $\mathrm{FO}^2[<,+1]$ that accepts the same language as ϕ'.
- The formula ψ' is over the enriched alphabet $(A \times 2^P)$, we therefore have to go back to a formula ψ of $\mathrm{FO}^2[\mathrm{reg}]$ over A. To do so, we convert every letter predicate $\binom{a}{Q} \in (A \times 2^P)$ on the position x into the formula "$a(x) \wedge_{p \in Q} \mathrm{mod}_p(x)$ ". Thus ψ is equivalent to ϕ, thanks to ψ' being equivalent to ϕ on $\mathcal{W}$.

□

The proof of the previous statement hinged on the characterization given by Lemma 6.2. To show the (nonneutral) Straubing Property of $\Sigma_2[\mathrm{arb}]$, a similar statement will need to be proved. To make the similarity more salient, we reword Lemma 6.2 as the equivalent statement:

Lemma 6.3. *If an lm-variety of regular languages $\mathcal{V}$ satisfies:*

$$\Delta_2[reg] \subseteq \mathcal{V} \subseteq FO[arb] \text{ and } \mathcal{V} \cap Neut \subseteq \Delta_2[<].$$

then $\mathcal{V} = \Delta_2[reg]$.

Proof. The second part of the proof of Theorem 6.1 shows that $\mathrm{FO}^2[\mathrm{reg}] = \Delta_2[\mathrm{reg}]$, while the same was already known if $<$ is the only available numerical predicate, so we can freely swap FO^2 for Δ_2 in the statement of Lemma 6.2. We need to show that the hypothesis $\mathcal{V} \cap \mathrm{Neut} \subseteq \Delta_2[<]$ is equivalent to $K \notin \mathcal{V}$. For the left-to-right implication, it is known [19] that $K \notin \mathrm{FO}^2[<] = \Delta_2[<]$. For the right-to-left implication, we use the hypothesis that $\mathcal{V} \subseteq \mathrm{FO}[\mathrm{arb}]$: [8, Theorem 9] shows that $\mathcal{V} \subseteq \mathrm{FO}[\mathrm{arb}]$ and $K \notin \mathcal{V}$ implies that $\mathcal{V} \subseteq \mathrm{FO}^2[\mathrm{reg}]$ and [8, Proposition 15] asserts that $\mathrm{FO}^2[\mathrm{reg}] \cap \mathrm{Neut} \subseteq \mathrm{FO}^2[<]$. Hence $\mathcal{V} \cap \mathrm{Neut} \subseteq \mathrm{FO}^2[<]$ and replacing FO^2 with Δ_2 concludes the proof. □

A *positive lm-variety* is defined just as an lm-variety, but without requiring closure under complement. The statement we need for Σ_2 thus reads:

Conjecture 6.4. *If a positive lm-variety of regular languages $\mathcal{V}$ satisfies:*

$$\Sigma_2[reg] \subseteq \mathcal{V} \subseteq FO[arb] \text{ and } \mathcal{V} \cap Neut \subseteq \Sigma_2[<]$$

then $\mathcal{V} = \Sigma_2[reg]$.

If the conjecture held, then using $\mathcal{V} = \Sigma_2[\mathrm{arb}] \cap \mathrm{Reg}$ would show the Straubing Property for $\Sigma_2[\mathrm{arb}]$. This conjecture is harder than its Δ_2 counterpart as the algebraic structure of $\Sigma_2[<]$ and its interactions with the regular predicates are poorly understood.

6.2 On the fine structure of AC^0

For this section, we use notations similar to [21] on circuit complexity (these correspond to the classes BC^0_i therein):

- AC^0_i is the class of polynomial-size, depth-i Boolean circuit families, where all the circuits in a family have the same kind of output gate (AND or OR);
- $\widehat{\mathrm{AC}^0_i}$ is defined similarly, the only difference being that the input gates of the circuits are allowed to compute any function of at most a constant number of positions of the input string. This class is equivalent to $\Sigma_i[\mathrm{arb}] \cup \Pi_i[\mathrm{arb}]$ (the i here is the number of quantifiers blocks, so that there are $i-1$ alternations between $\exists$ and $\forall$).

