The power of programs over monoids in DA

Nathan Grosshans[1], Pierre McKenzie[2], and Luc Segoufin[3]

1 **LSV, CNRS, ENS Paris-Saclay (Cachan, France) and DIRO, Université de Montréal (Montréal, Canada), nathan.grosshans@polytechnique.edu.**
This work is supported by grants from Digiteo France.
2 **DIRO, Université de Montréal (Montréal, Canada), mckenzie@iro.umontreal.ca.**
3 **INRIA and LSV, ENS Paris-Saclay (Cachan, France), luc.segoufin@inria.fr.**

Abstract

The program-over-monoid model of computation originates with Barrington's proof that it captures the complexity class NC^1. Here we make progress in understanding the subtleties of the model. First, we identify a new tameness condition on a class of monoids that entails a natural characterization of the regular languages recognizable by programs over monoids from the class. Second, we prove that the class known as **DA** satisfies tameness and hence that the regular languages recognized by programs over monoids in **DA** are precisely those recognizable in the classical sense by morphisms from **QDA**. Third, we show by contrast that the well studied class of monoids called **J** is not tame and we exhibit a regular language, recognized by a program over a monoid from **J**, yet not recognizable classically by morphisms from the class **QJ**. Finally, we exhibit a program-length-based hierarchy within the class of languages recognized by programs over monoids from **DA**.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases Programs over monoids, DA, lower-bounds

Digital Object Identifier 10.4230/LIPIcs.MFCS.2017.2

1 Introduction

A program of range n on alphabet Σ over a finite monoid M is a sequence of pairs (i, f) where i is a number between 1 and n and f is a function assigning an element of the monoid M to any letter of Σ. This program assigns to each word $w_1 w_2 \cdots w_n$ the monoid element obtained by multiplying out in M the elements $f(w_i)$, one per pair (i, f), in the order of the sequence. When associated with a subset F of M as an acceptance set, a program naturally defines the language L_n of words of length n to which it assigns a monoid element in F. A program sequence $(P_n)_{n\in\mathbb{N}}$ then defines the language formed by the union of the L_n.

Such sequences became the focus of attention when Barrington [3] made the striking discovery, in fact partly observed earlier [17], that polynomial length program sequences over the group S_5 and sequences of Boolean circuits of polynomial size, logarithmic depth and constant fan-in (defining the complexity class NC^1) recognize precisely the same languages.

A flurry of work followed. After all, a program over M is a mere generalization of a morphism from Σ^* to M and recognition by a morphism equates with acceptance by a finite automaton. Given the extensive algebraic automata theory available at the time [15, 11, 22], it was to be a matter of a few years before the structure of NC^1 got elucidated by algebraic means.

The "optimism period" produced many significant results. The classes $\mathsf{AC}^0 \subset \mathsf{ACC}^0 \subseteq \mathsf{NC}^1$ were characterized by polynomial length programs over the aperiodic, the solvable, and all monoids respectively [3, 6]. More generally for any variety **V** of monoids (a variety

being the undisputed best fit with the informal notion of a natural class of monoids) one can define the class $\mathcal{P}(\mathbf{V})$ of languages recognized by polynomial length programs over a monoid drawn from $\mathbf{V}$. In particular, if $\mathbf{A}$ is the variety of aperiodic monoids, then $\mathcal{P}(\mathbf{A})$ characterizes the complexity class AC^0 [6].

But sadly, the optimism period ended: although partial results in restricted settings were obtained, the holy grail of reproving significant circuit complexity results and forging ahead by recycling the deep theorems afforded by algebraic automata theory never materialized. The test case for the approach was to try to prove, independently from the known combinatorial arguments [1, 12, 14] and those based on approximating circuits by polynomials over some finite field [24, 27], that $\mathcal{P}(\mathbf{A})$ does not contain the parity language MOD_2, i.e., that $\mathrm{MOD}_2 \notin \mathsf{AC}^0$. But why has this failed?

The answer of course is that programs are much more complicated than morphisms: programs can read an input position more than once, in non-left-to-right order, possibly assigning a different monoid element each time. Linear-length programs can indeed trivially recognize non-regular languages. In the classical theory, any two varieties provably recognize distinct classes of languages [11, 22]. In the theory of recognition by polynomial length programs (we will speak then of p-recognition), distinct varieties can yield the same class, as do, for instance, any two varieties of monoids $\mathbf{V}$ and $\mathbf{W}$ that each contain a simple non-Abelian group, for which $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W}) = \mathsf{NC}^1$ [18, Theorem 4.1].

To further illustrate the subtle behavior of programs, consider the variety of monoids known as $\mathbf{J}$. $\mathbf{J}$ is the variety of monoids generated by the syntactic monoids of the languages such that membership can be decided by looking for the presence or absence of certain *subwords*, where u is a subword of v if u can be obtained from v by removing some letters of v [26]. It follows that $\mathbf{J}$ is unable to recognize the language defined by the regular expression $(a+b)^*ac^+$. Yet $(a+b)^*ac^+$ is p-recognizable over $\mathbf{J}$. To see this consider the language L of all words having ca as a subword but not the subwords cca, caa and cb. L is therefore recognized by a morphism φ to some monoid M of $\mathbf{J}$, i.e. $L = \varphi^{-1}(F)$ for some $F \subseteq M$. A program of range n over M can recognize the words of length n of $(a+b)^*ac^+$ by using the following trick: reading the input letters in the order $2, 1, 3, 2, 4, 3, 5, 4, \ldots, n, n-1$, assigning to each letter read its image in M by φ and using the same acceptance set F as for L. For instance, on input $abacc$ the program outputs $\varphi(baabcacc)$ which is in F, while on inputs $abbcc$ and $abacca$ the program outputs respectively $\varphi(babbcbcc)$ and $\varphi(baabcaccac)$ which are not in F.

Our paper is motivated by the need to better understand such subtle behaviors of polynomial length programs over monoids. Quite a bit of knowledge on such programs has accumulated over nearly thirty years (consider [5, 20, 21, 16, 28] beyond the references already mentioned). Yet, even within the realm of questions that do not hold pretense to major complexity class separations, gaps remain.

One beaming such gap concerns the variety of monoids $\mathbf{DA}$. The importance of $\mathbf{DA}$ in algebraic automata theory and its connections with other fields are well established (see [30] for an eloquent testimony). $\mathbf{DA}$ is a relatively "small" variety, well within the variety of aperiodic monoids. One could argue that "small" varieties will be sensitive to duplications and rearrangements in the order in which input letters are read by a program. Presumably in part for that reason, programs over $\mathbf{DA}$ have seemingly not been successfully analyzed prior to our work.

Our main result is a characterization of the regular languages recognized by polynomial length programs over $\mathbf{DA}$. We show that $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg}$ is precisely the class $\mathcal{L}(\mathbf{QDA})$ of languages recognized classically by morphisms in quasi-$\mathbf{DA}$ ($\mathbf{QDA}$). A surjective morphism

φ from Σ^* to a finite monoid M is quasi-**DA** if, though M might not be in **DA**, there is a number k such that the image by φ of all words over Σ whose length is a multiple of k forms a submonoid of M which is in **DA**. In this particular case, this statement is equivalent to the fact that the only power added by p-recognition relative to classical recognition through monoids in **DA** is the ability to count input positions modulo a constant. Our proof shows, independently from [1, 12, 14, 24, 27], that, for regular languages, p-recognition over **DA** does not distort the algebraic properties of the underlying morphisms beyond adding the ability for fixed modulo counting on lengths. (This is precisely the statement, once extended to the variety of all aperiodic monoids, that would yield the elusive semigroup-theoretic proof that $\mathrm{MOD}_2 \notin \mathsf{AC}^0$.)

Our main result builds upon a statement of independent interest, namely, that any variety of monoids **V** that fulfills an appropriate tameness condition satisfies $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$. The new tameness condition (see Definition 2) differs subtly but fundamentally from a similar notion developed for semigroups by Péladeau, Straubing and Thérien [21] and also studied in the case of monoids by Péladeau [20] and later Tesson [28] in their respective Ph.D. theses, and thus requires a separate treatment here. Proving that **DA** indeed satisfies our tameness condition is the main technical difficulty behind our characterization of $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg}$.

We further consider $\mathcal{P}(\mathbf{DA})$. With $\mathcal{C}_k$ the class of languages recognized by programs of length $\mathrm{O}(n^k)$ over **DA**, we prove that $\mathcal{C}_1 \subset \mathcal{C}_2 \subset \cdots \subset \mathcal{C}_k \subset \cdots \subset \mathcal{P}(\mathbf{DA})$ forms a strict hierarchy. We also relate this hierarchy to another algebraic characterization of **DA** and exhibit conditions on $M \in \mathbf{DA}$ under which any program over M can be rewritten as an equivalent subprogram (made of a subsequence of the original sequence of instructions) of length $\mathrm{O}(n^k)$, refining a result by Tesson and Thérien [29].

Our final result concerns the variety **J**. Observing that the regular language $(a+b)^*ac^+$ mentioned above is not recognizable by a morphism in **QJ**, we conclude that **J** is not a tame variety. Be it the chicken or the egg, this lack of tameness "explains" the unexpected power of $\mathcal{P}(\mathbf{J})$ witnessed in our example above. Furthermore, since **J** is a p-variety of monoids in the sense of [21, 20, 28], **J** explicitly shows that our notion of tameness and that of [21, 20, 28] differ.

Organization of the paper. In Section 2 we define programs over varieties of monoids, p-recognition by such programs and the necessary algebraic background. The definition of tameness for a variety **V** is given in Section 3 with our first result showing that regular languages recognized by $\mathcal{P}(\mathbf{V})$ are included in $\mathcal{L}(\mathbf{QV})$ when **V** is tame; we also quickly discuss the case of **J**, which isn't tame. We show that **DA** is tame in Section 4. Finally, Section 5 contains the hierarchy results about $\mathcal{P}(\mathbf{DA})$.

2 Preliminaries

This section is dedicated to introducing the mathematical material used throughout this paper. Concerning algebraic automata theory, we only quickly review the basics and refer the reader to the two classical references of the domain by Eilenberg [10, 11] and Pin [22].

General notations. Let $i, j \in \mathbb{N}$ be two natural numbers. We shall denote by $[\![i, j]\!]$ the set of all natural numbers $n \in \mathbb{N}$ verifying $i \leq n \leq j$. We shall also denote by $[i]$ the set $[\![1, i]\!]$.

Words and languages. Let Σ be a finite alphabet. We denote by Σ^* the set of all finite words over Σ. We also denote by Σ^+ the set of all finite non empty words over Σ, the empty word being denoted by ε. A *language over* Σ is a subset of Σ^*. A language is *regular* if it

can be defined using a regular expression. Given a language L, its *syntactic congruence* $\sim_L$ is the relation on Σ^* relating two words u and v whenever for all $x, y \in \Sigma^*$, $xuy \in L$ if and only if $xvy \in L$. It is easy to check that $\sim_L$ is an equivalence relation and a congruence for concatenation. The *syntactic morphism of* L is the mapping sending any word u to its equivalence class in the syntactic congruence.

The *quotient of a language* L *over* Σ *relative to the words* u *and* v is the language, denoted by $u^{-1}Lv^{-1}$, of the words w such that $uwv \in L$.

Monoids, semigroups and varieties. A *semigroup* is a set equipped with an associative law that we will write multiplicatively. A *monoid* is a semigroup with an identity. An example of a semigroup is Σ^+, the free semigroup over Σ. Similarly Σ^* is the free monoid over Σ. A *morphism* φ *from a semigroup* S *to a semigroup* T is a function from S to T such that $\varphi(xy) = \varphi(x)\varphi(y)$ for all $x, y \in S$. A morphism of monoids additionally requires that the identity is preserved. A semigroup T is a *subsemigroup* of a semigroup S if T is a subset of S and is equipped with the restricted law of S. Additionally the notion of submonoids requires the presence of the identity. The *Cartesian (or direct) product* of two semigroups is simply the semigroup given by the Cartesian product of the two underlying sets equipped with the Cartesian product of their laws.

A language L over Σ is *recognized by a monoid* M if there is a morphism h from Σ^* to M and a subset F of M such that $L = h^{-1}(F)$. We also say that *the morphism* h *recognizes* L. It is well known that a language is regular if and only if it is recognized by a finite monoid. Actually, as $\sim_L$ is a congruence, the quotient $\Sigma^*/\sim_L$ is a monoid, called *the syntactic monoid of* L, that recognizes L via the syntactic morphism of L. The syntactic monoid of L is finite if and only if L is regular. The quotient $\Sigma^+/\sim_L$ is analogously called *the syntactic semigroup of* L.

A *variety of monoids* is a class of finite monoids closed under submonoids, Cartesian product and morphic images. A *variety of semigroups* is defined similarly. When dealing with varieties, we consider only finite monoids and semigroups.

An element s of a semigroup is *idempotent* if $ss = s$. For any finite semigroup S there is a positive number (the minimum such number), *the idempotent number of* S, often denoted ω, such that for any element $s \in S$, s^ω is idempotent.

A variety can be defined by means of identities [25]. The variety is then the class of monoids such that each of them has all its elements satisfy the identities. For instance, the variety of aperiodic monoids $\mathbf{A}$ can be defined as the class of monoids satisfying the identity $x^\omega = x^{\omega+1}$, where x ranges over the elements of the monoid while ω is the idempotent power of the monoid. The variety of monoids $\mathbf{DA}$ is defined by the identity $(xy)^\omega = (xy)^\omega x (xy)^\omega$. The variety of monoids $\mathbf{J}$ is defined by the identity $(xy)^\omega = (xy)^\omega x = y(xy)^\omega$.

Quasi and locally V languages, modular counting and predecessor. If S is a semigroup we denote by S^1 the monoid S if S is already a monoid and $S \cup \{1\}$ otherwise.

The following definitions are taken from [23]. Let φ be a surjective morphism from Σ^* to a finite monoid M. For all k consider the subset $\varphi(\Sigma^k)$ of M. As M is finite there is a k such that $\varphi(\Sigma^{2k}) = \varphi(\Sigma^k)$. This implies that $\varphi(\Sigma^k)$ is a semigroup. The semigroup given by the smallest such k is called the *stable semigroup of* φ. If S is the stable semigroup of φ, S^1 is called *the stable monoid of* φ. If $\mathbf{V}$ is a variety of monoids, then we shall denote by $\mathbf{QV}$ the class of such surjective morphisms whose stable monoid is in $\mathbf{V}$ and by $\mathcal{L}(\mathbf{QV})$ the class of languages whose syntactic morphism is in $\mathbf{QV}$.

For $\mathbf{V}$ a variety of monoids, we say that a finite semigroup S is *locally* $\mathbf{V}$ if, for every

idempotent e of S, the monoid eSe belongs to $\mathbf{V}$; we denote by $\mathbf{LV}$ the class of locally-$\mathbf{V}$ finite semigroups, which happens to be a variety of semigroups. Finally, $\mathcal{L}(\mathbf{LV})$ denotes the class of languages whose syntactic semigroup is in $\mathbf{LV}$.

We now define languages recognized by $\mathbf{V} * \mathbf{Mod}$ and $\mathbf{V} * \mathbf{D}$. We do not use the standard algebraic definition using the wreath product as we won't need it, but directly a characterization of the languages recognized by such algebraic objects [7, 31].

Let $\mathbf{V}$ be a variety of monoids. We say that a language L over Σ is in $\mathcal{L}(\mathbf{V} * \mathbf{Mod})$ if there is a number $k \in \mathbb{N}_{>0}$ and a language L' over $\Sigma \times \{0, \cdots, k-1\}$ whose syntactic monoid is in $\mathbf{V}$, such that L is the set of words w that belong to L' after adding to each letter of w its position modulo k.

Similarly we say that a language L over Σ is in $\mathcal{L}(\mathbf{V} * \mathbf{D})$ if there is a number $k \in \mathbb{N}$ and a language L' over $\Sigma \times \Sigma^{\leq k}$ (where $\Sigma^{\leq k}$ denotes all words over Σ of length at most k) whose syntactic monoid is in $\mathbf{V}$, such that L is the set of words w that belong to L' after adding to each letter of w the word composed of the k (or less when near the beginning of w) letters preceding that letter. The variety of semigroups $\mathbf{V} * \mathbf{D}$ can then be defined as the one generated by the syntactic semigroups of the languages in $\mathcal{L}(\mathbf{V} * \mathbf{D})$ as defined above.

A variety $\mathbf{V}$ is said to be *local* if $\mathcal{L}(\mathbf{V} * \mathbf{D}) = \mathcal{L}(\mathbf{LV})$. This is not the usual definition of locality, defined using categories, but it is equivalent to it [31, Theorem 17.3]. One of the consequences of locality that we will use is that $\mathcal{L}(\mathbf{V} * \mathbf{Mod}) = \mathcal{L}(\mathbf{QV})$ [9, Corollary 18] (see also [8, 19]).

Programs over varieties of monoids. Programs over monoids form a non-uniform model of computation, first defined by Barrington and Thérien [6], extending Barrington's permutation branching program model [3]. Let M be a finite monoid and Σ a finite alphabet. A program P over M is a finite sequence of instructions of the form (i, f) where i is a positive integer and f a function from Σ to M. The *length* of P is the number of its instructions. A program has *range* n if all its instructions use a number less than n. A program P of range n defines a function from Σ^n, the words of length n, to M as follows. On input $w \in \Sigma^n$, each instruction (i, f) outputs the monoid element $f(w_i)$. A sequence of instructions then yields a sequence of elements of M and their product is the output $P(w)$ of the program.

A language L over Σ is p-*recognized* by a sequence of programs $(P_n)_{n\in\mathbb{N}}$ if for each n, P_n has range n and length polynomial in n and there exists a subset F_n of M such that $L \cap \Sigma^n$ is precisely the set of words w of length n such that $P_n(w) \in F_n$.

We denote by $\mathcal{P}(M)$ the class of languages p-recognized by a sequence of programs $(P_n)_{n\in\mathbb{N}}$ over M. If $\mathbf{V}$ is a variety of monoids we denote by $\mathcal{P}(\mathbf{V})$ the union of all $\mathcal{P}(M)$ for $M \in \mathbf{V}$.

The following is a simple fact about $\mathcal{P}(\mathbf{V})$. Let Σ and Γ be two finite alphabets and $\mu\colon \Sigma^* \to \Gamma^*$ be a morphism. We say that μ is length multiplying, or that μ *is an* lm-*morphism*, if there is a constant k such that for all $a \in \Sigma$, the length of $\mu(a)$ is k.

▶ **Lemma 1.** *[18, Corollary 3.5] Let* $\mathbf{V}$ *be a variety of monoids, then* $\mathcal{P}(\mathbf{V})$ *is closed under Boolean operations, quotients and inverse images of* lm-*morphisms.*

3 General results about regular languages and programs

Let $\mathbf{V}$ be a variety of monoids. By definition any regular language recognized by a monoid in $\mathbf{V}$ is p-recognized by a sequence of programs over a monoid in $\mathbf{V}$. Actually, since in a program over some monoid in $\mathbf{V}$, the monoid element output for each instruction can depend on the position of the letter read, hence in particular on its position modulo some

fixed number, it is easy to see that any regular language in $\mathcal{L}(\mathbf{V} * \mathbf{Mod})$ is p-recognized by a sequence of programs over some monoid in $\mathbf{V}$. In this section we show that for some "well behaved" varieties $\mathbf{V}$ the converse inclusion holds.