The implied hierarchies interleave in a strict way:

$$\mathrm{AC}^0_1 \subsetneq \widehat{\mathrm{AC}^0_1} \subsetneq \mathrm{AC}^0_2 \subsetneq \cdots$$

The strictness of the hierarchy was independently obtained by [21] and [9], both relying on previous bounds by Håstad [16], but with very different approaches. However, the question of finding *explicit* languages that separate this hierarchy is still open, to the best of our knowledge. We make some modest progress towards this:

Theorem 6.5. *Let A be the alphabet with three letters a, b and c. The language $K' = K \cdot bc^*b \cdot A^*$ is in $AC^0_3 \setminus \widehat{AC^0_2}$.*

Proof. ***Upper bound.*** It is easily seen that a word is in the language K if and only if:

- Between every two a's there is a b, and vice versa;
- The first (resp. last) nonneutral letter of the word is a (resp. b).

Each of these statements can be written as an AND of ORs; for the first one, it is easier seen on the complement: we have an $\vee$-gate that selects two positions p_1, p_2 and checks that there is an a at both positions and only c's in between. Thus K can be written as an AND of these, and so K has a circuit of depth exactly 2.

To build a circuit for K', we start with an $\vee$-gate that selects two positions p_1, p_2, and checks with an $\wedge$-gate that they both contain a b and that only c's appear in between. We add as input to that $\wedge$-gate the inputs of the $\wedge$-gate for K where the positions considered are restricted to be smaller than p_1. This thus correctly checks that the prefix up to p_1 is in K, and that it is followed by a word starting with bc^*b.

Lower bound. If $K' \in \widehat{\mathrm{AC}^0_2}$, then K' is either in $\Sigma_2[\mathrm{arb}]$ or $\Pi_2[\mathrm{arb}]$.

Assume $K' \in \Sigma_2[\mathrm{arb}]$, then $K' \in \Sigma_2[<]$ by Theorem 5.8. We show that $K' \notin \Sigma_2[<]$ using the equations provided by Theorem 2.3 with the wording appearing after the theorem. First, the word $(ab)^2$ is mapped to an idempotent: if $(ab)^2$ appears in a word, we can repeat it any number of times without changing membership to K'. Also, the word ba appears as a subword of $(ab)^2$. However $(ab)^6 \cdot bb \in K$, but $(ab)^2(ba)(ab)^2 \cdot bb \notin K'$, showing that $K' \notin \Sigma_2[<]$.

If we assume that $K' \in \Pi_2[\mathrm{arb}]$, we have to show that the complement of K' is not in $\Sigma_2[<]$. This time, we pick $(ab)^3$ as the idempotent, and bba as the subword. Then $(ab)^9 \notin K'$ but $(ab)^3(bba)(ab)^3 \in K'$, hence the complement of K' is not in $\Sigma_2[<]$. □

The language K itself appears very often in the literature pertaining to the fine separation of small circuit classes [7, 19]. This is no surprise: The language is the first of the family of *bounded-depth Dyck languages*. These are the well-parenthesized expressions that nest no more than a fixed value:

$$D_1^{(0)} = c^*$$
$$D_1^{(1)} = K = (ac^*b + c)^*,$$
$$D_1^{(i)} = (aD_1^{(i-1)}b + c)^*.$$

(Here, a can be interpreted as "opening parenthesis" and b as "closing.")

Saliently, these languages separate the class $\Sigma_i[<]$ from $\Sigma_{i+1}[<]$, with $D_1^{(i-1)}$ belonging to the latter but not the former [7]. It is open whether these languages also separate the $\Sigma_i[\mathrm{arb}]$ hierarchy, and thus the AC^0_i one. We can show that:

Theorem 6.6. $D_1^{(2)} \in AC^0_3 \setminus \widehat{AC^0_2}$.