For this we introduce the notion of an *sp*-variety of monoids. This notion is inspired by the notion of p-varieties (program-varieties) of monoids, that seems to have been originally defined by Péladeau in his Ph.D. thesis [20] and later used by Tesson in his own Ph.D. thesis [28]. The notion of a p-variety has also been defined for semigroups by Péladeau, Straubing and Thérien in [21] in order to obtain results similar to ours for varieties of semigroups of the form $\mathbf{V} * \mathbf{D}$.

Let μ be a morphism from Σ^* to a finite monoid M. We denote by $\mathcal{W}(\mu)$ the set of languages L over Σ such that $L = \mu^{-1}(F)$ for some subset F of M. Given a semigroup S there is a unique morphism $\eta_S \colon S^* \to S^1$ extending the identity on S, called the *evaluation morphism of* S. We write $\mathcal{W}(S)$ for $\mathcal{W}(\eta_S)$. We define $\mathcal{W}(M)$ similarly for any monoid M. It is easy to see that if $M \in \mathbf{V}$ then $\mathcal{W}(M) \subseteq \mathcal{P}(\mathbf{V})$. The tameness condition requires a converse of this observation.

▶ **Definition 2.** An sp-*variety of monoids*, which we will call a *tame* variety, is a variety $\mathbf{V}$ of monoids such that for any finite semigroup S, if $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$ then $S^1 \in \mathbf{V}$.

The p-variety of monoids in [20, 28] is similar to our *sp*-variety but the former only requires that any finite monoid M verifying $\mathcal{W}(M) \subseteq \mathcal{P}(\mathbf{V})$ must in fact belong to $\mathbf{V}$. This implies that any *sp*-variety of monoids is also a p-variety of monoids, but the converse is not always true. For instance, $\mathbf{J}$ is a p-variety of monoids [28], but Proposition 5 below states that $\mathbf{J}$ is not an *sp*-variety.

An example of an *sp*-variety of monoids is the class of aperiodic monoids $\mathbf{A}$. This is a consequence of the result that for any number $k > 1$, checking if the number of a modulo k in a word is congruent to 0 is not in $\mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ [12, 1] (a language we shall denote by $\mathrm{MOD_k}$ over the alphabet $\{0, 1\}$). Towards a contradiction, assume there would exist a semigroup S such that S^1 is not aperiodic but still $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{A})$. Then there is an x in S such that $x^\omega \neq x^{\omega+1}$. Consider the morphism $\mu \colon \{a, b\}^* \to S^1$ sending a to $x^{\omega+1}$ and b to x^ω, and the language $L = \mu^{-1}(x^\omega)$. It is easy to see that L is the language of all words with a number of a congruent to 0 modulo k, where k is the smallest number such that $x^{\omega+k} = x^\omega$. As $x^\omega \neq x^{\omega+1}$, $k > 1$. From $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{A})$ it follows that $\eta_S^{-1}(x^\omega)$ is p-recognized by a sequence of programs $(P_n)_{n\in\mathbb{N}}$ over an aperiodic monoid. For each $n \in \mathbb{N}$, we can transform P_n into P'_n where each instruction (i, f) in P_n simply becomes the instruction (i, f') in P'_n with $f'(\sigma) = f(\mu(\sigma))$ for all $\sigma \in \{a, b\}$, so that $P'_n(w) = P_n(\mu(w_1)\mu(w_2)\cdots\mu(w_n))$ for all $w \in \{a, b\}^n$. This entails that the sequence of programs $(P'_n)_{n\in\mathbb{N}}$ p-recognizes L, hence L is in $\mathcal{P}(\mathbf{A})$, a contradiction.

The following is the desired consequence of tameness.

▶ **Proposition 3.** *Let* $\mathbf{V}$ *be an* sp-*variety of monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$.

A similar result was proven for varieties of semigroups of the form $\mathbf{V} * \mathbf{D}$: if $\mathbf{V} * \mathbf{D}$ is a p-variety then the regular languages in $\mathcal{P}(\mathbf{V} * \mathbf{D})$ are exactly those in $\mathcal{L}(\mathbf{V} * \mathbf{D} * \mathbf{Mod})$ [21] (programs over semigroups being defined in the obvious way). Our proof follows the same lines.

Proof. Let L be a regular language in $\mathcal{P}(M)$ for some $M \in \mathbf{V}$. Let M_L be the syntactic monoid of L and η_L its syntactic morphism. Let S be the stable semigroup of η_L, in particular $S = \eta_L(\Sigma^k)$ for some k. We wish to show that S^1 is in $\mathbf{V}$.

We show that $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$ and conclude from the fact that $\mathbf{V}$ is an *sp*-variety that $S^1 \in \mathbf{V}$ as desired. Let $\eta_S \colon S^* \to S^1$ be the evaluation morphism of S. Consider $m \in S$ and consider $L' = \eta_S^{-1}(m)$. We wish to show that $L' \in \mathcal{P}(\mathbf{V})$. This implies that $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$ by closure under union, Lemma 1.

Let $L'' = \eta_L^{-1}(m)$. Since m belongs to the syntactic monoid of L and η_L is the syntactic morphism of L, a classical algebraic argument [22, Chapter 2, proof of Lemma 2.6] shows that L'' is a Boolean combination of quotients of L or their complements. By Lemma 1, we conclude that $L'' \in \mathcal{P}(\mathbf{V})$.

By definition of S, for any element s of S there is a word u_s of length k such that $\eta_L(u_s) = s$. Notice that this is precisely where we need to work with S and not S^1.

Let $f \colon S^* \to \Sigma^*$ be the *lm*-morphism sending s to u_s and notice that $L' = f^{-1}(L'')$. The result follows by closure of $\mathcal{P}(\mathbf{V})$ under inverse images of *lm*-morphisms, Lemma 1. ◀

We don't know whether it is always true that for *sp*-varieties of monoids $\mathbf{V}$, $\mathcal{L}(\mathbf{QV})$ is included into $\mathcal{P}(\mathbf{V})$. We can only prove it for local varieties.

▶ **Proposition 4.** *Let* $\mathbf{V}$ *be a local* sp*-variety of monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QV})$.

Proof. This follows from the fact that for local varieties $\mathcal{L}(\mathbf{QV}) = \mathcal{L}(\mathbf{V} * \mathbf{Mod})$ [9]. The result follows from Proposition 3, as we always have $\mathcal{L}(\mathbf{V} * \mathbf{Mod}) \subseteq \mathcal{P}(\mathbf{V})$. ◀

As $\mathbf{A}$ is local [31, Example 15.5] and an *sp*-variety, it follows from Proposition 4 that the regular languages in $\mathcal{P}(\mathbf{A})$, hence in AC^0, are precisely those in $\mathcal{L}(\mathbf{QA})$, which is the characterization of the regular languages in AC^0 obtained by Barrington, Compton, Straubing and Thérien [4].

We will see in the next section that $\mathbf{DA}$ is an *sp*-variety. As it is also local [2], we get from Proposition 4 that the regular languages of $\mathcal{P}(\mathbf{DA})$ are precisely those in $\mathcal{L}(\mathbf{QDA})$.

As explained in the introduction, the language $(a+b)^*ac^+$ can be *p*-recognized by a program over $\mathbf{J}$. A simple algebraic argument shows that it is not in $\mathcal{L}(\mathbf{QJ})$. Hence, by Proposition 3, we have the following result:

▶ **Proposition 5.** $\mathbf{J}$ *is not an* sp*-variety of monoids.*

4 The case of DA

In this section, we prove that $\mathbf{DA}$ is an *sp*-variety of monoids. Combined with the fact that $\mathbf{DA}$ is local [2], we obtain the following result by Proposition 4.

▶ **Theorem 6.** $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA})$.

The result follows from the following main technical contribution:

▶ **Proposition 7.** $(c+ab)^*$, $(b+ab)^*$ *and* $((b^*ab^*)^k)^*$ *for any integer* $k \geq 2$ *are regular languages not in* $\mathcal{P}(\mathbf{DA})$.

Before proving the proposition we first show that it implies that $\mathbf{DA}$ is an *sp*-variety of monoids. Assume S is a finite semigroup such that $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{DA})$. Let $\eta_S \colon S^* \to S^1$ be the evaluation morphism of S. We need to show that S^1 is in $\mathbf{DA}$.

Assume first that S^1 is aperiodic. Towards a contradiction, assume S^1 is not in $\mathbf{DA}$. Then, there exist x, y in S such that $(xy)^\omega \neq (xy)^\omega x (xy)^\omega$.

Set $e = (xy)^\omega$, $f = (yx)^\omega$, $s = ex$ and $t = ye$. Our hypothesis says that $exe \neq e$. We now have two cases, depending on whether $fyf = f$ or not.

So, suppose $fyf \neq f$. In that case, let $\mu\colon \{a,b,c\}^* \to S^1$ be the morphism sending a to s, b to t and c to e and consider the language $L = \mu^{-1}(\{1,e\})$. Assume that L contains a word w with two consecutive a. Then $w = w_1aaw_2$ and as $w \in L$, either $e = \mu(w_1)exex\mu(w_2)$ or $1 = \mu(w_1)exex\mu(w_2)$. In both cases $e = u_1exeu_2$ for some suitable values of u_1 and u_2. This implies that $e = u_1^\omega e(xeu_2)^\omega = (u_1xe)^\omega eu_2^\omega$. Because S^1 is aperiodic, this implies: $e = exeu_2 = eu_2$. Hence $exe = e$, contradicting the fact that $exe \neq e$. Similar arguments show that L cannot contain a word with two consecutive b, a factor ac or a factor cb.

Any word in L is of the form $u_0v_1u_1\cdots u_{k-1}v_ku_k$ where $k \in \mathbb{N}$, $v_1,\dots,v_k \in c^+$ and $u_0,\dots,u_k \in (a+b)^*$. As w does not contain aa nor bb as a factor, we have that $u_0,\dots,u_k \in (b+\varepsilon)(ab)^*(a+\varepsilon)$. When $k \geq 1$, as moreover w does not contain ac nor cb as a factor, it follows that $u_1,\dots,u_{k-1} \in (ab)^*$, $u_0 \in (b+\varepsilon)(ab)^*$ and $u_k \in (ab)^*(a+\varepsilon)$; now since $\mu(ab) = exye = e$ by aperiodicity and $\mu(b)e = yee = ye \notin \{1,e\}$ (otherwise $fyf = f$), $e\mu(a) = eex = ex \notin \{1,e\}$ (otherwise $exe = e$), $\mu(b)e\mu(a) = yeeex = yex = f \notin \{1,e\}$ (by aperiodicity and as otherwise $exe = e$), w cannot start with b or end with a, hence $u_0, u_k \in (ab)^*$. And similarly, $u_0 \in (ab)^*$ when $k = 0$. Therefore, $L = (c+ab)^*$.

The other case, when $fyf = f$, is treated similarly using the morphism $\mu\colon \{a,b\}^* \to S^1$ sending a to s and b to t and considering the language $L = \mu^{-1}(\{1,e,t\})$. Using arguments as for the previous case, one can conclude that $L = (b+ab)^*$.

Assume now that S^1 is not aperiodic. As in the two previous cases, we can then prove that there exist a morphism $\mu\colon \{a,b\}^* \to S^1$ and a subset $F \subseteq S^1$ such that $L = \mu^{-1}(F)$ is the regular language $((b^*ab^*)^k)^*$ for some $k \in \mathbb{N}, k \geq 2$ of all words over $\{a,b\}$ with a number of a congruent to 0 modulo k.

In all cases, we have a language L defined as $\mu^{-1}(Q)$ for some subset Q of S^1 and some morphism μ to S^1 sending letters to elements of S. As $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{DA})$, it follows that $\eta_S^{-1}(Q)$ is p-recognized by a sequence of programs $(P_n)_{n\in\mathbb{N}}$ over a monoid in $\mathbf{DA}$. As in the example prior to Proposition 3 in the previous section, for each $n \in \mathbb{N}$, we can transform P_n into P'_n over the same monoid so that the sequence of programs $(P'_n)_{n\in\mathbb{N}}$ p-recognizes L. In all cases, we get a contradiction with Proposition 7.

In the remaining part of this section we prove Proposition 7.

Proof of Proposition 7. The idea of the proof is the following. We work by contradiction and assume that we have a sequence of programs over some monoid M of $\mathbf{DA}$ deciding one of the targeted language. Let n be much larger than the size of M, and let P_n be the program running on words of length n. Consider a language of the form Δ^* for some finite set Δ of words (for instance assume $\Delta = \{c, ab\}$, $\Delta = \{b, ab\}$, ...). We will show that we can fix a constant (depending on M and Δ but not on n) number of entries to P_n such that P_n always accepts or always rejects and there is a completion of the fixed entries in Δ^*; hence, if Δ was chosen so that there is actually a completion of the fixed entries in the targeted language and one outside of it, P_n cannot recognize it. We cannot prove this for all Δ, in particular it will not work for $\Delta = \{ab\}$ and indeed $(ab)^*$ is in $\mathcal{P}(\mathbf{DA})$. The key property of our Δ is that after fixing any letter at any position, except maybe for a constant number of positions, one can still complete the word into one within Δ^*. This is not true for $\Delta = \{ab\}$ because after fixing a b in an odd position all completions fall outside of $(ab)^*$.

We now add some technical details.

Let Δ be a finite set of words. Let Σ be the corresponding finite alphabet and let $\perp$ be a letter not in Σ. A *mask* is a word over $\Sigma \cup \{\perp\}$. The positions of a mask carrying a $\perp$ are called *free* while the positions carrying a letter in Σ are called *fixed*. A mask λ' is a *submask* of a mask λ if it is formed from λ by replacing some occurrences of $\perp$ by a letter in Σ.

A *completion* of a mask λ is a word w over Σ that is built from λ by replacing all occurrences of $\perp$ by a letter in Σ. Notice that all completions of a mask have the same length as the mask itself. A mask λ is Δ-*compatible* if it has a completion in Δ^*.

The *dangerous* positions of a mask λ are the positions within distance $2l-2$ of the fixed positions or within distance $l-1$ of the beginning or the end of the mask, where l is the maximal length of a word in Δ. A position that is not dangerous is said to be *safe*.

We say that Δ is *safe* if the following holds. Let λ be a Δ-compatible mask. Let i be any free position of λ that is not dangerous. Let a be any letter in Σ. Then the submask of λ constructed by fixing a at position i is Δ-compatible. We have already seen that $\Delta = \{ab\}$ is not safe. However our targeted Δ, $\Delta = \{c, ab\}$, $\Delta = \{b, ab\}$, $\Delta = \{a, b\}$, are safe. We always consider Δ to be safe in the following.

Finally, we say that a completion w of a mask λ is *safe* if w is a completion of λ belonging to Δ^* or is constructed from a completion of λ in Δ^* by modifying only letters at safe positions of λ, the dangerous positions remaining unchanged.

The following lemma is the key to the proof. It shows that modulo fixing a few entries, one can fix the output.

▶ **Lemma 8.** *Let M be a monoid in* **DA**. *Let λ be a Δ-compatible mask of length n, let P be a program over M of range n, let u and v be elements of M. Then there is an element t of M and a Δ-compatible submask λ' of λ obtained by fixing a number of free positions independent from n such that any safe completion w of λ' verifies $uP(w)v = t$.*

The proof of Lemma 8 is technical. As often in this setting, it relies on Green's relations for decomposing monoids. Then, at each stage of the decomposition, a small number of safe positions are fixed accordingly. Details can be found in Appendix A.1.

Setting $\Delta = \{c, ab\}$ or $\Delta = \{b, ab\}$, when applying Lemma 8 for some monoid $M \in \mathbf{DA}$ with the trivial Δ-compatible mask λ of length n containing only free positions, with P some program over M of range n and with u and v the identity of M, the resulting mask λ' has the property that we have an element t of M such that $P(w) = t$ for any safe completion w of λ'. Since the mask λ' is Δ-compatible and as long as n is big enough, we have a safe completion $w_0 \in \Delta^*$ and a safe completion $w_1 \notin \Delta^*$. Hence P cannot recognize Δ^*. This implies that $(c+ab)^* \notin \mathcal{P}(M)$ and $(b+ab)^* \notin \mathcal{P}(M)$. Finally, for any $k \in \mathbb{N}, k \geq 2$, we can prove that $((b^*ab^*)^k)^* \notin \mathcal{P}(M)$ by setting $\Delta = \{a, b\}$ and completing the mask given by the lemma by setting the letters in such a way that we have the right number of a modulo k in one case and not in the other case.

This concludes the proof of Proposition 7 because the argument above holds for any monoid in **DA**.

5 A fine hierarchy in $\mathcal{P}(\mathbf{DA})$

The definition of p-recognition by a sequence of programs over a monoid given in Section 2 requires that for each n, the program reading the entries of length n has a length polynomial in n. In the case of $\mathcal{P}(\mathbf{DA})$, the polynomial length restriction is without loss of generality: any program over a monoid in **DA** is equivalent to one of polynomial length over the same monoid [29] (in the sense that they recognize the same languages). In this section, we show that this does not collapse further: in the case of **DA**, programs of length $O(n^{k+1})$ express strictly more than those of length $O(n^k)$.

Following [13], we use an alternative definition of the languages recognized by a monoid in **DA**. We define by induction a hierarchy of classes of languages $\mathcal{SUM}_k$, where $\mathcal{SUM}$

stands for *strongly unambiguous monomial*. A language L is in $\mathcal{SUM}_0$ if it is of the form A^* for some finite alphabet A. A language L is in $\mathcal{SUM}_k$ if it is in $\mathcal{SUM}_{k-1}$ or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i$ and $L_2 \in \mathcal{SUM}_j$ and some letter a with $i + j = k - 1$ such that no word of L_1 contains the letter a or no word of L_2 contains the letter a.

Gavaldà and Thérien showed that a language L is recognized by a monoid in **DA** iff there is a k such that L is a Boolean combination of languages in $\mathcal{SUM}_k$ [13]. For each $k \in \mathbb{N}$, we denote by $\mathbf{DA_k}$ the variety of monoids generated by the syntactic monoids of the Boolean combinations of languages in $\mathcal{SUM}_k$. It can be checked that, for each k, $\mathbf{DA_k}$ forms a variety of monoids recognizing precisely Boolean combinations of languages in $\mathcal{SUM}_k$ (see Appendix A.2).

5.1 Strict hierarchy

For each k we exhibit a language $L_k \subseteq \{0,1\}^*$ that can be recognized by a sequence of programs of length $\mathrm{O}(n^k)$ over a monoid M_k in **DA** but cannot be recognized by any sequence of programs of length $\mathrm{O}(n^{k-1})$ over any monoid in **DA**.

The language L_k expresses a property of the first k occurrences of 1 in the input word. To define L_k we say that S is a *k-set over n* if S is a set where each element is an ordered tuple of k distinct elements of $[n]$. For any sequence $\Delta = (S_n)_{n\in\mathbb{N}}$ of k-sets over n, we set $L_\Delta = \bigcup_{n\in\mathbb{N}} K_{n,S_n}$, where K_{n,S_n} is the set of words over $\{0,1\}$ of length n such that for each of them, it contains at least k occurrences of 1 and the ordered k-tuple of the positions of the first k occurrences of 1 belongs to S_n.

On the one hand, we show that for all k there is a monoid M_k in **DA** such that for all Δ the language L_Δ is recognized by a sequence of programs over M_k of length $\mathrm{O}(n^k)$. The proof is done by an inductive argument on k.

On the other hand, we show that there is a Δ such that for any finite monoid M and any sequence of programs $(P_n)_{n\in\mathbb{N}}$ over M of length $\mathrm{O}(n^{k-1})$, L_Δ is not recognized by $(P_n)_{n\in\mathbb{N}}$. This is done using a counting argument: for some monoid size i, for n big enough, the number of languages in $\{0,1\}^n$ recognized by a program over some monoid of size i of length at most $\alpha \cdot n^{k-1}$ is upper-bounded by a number that turns out to be asymptotically smaller than the number of different possible K_{n,S_n}.