Proof. ***Upper bound.*** It can be shown [7, Lemma 4^+] that:

$$D_1^{(2)} = KbA^* \cup A^*bc^*bKbA^* \cup A^*aK \cup A^*aKac^*aA^*.$$

We can use an $\vee$-gate to select the positions of the letters a, b mentioned in that expression, and then either use an $\wedge$-gate to verify that only c's appear between them, or use the AND of ORs circuit for K from the previous proof to check that a word of K appears between two positions.

Lower bound. We again use the wording appearing after Theorem 2.3. As in the previous proof, we need to show that neither $D_1^{(2)}$ nor its complement are in $\Sigma_2[<]$. For the language itself, we pick $(ab)^3$ as the word mapping to an idempotent and bba as the subword. We indeed have that $(ab)^9 \in D_1^{(2)}$, but $(ab)^3(bba)(ab)^3 \notin D_1^{(2)}$. For the complement, we pick $(ab)^2$ as the word mapping to an idempotent and aab as the subword; they satisfy $(ab)^6 \cdot b \notin D_1^{(2)}$ but $(ab)^2(aab)(ab)^2 \cdot b \in D_1^{(2)}$. □

7 CONCLUSION

We have shown the Neutral Straubing Property for Σ_2:

$$\Sigma_2[\mathrm{arb}] \cap \mathrm{Reg} \cap \mathrm{Neut} = \Sigma_2[<] \cap \mathrm{Neut}.$$

To do so, we developed a new lower bound technique against circuits of depth 3 that relies on the *entailment* relation of a language. This relation indicates how dependent the positions within a language are on one another. We believe that this relation may be exploited to show Straubing Properties at higher levels of the Σ_i hierarchy.

Dropping the neutral-letter restriction from our main result is an interesting task. Although it would not imply a much stronger statement in terms of circuits, it is still a stain on the clean statement that is the Straubing Property. We note that Conjecture 6.4 is implied by the so-called *locality* property of the algebraic counterpart of $\Sigma_2[<]$; showing locality is a notoriously hard problem in algebraic language theory (see, e.g., [1, 29, 33, 35]).

Showing Straubing Properties remains a challenging and wide open problem. If we are to follow our approach for Σ_i, $i \geq 3$, it requires in particular a decidable characterization of the form of Theorem 2.3. This is usually provided by an equational characterization of the class, and although the general shape of the equations for each $\Sigma_i[<]$ is known [26, Theorem 6.2], they do not readily imply decidability; in fact, the decidability of $\Sigma_i[<]$ for $i \geq 5$ is open [25].

An outstanding example of the connection between the Straubing Property of a circuit class and its computational power is given in [8, 19]: $\mathrm{FO}^2[\mathrm{arb}]$ has the Straubing Property if and only if addition cannot be computed with a linear number of gates. This latter question, pertaining to the precise complexity of addition, was asked, in particular, by Furst, Saxe, and Sipser [11, Section 5]. Note that even though $\Delta_2[\mathrm{reg}] = \mathrm{FO}^2[\mathrm{reg}]$, this is not known to hold for the set of arbitrary numerical predicates.

ACKNOWLEDGMENTS

We wish to thank Nikhil Balaji and Sébastien Tavenas for fruitful discussions. We also thanks the anonymous reviewers for their detailed and accurate remarks that allowed us to improve the quality of the paper. The work of the last author was supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation), grant 448468041.

REFERENCES

[1] Jorge Almeida. 1996. A Syntactical Proof of Locality of da. *Int. J. Algebra Comput.* 6, 2 (1996), 165–178. https://doi.org/10.1142/S021819679600009X
[2] David A. Mix Barrington. 1989. Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1. *J. Comput. Syst. Sci.* 38, 1 (1989), 150–164. https://doi.org/10.1016/0022-0000(89)90037-8
[3] David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. 1992. Regular Languages in NC^1. *J. Comput. Syst. Sci.* 44, 3 (1992), 478–499. https://doi.org/10.1016/0022-0000(92)90014-A
[4] David A. Mix Barrington, Neil Immerman, Clemens Lautemann, Nicole Schweikardt, and Denis Thérien. 2005. First-order expressibility of languages with neutral letters or: The Crane Beach conjecture. *J. Comput. Syst. Sci.* 70, 2 (2005), 101–127. https://doi.org/10.1016/j.jcss.2004.07.004
[5] David A. Mix Barrington and Denis Thérien. 1988. Finite monoids and the fine structure of NC^1. *J. ACM* 35, 4 (1988), 941–952. https://doi.org/10.1145/48014.63138
[6] Mikolaj Bojanczyk. 2009. Factorization Forests. In *Developments in Language Theory, 13th International Conference, DLT 2009, Stuttgart, Germany, June 30 - July 3, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5583)*, Volker Diekert and Dirk Nowotka (Eds.). Springer, 1–17. https://doi.org/10.1007/978-3-642-02737-6_1
[7] Janusz A. Brzozowski and Robert Knast. 1978. The Dot-Depth Hierarchy of Star-Free Languages is Infinite. *J. Comput. Syst. Sci.* 16, 1 (1978), 37–55. https://doi.org/10.1016/0022-0000(78)90049-1
[8] Michaël Cadilhac and Charles Paperman. 2021. The Regular Languages of Wire Linear AC^0. (Dec. 2021). https://hal.archives-ouvertes.fr/hal-03466451 Submitted to Acta Informatica, special issue for the 70th birthday of Klaus-Jörn Lange.
[9] Liming Cai, Jianer Chen, and Johan Håstad. 1998. Circuit Bottom Fan-In and Computational Power. *SIAM J. Comput.* 27, 2 (1998), 341–355. https://doi.org/10.1137/S0097539795282432
[10] Ning Ding, Yanli Ren, and Dawu Gu. 2017. PAC Learning Depth-3 AC^0 Circuits of Bounded Top Fanin. In *International Conference on Algorithmic Learning Theory, ALT 2017, 15-17 October 2017, Kyoto University, Kyoto, Japan (Proceedings of Machine Learning Research, Vol. 76)*, Steve Hanneke and Lev Reyzin (Eds.). PMLR, 667–680. http://proceedings.mlr.press/v76/ding17a.html
[11] Merrick L. Furst, James B. Saxe, and Michael Sipser. 1984. Parity, Circuits, and the Polynomial-Time Hierarchy. *Math. Syst. Theory* 17, 1 (1984), 13–27. https://doi.org/10.1007/BF01744431
[12] Ricard Gavaldà and Denis Thérien. 2003. Algebraic Characterizations of Small Classes of Boolean Functions. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03)*. Springer-Verlag, Berlin, Heidelberg, 331–342.
[13] Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. 2017. The Power of Programs over Monoids in DA. In *42nd International Symposium on Mathematical*

Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark (LIPIcs, Vol. 83), Kim G. Larsen, Hans L. Bodlaender, and Jean-François Raskin (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2:1–2:20. https://doi.org/10.4230/LIPIcs.MFCS.2017.2

[14] András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. 1993. Threshold Circuits of Bounded Depth. *J. Comput. Syst. Sci.* 46, 2 (1993), 129–154. https://doi.org/10.1016/0022-0000(93)90001-D

[15] Johan Håstad, Stasys Jukna, and Pavel Pudlák. 1995. Top-Down Lower Bounds for Depth-Three Circuits. *Comput. Complex.* 5, 2 (1995), 99–112. https://doi.org/10.1007/BF01268140

[16] John Håstad. 1989. Almost Optimal Lower Bounds for Small Depth Circuits. *Adv. Comput. Res.* 5 (1989), 143–170.