Upper bound. We start with the upper bound. Notice that for $\Delta = (S_n)_{n\in\mathbb{N}}$, the language of words of length n of L_Δ is exactly K_{n,S_n}. Hence the fact that L_Δ can be recognized by a sequence of programs over a monoid in **DA** of length $\mathrm{O}(n^k)$ is a consequence of the following proposition.

▶ **Proposition 9.** *For all $k \in \mathbb{N}_{>0}$ there is a monoid $M_k \in \mathbf{DA_k}$ such that for all $n \in \mathbb{N}$ and all S_n k-sets over n, the language K_{n,S_n} is recognized by a program over M_k of length at most $4n^k$.*

Proof. We first define by induction on k a family of languages Z_k over the alphabet $Y_k = \{\bot_l, \top_l \mid 1 \le l \le k\}$. For $k = 0$, Z_0 is $\{\varepsilon\}$. For $k > 0$, Z_k is the set of words containing $\top_k$ and such that the first occurrence of $\top_k$ has no $\bot_k$ to its left, and the sequence between the first occurrence of $\top_k$ and the first occurrence of $\bot_k$ or $\top_k$ to its right, or the end of the word if there is no such letter, belongs to Z_{k-1}. A simple induction on k shows that Z_k is defined by the following expression

$$Y_{k-1}^* \top_k Y_{k-2}^* \top_{k-1} \cdots Y_1^* \top_2 \top_1 Y_k^*$$

and therefore it is in $\mathcal{SUM}_k$ and its syntactic monoid M_k is in $\mathbf{DA_k}$.

Fix n. If $n = 0$, the proposition follows trivially, otherwise, we define by induction on k a program $P_k(i, S)$ for every k-set S over n and every $1 \leq i \leq n+1$ that will for the moment output letters of Y_k instead of outputting elements of M_k.

For any $k > 0$ and S a k-set over n, let $f_{j,S}$ be the function with $f_{j,S}(0) = \varepsilon$ and $f_{j,S}(1) = \top_k$ if j is the first element of some ordered k-tuple of S, $f_{j,S}(1) = \bot_k$ otherwise. We also let g_k be the function with $g_k(0) = \varepsilon$ and $g_k(1) = \bot_k$. If S is a k-set over n and $j \leq n$ then $S|j$ denotes the $(k-1)$-set over n containing the ordered $(k-1)$-tuples $\bar{t}$ such that $(j, \bar{t}) \in S$.

For $k > 0$, $1 \leq i \leq n+1$ and S a k-set over n, the program $P_k(i, S)$ is the following sequence of instructions:

$$(i, f_{i,S})P_{k-1}(i+1, S|i)(i, g_k) \cdots (n, f_{n,S})P_{k-1}(n+1, S|n)(n, g_k).$$

In other words, the program guesses the first occurrence $j \geq i$ of 1, returns $\bot_k$ or $\top_k$ depending on whether it is the first element of an ordered k-tuple in S, and then proceeds for the next occurrences of 1 by induction.

For $k = 0$, $1 \leq i \leq n+1$ and S a 0-set over n (that is empty or contains ε, the only ordered 0-tuple of elements of $[n]$), the program $P_0(i, S)$ is the empty program ε.

A simple computation shows that for any $k \in \mathbb{N}_{>0}$, $1 \leq i \leq n+1$ and S a k-set over n, the number of instructions in $P_k(i, S)$ is at most $4n^k$.

A simple induction on k shows that when running on a word $w \in \{0,1\}^n$, for any $k \in \mathbb{N}_{>0}$, $1 \leq i \leq n+1$ and S a k-set over n, $P_k(i, S)$ returns a word in Z_k iff the ordered k-tuple of the positions of the first k occurrences of 1 starting at position i in w exists and is an element of S.

For any $k > 0$ and S_n a k-set over n, it remains to apply the syntactic morphism of Z_k to the output of the functions in the instructions of $P_k(1, S_n)$ to get a program over M_k of length at most $4n^k$ recognizing K_{n,S_n}. ◀

Lower bound. The following claim is a simple counting argument.

▶ **Claim 10.** *For all $i \in \mathbb{N}_{>0}$ and $n \in \mathbb{N}$, the number of languages in $\{0,1\}^n$ recognized by programs over a monoid of size i, reading inputs of length n over the alphabet $\{0,1\}$, with at most $l \in \mathbb{N}$ instructions, is bounded by $i^{i^2} 2^i \cdot (n \cdot i^2)^l$.*

Proof. Fix a monoid M of size i. Since a program over M of range n with less than l instructions can always be completed into such a program with exactly l instructions recognizing the same languages in $\{0,1\}^n$ (using the identity of M), we only consider programs with exactly l instructions. As $\Sigma = \{0,1\}$, there are $n \cdot i^2$ choices for each of the l instructions of a range n program over M reading inputs in $\{0,1\}^*$. Such a program can recognize at most 2^i different languages in $\{0,1\}^n$. Hence, the number of languages in $\{0,1\}^n$ recognized by programs over M of length at most l is at most $2^i \cdot (n \cdot i^2)^l$. The result follows from the facts that there are at most i^{i^2} isomorphism classes of monoids of size i and that two isomorphic monoids allow to recognize the same languages in $\{0,1\}^n$. ◀

If for some $k \in \mathbb{N}_{>0}$ and $1 \leq i \leq \alpha$, $\alpha \in \mathbb{N}_{>0}$, we apply Claim 10 for all $n \in \mathbb{N}$, $l = \alpha \cdot n^{k-1}$, we get a number of languages upper-bounded by $n^{\mathrm{O}(n^{k-1})}$, which is asymptotically strictly smaller than the number of distinct K_{n,S_n}, which is $2^{\binom{n}{k}}$.

Hence, for all $j \in \mathbb{N}_{>0}$, there exist an $n_j \in \mathbb{N}$ and T_j a k-set over n_j such that no program over a monoid of size $1 \leq i \leq j$ and of length at most $j \cdot n^{k-1}$ recognizes K_{n_j,T_j}. Moreover, we can assume without loss of generality that the sequence $(n_j)_{j \in \mathbb{N}_{>0}}$ is increasing. Let

$\Delta = (S_n)_{n\in\mathbb{N}}$ be such that $S_{n_j} = T_j$ for all $j \in \mathbb{N}_{>0}$ and $S_n = \emptyset$ for any $n \in \mathbb{N}$ verifying that it is not equal to any n_j for $j \in \mathbb{N}_{>0}$. We show that no sequence of programs over a finite monoid of length $\mathrm{O}(n^{k-1})$ can recognize L_Δ. If this were the case, then let i be the size of the monoid. Let $j \geq i$ be such that for any $n \in \mathbb{N}$, the n-th program has length at most $j \cdot n^{k-1}$. But, by construction, we know that there does not exist any such program of range n_j recognizing K_{n_j,T_j}, a contradiction.

This implies the following hierarchy (where $\mathcal{P}(\mathbf{V}, s(n))$ for some variety of monoids $\mathbf{V}$ and a function $s\colon \mathbb{N} \to \mathbb{N}$ denotes the class of languages recognizable by a sequence of programs of length $\mathrm{O}(s(n))$):

▶ **Proposition 11.** *For all $k \in \mathbb{N}$, $\mathcal{P}(\mathbf{DA}, n^k) \subsetneq \mathcal{P}(\mathbf{DA}, n^{k+1})$.*

5.2 Collapse

Tesson and Thérien showed that any program over a monoid M in **DA** is equivalent to one of polynomial length [29]. We now show that if we further assume that M is in $\mathbf{DA_k}$ then the length can be assumed to be $\mathrm{O}(n^k)$.

▶ **Proposition 12.** *Let $k > 0$. Let $M \in \mathbf{DA_k}$. Then any program over M is equivalent to a program over M of length $\mathrm{O}(n^k)$.*

The equivalent program of length $\mathrm{O}(n^k)$ is actually a subprogram of the initial one. For each possible acceptance set, an input word to the program is accepted iff the word over the alphabet M produced by the program belongs to some fixed Boolean combination of languages in $\mathcal{SUM}_k$. The idea is then just to keep enough instructions so that membership of the produced word over M in each of these languages does not change. For each of those languages, the set of instructions to keep is defined by induction on k using the inductive definition of $\mathcal{SUM}_k$ given at the beginning of this section. Roughly, at each step, for each input letter and each input position, the small program keeps the first or last instruction of the big program producing the "pivot element" when reading this input letter at that position. The number of instructions kept in the end is then in $\mathrm{O}(n^k)$. The details can be found in Appendix A.3.

6 Conclusion

For local and tame varieties $\mathbf{V}$ we have shown that the regular languages recognized by programs over $\mathbf{V}$ are exactly those in $\mathcal{L}(\mathbf{QV})$. It is not clear whether locality is necessary. We don't have any example of a tame variety $\mathbf{V}$ for which $\mathcal{L}(\mathbf{QV})$ is not included into $\mathcal{P}(\mathbf{V})$. We leave this question for future work.

We have shown that **DA** is tame but that **J** is not. Another example of a tame variety is **A**. However we needed the fact that $\mathrm{MOD_m}$ is not in $\mathsf{AC^0}$ for all $m \geq 2$ in order to prove tameness. It would be interesting to give a direct algebraic proof of this result. As this would in particular imply that $\mathrm{MOD_2}$ is not in $\mathsf{AC^0}$ by Proposition 4, it is certainly a challenging task.

Finding the regular languages recognized by programs over **J** is left for future work.

To conclude we should add, in fairness, that the progress reported here does not obviously bring us closer to major $\mathsf{NC^1}$ complexity subclasses separations, but it does uncover new ways in which a program can or cannot circumvent the limitations imposed by the underlying monoid algebraic structure available to it.

References

1 Miklós Ajtai. Σ^1_1-formulae on finite structures. In *Ann. Pure and Appl. Logic*, volume 24, pages 1–48, 1983.

2 Jorge Almeida. A syntactical proof of locality of DA. *Int. J. of Algebra and Computation (IJAC)*, 6(2):165–178, 1996.

3 David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.

4 David A. Mix Barrington, Kevin J. Compton, Howard Straubing, and Denis Thérien. Regular languages in NC^1. *J. Comput. Syst. Sci.*, 44(3):478–499, 1992.

5 David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Inf. Comput.*, 89(2):109–132, 1990.

6 David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988.

7 Laura Chaubard, Jean-Éric Pin, and Howard Straubing. Actions, wreath products of C-varieties and concatenation product. *Theoretical Computer Science*, 356(1-2):73–89, 2006.

8 Luc Dartois. *Méthodes algébriques pour la théorie des automates*. PhD thesis, Université Paris Diderot, Paris, 2014.

9 Luc Dartois and Charles Paperman. Adding modular predicates. *CoRR*, abs/1401.6576, 2014. URL: `http://arxiv.org/abs/1401.6576`.

10 Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, New York, 1974.

11 Samuel Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York, 1976.

12 Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

13 Ricard Gavaldà and Denis Thérien. Algebraic characterizations of small classes of Boolean functions. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2607 of *Lecture Notes in Computer Science*, pages 331–342. Springer, 2003.

14 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 6–20. ACM, 1986.

15 Kenneth Krohn and John L. Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. In *Trans. Amer. Math. Soc.*, volume 116, pages 450–464, 1965.

16 Alexis Maciel, Pierre Péladeau, and Denis Thérien. Programs over semigroups of dot-depth one. *Theor. Comput. Sci.*, 245(1):135–148, 2000.

17 Ward D. Maurer and John L. Rhodes. A property of finite simple non-abelian groups. In *Amer. Math. Soc.*, volume 16, pages 552–554, 1965.

18 Pierre McKenzie, Pierre Péladeau, and Denis Thérien. NC^1: The automata-theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

19 Charles Paperman. *Circuits booléens, prédicats modulaires et langages réguliers*. PhD thesis, Université Paris Diderot, Paris, 2014.

20 Pierre Péladeau. *Classes de circuits booléens et variétés de monoïdes*. PhD thesis, Université Pierre-et-Marie-Curie (Paris-VI), Paris, France, 1990.

21 Pierre Péladeau, Howard Straubing, and Denis Thérien. Finite semigroup varieties defined by programs. *Theor. Comput. Sci.*, 180(1-2):325–339, 1997.

22 Jean-Éric Pin. *Varieties of formal languages*. North Oxford, London and Plenum, New-York, 1986. (Traduction de Variétés de langages formels).

23 Jean-Éric Pin and Howard Straubing. Some results on $\mathcal{C}$-varieties. *RAIRO-Theor. Inf. Appl.*, 39(1):239–262, 2005.

24 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes*, 41(4):333–338, 1987.

25 Jan Reiterman. The Birkhoff theorem for finite algebras. *algebra universalis*, 14(1):1–10, 1982.

26 Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages, 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 214–222. Springer, 1975.

27 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82. ACM, 1987.

28 Pascal Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, McGill University, Montreal, Canada, 2003.

29 Pascal Tesson and Denis Thérien. The computing power of programs over finite monoids. *J. Autom. Lang. Comb.*, 7(2):247–258, November 2001.

30 Pascal Tesson and Denis Thérien. Diamonds are forever: the variety DA. *Semigroups, algorithms, automata and languages*, 1:475–500, 2002.

31 Bret Tilson. Categories as algebra: An essential ingredient in the theory of monoids. *J. of Pure and Applied Algebra*, 48(1-2), 1987.

A Appendix

A.1 Missing proofs from Section 4

Let M be a monoid in **DA** whose identity we will denote by 1.

Given two elements u, u' of M we say that $u \leq_J u'$ if there are elements v, v' of M such that $u' = vuv'$. We write $u \sim_J u'$ if $u \leq_J u'$ and $u' \leq_J u$. We write $u <_J u'$ if $u \leq_J u'$ and $u' \not\sim_J u$. Given two elements u, u' of M we say that $u \leq_R u'$ if there is an element v of M such that $u' = uv$. We write $u \sim_R u'$ if $u \leq_R u'$ and $u' \leq_R u$. We write $u <_R u'$ if $u \leq_R u'$ and $u' \not\sim_R u$. Given two elements u, u' of M we say that $u \leq_L u'$ if there is an element v of M such that $u' = vu$. We write $u \sim_L u'$ if $u \leq_L u'$ and $u' \leq_L u$. We write $u <_L u'$ if $u \leq_L u'$ and $u' \not\sim_L u$.

We shall use the following well-known fact about these preorders and equivalence relations (see [22, Chapter 3, Proposition 1.4]).

▶ **Lemma 13.** *For all elements u and v of M, if $u \leq_R v$ and $u \sim_J v$, then $u \sim_R v$. Similarly, if $u \leq_L v$ and $u \sim_J v$, then $u \sim_L v$.*

An element r of M is R-bad for u if $u <_R ur$. Similarly an element r of M is L-bad for v if $u <_L rv$. It follows from $M \in$ **DA** that being R-bad or L-bad only depends on the $\sim_R$ or $\sim_L$ class, respectively:

▶ **Lemma 14.** *[Folklore] If M is in* **DA**, *then $u \sim_R u'$ and $ur \sim_R u$ implies $u'r \sim_R u$. Similarly $u \sim_L u'$ and $ru \sim_L u$ implies $ru' \sim_L u$.*

Let Δ be a finite set of words and Σ be the corresponding finite alphabet, Δ being safe, and let $n \in \mathbb{N}$. We are now going to prove the main technical lemma that allows us to assert that after fixing a constant number of positions in the input of a program over M, it can still be completed into a word of Δ^*, but the program cannot make the difference between any two possible completions anymore. To prove the lemma, we define a relation $\prec$ on the set of quadruplets (λ, P, u, v) where λ is a mask of length n, P is a program over M for words of length n and u and v are two elements. We will say that an element $(\lambda_1, P_1, u_1, v_1)$ is strictly smaller than $(\lambda_2, P_2, u_2, v_2)$, written $(\lambda_1, P_1, u_1, v_1) \prec (\lambda_2, P_2, u_2, v_2)$ if λ_1 is a submask of λ_2, P_1 is a subprogram of P_2 and one of the following cases occurs:

1. $u_2 <_R u_1$ and $v_1 = v_2$ and P_1 is a suffix of P_2 and $u_1 P_1(w) v_1 = u_2 P_2(w) v_2$ for all safe completions w of λ_1;
2. $v_2 <_L v_1$ and $u_1 = u_2$ and P_1 is a prefix of P_2 and $u_1 P_1(w) v_1 = u_2 P_2(w) v_2$ for all safe completions w of λ_1;
3. $u_2 = u_1$ and $v_1 = 1$ and P_1 is a prefix of P_2 and $u_1 P_1(w) v_1 <_J u_2 P_2(w) v_2$ for all safe completions w of λ_1;
4. $v_2 = v_1$ and $u_1 = 1$ and P_1 is a suffix of P_2 and $u_1 P_1(w) v_1 <_J u_2 P_2(w) v_2$ for all safe completions w of λ_1.

Note that, since M is finite, this relation is well-founded (that is, it has no infinite decreasing chain, an infinite sequence of quadruplets $\mu_0, \mu_1, \mu_2, \ldots$ such that $\mu_{i+1} \prec \mu_i$ for all $i \in \mathbb{N}$) and the maximal length of any decreasing chain depends only on M.

▶ **Lemma 8.** Let λ be a Δ-compatible mask of length n, let P be a program over M of range n, let u and v be elements of M. Then there is an element t of M and a Δ-compatible submask λ' of λ obtained by fixing a number of free positions independent from n such that any safe completion w of λ' verifies $uP(w)v = t$.

Proof. The proof goes by induction on $\prec$.

Let λ be a Δ-compatible mask of length n, let P be a program over M for words of length n, let u and v be elements of M such that (λ, P, u, v) is of height h, and assume that for any quadruplet (λ', P', u', v') strictly smaller than (λ, P, u, v), the lemma is verified. Consider the following conditions concerning the quadruplet (λ, P, u, v):

(a) there does not exist any instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is R-bad for u;
(b) v is not R-bad for u;
(c) there does not exist any instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is L-bad for v;
(d) u is not L-bad for v.

We will now do a case analysis based on which of these conditions are violated or not.

Case 1: condition (a) is violated. So there exists some instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is R-bad for u. Let i be the smallest number of such an instruction.
Let P' be the subprogram of P until instruction $i-1$. Let w be a safe completion of λ. By minimality of i and by Lemma 14, it follows that $u \sim_R uP'(w)$.
So, because $f(a)$ is R-bad for u, any safe completion w of λ', which is also a safe completion of λ, is such that $u \sim_R uP'(w) <_R uP'(w)f(a) \leq_R uP(w)v$ by Lemma 14, hence $uP'(w) <_J uP(w)v$ by Lemma 13. So $(\lambda', P', u, 1) \prec (\lambda, P, u, v)$, therefore, by induction we get a Δ-compatible submask λ_1 of λ' and a monoid element t_1 such that $uP'(w) = t_1$ for all safe completions w of λ_1.
Let P'' be the subprogram of P starting from instruction $i+1$. Notice that, since $u \sim_R t_1$ (by what we have proven just above), $u <_R t_1 f(a)$ (by Lemma 14) and $t_1 f(a)P''(w)v = uP'(w)f(a)P''(w)v = uP(w)v$ for all safe completions w of λ_1. Hence, $(\lambda_1, P'', t_1 f(a), v)$ is strictly smaller than (λ_1, P, u, v) and by induction we get a Δ-compatible submask λ_2 of λ_1 and a monoid element t such that $t_1 f(a)P''(w)v = t$ for all safe completions w of λ_2.
Hence any safe completion w of λ_2 is such that

$$uP(w)v = uP'(w)f(a)P''(w)v = t_1 f(a)P''(w)v = t \ .$$

Hence λ_2 and t are the desired solutions.