[17] Stasys Jukna. 2011. *Extremal Combinatorics - With Applications in Computer Science* (second ed.). Springer. https://doi.org/10.1007/978-3-642-17364-6

[18] Michal Koucký. 2009. Circuit Complexity of Regular Languages. *Theory Comput. Syst.* 45, 4 (2009), 865–879. https://doi.org/10.1007/s00224-009-9180-z

[19] Michal Koucký, Pavel Pudlák, and Denis Thérien. 2005. Bounded-depth circuits: separating wires from gates. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, Harold N. Gabow and Ronald Fagin (Eds.). ACM, 257–265. https://doi.org/10.1145/1060590.1060629

[20] M. Lothaire. 1997. *Combinatorics on words* (second ed.). Cambridge University Press.

[21] Alexis Maciel, Pierre Péladeau, and Denis Thérien. 2000. Programs over semigroups of dot-depth one. *Theor. Comput. Sci.* 245, 1 (2000), 135–148. https://doi.org/10.1016/S0304-3975(99)00278-9

[22] 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 O. Shallit (Eds.). World Scientific, 177–202. https://doi.org/10.1142/9789813148208_0008

[23] Jean-Éric Pin and Pascal Weil. 1997. Polynomial Closure and Unambiguous Product. *Theory Comput. Syst.* 30, 4 (1997), 383–422. https://doi.org/10.1007/BF02679467

[24] Jean-Éric Pin. 1995. A variety theorem without complementation. *Russian Mathematics (Izvestija vuzov. Matematika)* 39 (1995), 80–90.

[25] Thomas Place. 2018. Separating regular languages with two quantifier alternations. *Log. Methods Comput. Sci.* 14, 4 (2018). https://doi.org/10.23638/LMCS-14(4:16)2018

[26] Thomas Place and Marc Zeitoun. 2019. Going Higher in First-Order Quantifier Alternation Hierarchies on Words. *J. ACM* 66, 2 (2019), 12:1–12:65. https://doi.org/10.1145/3303991

[27] Michael Sipser. 1983. Borel Sets and Circuit Complexity. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, David S. Johnson, Ronald Fagin, Michael L. Fredman, David Harel, Richard M. Karp, Nancy A. Lynch, Christos H. Papadimitriou, Ronald L. Rivest, Walter L. Ruzzo, and Joel I. Seiferas (Eds.). ACM, 61–69. https://doi.org/10.1145/800061.808733

[28] Michael Sipser. 1984. A Topological View of Some Problems in Complexity Theory. In *Mathematical Foundations of Computer Science 1984, Praha, Czechoslovakia, September 3-7, 1984, Proceedings (Lecture Notes in Computer Science, Vol. 176)*, Michal Chytil and Václav Koubek (Eds.). Springer, 567–572. https://doi.org/10.1007/BFb0030341

[29] Howard Straubing. 1985. Finite semigroup varieties of the form $V \star D$. *Journal of Pure and Applied Algebra* 36 (1985), 53–94. https://doi.org/10.1016/0022-4049(85)90062-3

[30] Howard Straubing. 1991. Constant-Depth periodic Circuits. *Int. J. Algebra Comput.* 1, 1 (1991), 49–88. https://doi.org/10.1142/S0218196791000043

[31] Howard Straubing. 1994. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston. https://doi.org/10.1007/978-1-4612-0289-9

[32] Howard Straubing. 2001. Languages Defined with Modular Counting Quantifiers. *Inf. Comput.* 166, 2 (2001), 112–132. https://doi.org/10.1006/inco.2000.2923

[33] Howard Straubing. 2015. A new proof of the locality of R. *Int. J. Algebra Comput.* 25, 1-2 (2015), 293–300. https://doi.org/10.1142/S0218196715400111

[34] Denis Thérien and Thomas Wilke. 1998. Over Words, Two Variables Are as Powerful as One Quantifier Alternation. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, Jeffrey Scott Vitter (Ed.). ACM, 234–240. https://doi.org/10.1145/276698.276749

[35] Bret Tilson. 1987. Categories as algebra: An essential ingredient in the theory of monoids. *Journal of Pure and Applied Algebra* 48, 1 (1987), 83–198. https://doi.org/10.1016/0022-4049(87)90108-3