Case 2: condition (a) is verified but condition (b) is violated, so v is R-bad for u and Case 1 does not apply.
Let w be a safe completion of λ: for any instruction (x, f) of P, as the submask λ' of λ formed by setting position x to w_x (if it wasn't already the case) is Δ-compatible (by the fact that Δ is safe and w is a safe completion of λ), $f(w_x)$ cannot be R-bad for u, otherwise condition (a) would be violated, so $u \sim_R uf(w_x)$. Hence, by Lemma 14, $u \sim_R uP(w)$ for all safe completions w of λ. Notice then that $u \sim_R uP(w) <_R uP(w)v$ (by Lemma 14), hence $uP(w) <_J uP(w)v$ (by Lemma 13) for all safe completions w of λ. So $(\lambda, P, u, 1) \prec (\lambda, P, u, v)$, therefore we obtain a monoid element t_1 and a Δ-compatible submask λ' of λ by induction such that $uP(w) = t_1$ for all completions w of λ'. $t = t_1 v$ is the desired element of M.

Case 3: condition (c) is violated. So there exists some instruction (x, f) of P such that for some letter a the submask λ' of λ formed by setting position x to a (if it wasn't already the case) is Δ-compatible and $f(a)$ is L-bad for v.
We proceed as for Case 1 by symmetry.

Case 4: condition (c) is verified but condition (d) is violated, so u is L-bad for v and Case 3 does not apply.
We proceed as for Case 2 by symmetry.

Case 5: conditions (a), (b), (c) and (d) are verified.
As it Case 2 and Case 4 we get that $u \sim_R uP'(w)$ and $v \sim_L P''(w)v$ for any prefix P' of P, any suffix P'' of P and all safe completions w of λ. Moreover, since condition (b) and condition (d) are verified, by Lemma 14, we get that $uP(w)v \sim_R u$ and $uP(w)v \sim_L v$ for all safe completions w of λ.
Let w_0 be a completion of λ that is in Δ^*. Let λ' be the submask of λ fixing all dangerous positions of λ using w_0. Then, for any completion w of λ', which is a safe completion of λ by construction, we have that $uP(w)v \sim_R u$ and $uP(w)v \sim_L v$. As M is aperiodic, this implies that there is a t in M such that $uP(w)v = t$ for all completions w of λ' (see [22, Chapter 3, Proposition 4.2]).

This concludes the proof of the lemma. ◀

A.2 $\mathcal{SUL}_k$ is a variety of languages

A *variety of languages* is a class of languages over arbitrary finite alphabets closed under Boolean operations, quotients and inverses of morphisms (i.e. if L is a language in the class over a finite alphabet Σ, if Γ is some other finite alphabet and $\varphi\colon \Gamma^* \to \Sigma^*$ is a morphism of monoids, then $\varphi^{-1}(L)$ is also in the class).

Eilenberg showed [11, Chapter VII, Section 3] that there is a bijective correspondence between varieties of monoids and varieties of languages: to each variety of monoids $\mathbf{V}$ we can bijectively associate $\mathcal{L}(\mathbf{V})$ the variety of languages whose syntactic monoids belong to $\mathbf{V}$ and, conversely, to each variety of languages $\mathcal{V}$ we can bijectively associate $\mathbf{M}(\mathcal{V})$ the variety of monoids generated by the syntactic monoids of the languages of $\mathcal{V}$, and these correspondences are mutually inverse.

We denote by $\mathcal{SUL}_k$ the class of regular languages that are Boolean combinations of languages in $\mathcal{SUM}_k$.

In this appendix, we show that, for all $k \in \mathbb{N}$, $\mathcal{SUL}_k$ is a variety of languages. As $\mathbf{DA_k}$ is the variety of monoids generated by the syntactic monoids of the languages in $\mathcal{SUL}_k$, by Eilenberg's theorem, we know that, conversely, all the regular languages whose syntactic monoids lie in $\mathbf{DA_k}$ are in $\mathcal{SUL}_k$.

Closure under Boolean operations is obvious by construction. Closure under quotients and inverses of morphisms is respectively given by the following two lemmas and by the fact that both quotients and inverses of morphisms commute with Boolean operations.

Given a word u over a given finite alphabet Σ, we will denote by $\mathrm{alph}(u)$ the set of letters of Σ that appear in u.

▶ **Lemma 15.** *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k$ over a finite alphabet Σ and $u \in \Sigma^*$, $u^{-1}L$ and Lu^{-1} both are a union of languages in $\mathcal{SUM}_k$ over Σ.*

Proof. We prove it by induction on k.

- Base case: $k = 0$.

Let $L \in \mathcal{SUM}_0$ over a finite alphabet Σ and $u \in \Sigma^*$. This means that $L = A^*$ for some $A \subseteq \Sigma$. We have two cases: either $\mathrm{alph}(u) \nsubseteq A$ and then $u^{-1}L = Lu^{-1} = \emptyset$; or $\mathrm{alph}(u) \subseteq A$ and then $u^{-1}L = Lu^{-1} = A^* = L$. So $u^{-1}L$ and Lu^{-1} both are a union of languages in $\mathcal{SUM}_0$ over Σ. The base case is hence proved.

- Inductive step. Let $k \in \mathbb{N}_{>0}$ and assume that the lemma is true for all $k' \in \mathbb{N}, k' < k$.
  Let $L \in \mathcal{SUM}_k$ over a finite alphabet Σ and $u \in \Sigma^*$. This means that either L is in $\mathcal{SUM}_{k-1}$ and the lemma is proved by applying the inductive hypothesis directly for L and u, or $L = L_1 a L_2$ for some languages $L_1 \in \mathcal{SUM}_i$ and $L_2 \in \mathcal{SUM}_j$ and some letter $a \in \Sigma$ with $i + j = k - 1$ and, either no word of L_1 contains the letter a or no word of L_2 contains the letter a. We shall only treat the case in which a does not appear in any of the words of L_1; the other case is treated symmetrically.
  There are again two cases to consider, depending on whether a does appear in u or not. If $a \notin \mathrm{alph}(u)$, then it is straightforward to check that $u^{-1}L = (u^{-1}L_1)aL_2$ and $Lu^{-1} = L_1 a(L_2 u^{-1})$. By the inductive hypothesis, we get that $u^{-1}L_1$ is a union of languages in $\mathcal{SUM}_i$ over Σ and that $L_2 u^{-1}$ is a union of languages in $\mathcal{SUM}_j$ over Σ. Moreover, it is direct to see that no word of $u^{-1}L_1$ contains the letter a. By distributivity of concatenation over union, we finally get that $u^{-1}L$ and Lu^{-1} both are a union of languages in $\mathcal{SUM}_k$ over Σ.
  If $a \in \mathrm{alph}(u)$, then let $u = u_1 a u_2$ with $u_1, u_2 \in \Sigma^*$ and $a \notin \mathrm{alph}(u_1)$. It is again straightforward to see that

$$u^{-1}L = \begin{cases} {u_2}^{-1}L_2 & \text{if } u_1 \in L_1 \\ \emptyset & \text{otherwise} \end{cases}$$

  and

$$Lu^{-1} = L_1 a(L_2 u^{-1}) \cup \begin{cases} L_1 {u_1}^{-1} & \text{if } u_2 \in L_2 \\ \emptyset & \text{otherwise} \end{cases}.$$

  Again, by the inductive hypothesis, we get that $L_1 {u_1}^{-1}$ is a union of languages in $\mathcal{SUM}_i$ over Σ and that both ${u_2}^{-1}L_2$ and $L_2 u^{-1}$ are unions of languages in $\mathcal{SUM}_j$ over Σ. And, again, by distributivity of concatenation over union, we get that $u^{-1}L$ and Lu^{-1} both are a union of languages in $\mathcal{SUM}_k$ over Σ.
  This concludes the inductive step and therefore the proof of the lemma. ◀

▶ **Lemma 16.** *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k$ over a finite alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism of monoids where Γ is another finite alphabet, $\varphi^{-1}(L)$ is a union of languages in $\mathcal{SUM}_k$ over Γ.*

Proof. We prove it by induction on k.

- Base case: $k = 0$.
  Let $L \in \mathcal{SUM}_0$ over a finite alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism of monoids where Γ is another finite alphabet. This means that $L = A^*$ for some $A \subseteq \Sigma$. It is straightforward to check that $\varphi^{-1}(L) = B^*$ where $B = \{b \in \Gamma \mid \varphi(b) \in A^*\}$. B^* is certainly a union of languages in $\mathcal{SUM}_0$ over Σ. The base case is hence proved.
- Inductive step. Let $k \in \mathbb{N}_{>0}$ and assume that the lemma is true for all $k' \in \mathbb{N}, k' < k$.
  Let $L \in \mathcal{SUM}_k$ over a finite alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism of monoids where Γ is another finite alphabet. This means that either L is in $\mathcal{SUM}_{k-1}$ and the lemma is proved by applying the inductive hypothesis directly for L and φ, or $L = L_1 a L_2$ for

some languages $L_1 \in \mathcal{SUM}_i$ and $L_2 \in \mathcal{SUM}_j$ and some letter $a \in \Sigma$ with $i + j = k - 1$ and, either no word of L_1 contains the letter a or no word of L_2 contains the letter a. We shall only treat the case in which a does not appear in any of the words of L_1; the other case is treated symmetrically.
Let us define $B = \{b \in \Gamma \mid a \in \text{alph}(\varphi(b))\}$ as the set of letters of Γ whose image word by φ contains the letter a. For each $b \in B$, we shall also let $\varphi(b) = u_{b,1}au_{b,2}$ with $u_{b,1}, u_{b,2} \in \Sigma^*$ and $a \notin u_{b,1}$. It is not too difficult to see that we then have

$$\varphi^{-1}(L) = \bigcup_{b \in B} \varphi^{-1}(L_1 {u_{b,1}}^{-1}) b \varphi^{-1}({u_{b,2}}^{-1} L_2) \ .$$

By the inductive hypothesis, by Lemma 15 and by the fact that inverses of morphisms commute with unions, we get that $\varphi^{-1}(L_1 {u_{b,1}}^{-1})$ is a union of languages in $\mathcal{SUM}_i$ over Γ and that $\varphi^{-1}({u_{b,2}}^{-1} L_2)$ is a union of languages in $\mathcal{SUM}_j$ over Γ. Moreover, it is direct to see that no word of $\varphi^{-1}(L_1 {u_{b,1}}^{-1})$ contains the letter b for all $b \in B$. By distributivity of concatenation over union, we finally get that $\varphi^{-1}(L)$ is a union of languages in $\mathcal{SUM}_k$ over Γ.
This concludes the inductive step and therefore the proof of the lemma. ◀

A.3 Collapse

In this appendix we prove Proposition 12, stating that when M is in $\mathbf{DA_k}$ then any program over M is equivalent to one of length $\mathrm{O}(n^k)$.

Recall that if P is a program over some monoid M of range n, then $P(w)$ denotes the element of M resulting from the execution of the program P on w. It will be convenient here to also work with the word over M resulting from the sequence of executions of each instruction of P on w. We denote this word by $EP(w)$.

The result is a consequence of the following lemma and the fact that for any acceptance set $F \subseteq M$, a word $w \in \Sigma^n$ (where Σ is the input alphabet) is accepted iff $EP(w) \in L$ where L is a language in $\mathcal{SUL}_k$, a Boolean combination of languages in $\mathcal{SUM}_k$.

▶ **Lemma 17.** *Let Σ be a finite alphabet, M a finite monoid, and n, k natural numbers.*

For any program P over M of range n, any set $\Gamma \subseteq M$ and any language K over Γ in $\mathcal{SUM}_k$, there exists a subsequence Q of the sequence of instructions P of length $\mathrm{O}(n^{\max\{k,1\}})$ such that for any subsequence Q' of the sequence of instructions P containing Q as a subsequence, we have for all words w over Σ of length n:

$$EP(w) \in K \Leftrightarrow EQ'(w) \in K \ .$$

Proof. A program P over M of range n is a finite sequence (p_i, f_i) of instructions where each p_i is a natural number which is at most n and each f_i is a function from Σ to M. We denote by l the number of instructions of P. For each set $I \subseteq [l]$ we denote by $P[I]$ the program over M consisting of the subsequence of instructions of P obtained after removing all instructions whose index is not in I. In particular, $P[1, m]$ denotes the initial sequence of instructions of P, until instruction number m.

We prove the lemma by induction on k.

- Inductive step. Let $k \geq 2$ and assume the lemma proved for all $k' < k$. Let n be a natural number, P a program over M of range n and length l, $\Gamma \subseteq M$ and any language K over Γ in $\mathcal{SUM}_k$. By definition, $K = K_1 \gamma K_2$ for some languages $K_1 \in \mathcal{SUM}_i$ and $K_2 \in \mathcal{SUM}_j$ with $i + j = k - 1$. Moreover either γ does not occur in any of the words of

K_1 or it does not occur in any of the words of K_2. We only treat the case where γ does not appear in any of the words in K_1. The other case is treated similarly by symmetry.
For each $p \leq n$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \gamma$. We let I_γ be the set of indices of these instructions for all a and p. Notice that the size of I is in $\mathrm{O}(n)$.
For all $i \in I_\gamma$, we let $J_{i,1}$ be the set of indices of the instructions within $P[1, i-1]$ obtained by induction for K_1 and $J_{i,2}$ be the same for $P[i+1, l]$ and K_2.
We now let I be the union of I_γ and $J_{i,1}$ and $J'_{i,2} = \{j + i \mid j \in J_{i,2}\}$ for all $i \in I_\gamma$. We claim that $P[I]$ has the desired properties.
First notice that by induction the sizes of $J_{i,1}$ and $J'_{i,2}$ for all $i \in I_\gamma$ are in $\mathrm{O}(n^{\max\{k-1,1\}})$ $= \mathrm{O}(n^{k-1})$ and because the size of I_γ is linear in n, the size of I is in $\mathrm{O}(n^k) = \mathrm{O}(n^{\max\{k,1\}})$ as required.
Now take $w \in \Sigma^n$.
Assume now that $EP(w) \in K$. Let i be the position in $EP(w)$ of label γ witnessing the membership in K. Let (p_i, f_i) be the corresponding instruction of P. In particular we have that $f_i(w_{p_i}) = \gamma$. Because γ does not occur in any word of K_1, for all $j < i$ such that $p_j = p_i$ we cannot have $f_j(w_{p_j}) = \gamma$. Hence $i \in I_\gamma$. By induction we have that $EP[1, i-1][J](w) \in K_1$ for any set J containing $J_{i,1}$ and $EP[i+1, l][J](w) \in K_2$ for any set J containing $J_{i,2}$. Hence, if we set $I_1 = \{j \in I \mid j < i\}$ as the subset of I of elements less than i and $I_2 = \{j - i \in I \mid j > i\}$ as the subset of I of elements greater than i translated by $-i$, we have $EP[I](w) = EP[1, i-1][I_1](w)\gamma EP[i+1, l][I_2](w) \in K_1\gamma K_2 = K$ as desired.
Assume finally that $EP[I](w) \in K$. Let i be the index in I whose instruction provides the letter γ witnessing the fact that $EP[I](w) \in K$. If $i \in I_\gamma$ we conclude easily by induction. If not this shows that there is an instruction (p_j, f_j) with $j < i$, $j \in I$, $p_j = p_i$ and $f_j(w_{p_j}) = \gamma$. But that would contradict the fact that γ cannot occur in K_1.

- Base case. There are two subcases to consider.
  - $k = 1$. Let n be a natural number, P a program over M of range n and length l, $\Gamma \subseteq M$ and any language K over Γ in $\mathcal{SUM}_1$.
    Then $K = A_1^*\gamma A_2^*$ for some finite alphabets $A_1 \subseteq \Gamma$ and $A_2 \subseteq \Gamma$. Moreover either $\gamma \notin A_1$ or $\gamma \notin A_2$. We only treat the case where γ does not belong to A_1, the other case is treated similarly by symmetry.
    We use the same idea as in the inductive step.
    For each $p \leq n$, each $\alpha \in \Gamma$ and $a \in \Sigma$ consider within the sequence of instructions of P the first and last instruction of the form (p, f) with $f(a) = \alpha$. We let I be the set of indices of these instructions for all a, α and p. Notice that the size of I is in $\mathrm{O}(n) = \mathrm{O}(n^{\max\{k,1\}})$.
    We claim that $P[I]$ has the desired properties. Take $w \in \Sigma^n$.
    Assume now that $EP(w) \in K$. Let i be the position in $EP(w)$ of label γ witnessing the membership in K. Let (p_i, f_i) be the corresponding instruction of P. In particular we have that $f_i(w_{p_i}) = \gamma$ and this is the γ witnessing the membership in K. Because $\gamma \notin A_1$, for all $j < i$ such that $p_j = p_i$ we cannot have $f_j(w_{p_j}) = \gamma$. Hence $i \in I$. From $EP(w) \in K$ it follows that $EP[I \cap [\![1, i-1]\!]](w) \in A_1$ and $EP[I \cap [\![i+1, l]\!]](w) \in A_2$ showing that $EP[I](w) = EP[I \cap [\![1, i-1]\!]](w)\gamma EP[I \cap [\![i+1, l]\!]](w) \in K$ as desired.
    Assume finally that $EP[I](w) \in K$. This means that $EP[I \cap [\![1, i-1]\!]](w) \in A_1^*$ and $EP[I \cap [\![i+1, l]\!]](w) \in A_2^*$. Let i be the index in I whose instruction provides the letter γ witnessing the fact that $EP[I](w) \in K$. If there is an instruction (p_j, f_j), with $j < i$ and $f_j(w_{p_j}) \notin A_1$ then either $j \in I$ and we get a direct contradiction with the

fact that $EP[I \cap [\![1, i-1]\!]](w) \in A_1^*$, or $j \notin I$ and we get a smaller $j' \in I$ with the same property, contradicting again the fact that $EP[I \cap [\![1, i-1]\!]](w) \in A_1^*$. Hence for all $j < i$, $f_j(w_{p_j}) \in A_1$. By symmetry we have that for all $j > i$, $f_j(w_{p_j}) \in A_2$, showing that $EP(w) \in A_1^* \gamma A_2^* = K$ as desired.

- $k = 0$. Let n be a natural number, P a program over M of range n and length l, $\Gamma \subseteq M$ and any language K over Γ in $\mathcal{SUM}_0$.
Then $K = A^*$ for some finite alphabet $A \subseteq \Gamma$.
We again use the same idea as before.
For each $p \leq n$, each $\alpha \in \Gamma$ and $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \alpha$. We let I be the set of indices of these instructions for all a, α and p. Notice that the size of I is in $\mathrm{O}(n) = \mathrm{O}(n^{\{k,1\}})$.
We claim that $P[I]$ has the desired properties. Take $w \in \Sigma^n$.
Assume now that $EP(w) \in K$. As $EP[I](w)$ is a subword of $EP(w)$, it follows directly that $EP[I](w) \in A^* = K$ as desired.
Assume finally that $EP[I](w) \in K$. If there is an instruction (p_j, f_j), with $j \in [l]$ and $f_j(w_{p_j}) \notin A$ then either $j \in I$ and we get a direct contradiction with the fact that $EP[I](w) \in A^* = K$, or $j \notin I$ and we get a smaller $j' \in I$ with the same property, contradicting again the fact that $EP[I](w) \in A^* = K$. Hence for all $j \in [l]$, $f_j(w_{p_j}) \in A$, showing that $EP(w) \in A^* = K$ as desired. ◀