Tameness and the power of programs over monoids in DA

Nathan Grosshans, Pierre Mckenzie, Luc Segoufin

TAMENESS AND THE POWER OF PROGRAMS OVER MONOIDS IN DA *

NATHAN GROSSHANS [a], PIERRE MCKENZIE [b], AND LUC SEGOUFIN [c]

[a] Fachbereich Elektrotechnik/Informatik, Universität Kassel, Kassel, Germany
e-mail address: nathan.grosshans@polytechnique.edu
URL: https://nathan.grosshans.me

[b] DIRO, Université de Montréal, Montréal, Canada
e-mail address: mckenzie@iro.umontreal.ca

[c] Inria, DI ENS, ENS, CNRS, PSL University, Paris, France
e-mail address: luc.segoufin@inria.fr

Abstract. The program-over-monoid model of computation originates with Barrington's proof that the model 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. Finally, we exhibit a program-length-based hierarchy within the class of languages recognized by programs over monoids from **DA**.

1. Introduction

A program of range n on alphabet Σ over a finite monoid M is a sequence of pairs (i, f) where $1 \leq i \leq n$ and $f : \Sigma \to M$ is a function. 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 an accepting set $F \subseteq M$ is specified, the program naturally defines the language L_n of words of length n assigned an element in F. A program sequence $(P_n)_{n \in \mathbb{N}}$ then defines the language formed by the union of the L_n.

A program over M is a generalization of a morphism from Σ^* to M, and recognition by a morphism equates with acceptance by a finite automaton. Moving from morphisms to programs has a significant impact on the expressive power as shown by the seminal result of Barrington [Bar89][1] that polynomial-length program sequences over the group S_5

Key words and phrases: Programs over monoids, tameness, DA, lower bounds.

* Revised and extended version of [GMS17] that includes a more inclusive definition of tameness, thus strengthening the statement that **J** is not a tame variety, as explained in Section 3.

[1] in fact extending the scope of an observation made earlier by Maurer and Rhodes [MR65]

Preprint submitted to
Logical Methods in Computer Science

capture the complexity class NC^1 (of languages accepted by bounded fan-in Boolean circuits of logarithmic depth).

Barrington's result was followed by several results strengthening the correspondence between circuit complexity and programs over finite monoids. 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 [Bar89, BT88]. More generally for any variety $\mathbf{V}$ of finite 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 finite aperiodic monoids, then $\mathcal{P}(\mathbf{A})$ characterizes the complexity class AC^0 [BT88]. It was further observed that in a formal sense, understanding the regular languages of $\mathcal{P}(\mathbf{V})$ is sufficient to understand the expressive power of $\mathcal{P}(\mathbf{V})$ (see [MPT91], but also [Str94] for a logical point of view).

In view of the above results it is plausible that algebraic automata theory methods could help separating complexity classes within NC^1. But although partial results in restricted settings were obtained, no breakthrough was achieved this way.

The reason of course is that programs are much more complicated than morphisms: programs can read the letter at an input position more than once, in non-left-to-right order, possibly assigning a different monoid element each time. This complication can be illustrated with the following example. Consider the variety of finite monoids known as $\mathbf{J}$. This is the variety generated by the syntactic monoids of all languages defined by the presence or absence of certain *subwords*, where u is a subword of v if u can be obtained from v by deleting letters [Sim75]. One deduces that monoids in $\mathbf{J}$ are unable to morphism-recognize the language defined by the regular expression $(a+b)^*ac^+$. Yet a sequence of programs over a monoid in $\mathbf{J}$ recognizes $(a+b)^*ac^+$ by the following trick. Consider the language L of all words having ca as a subword but having as subwords neither cca, caa nor cb. Being defined by the occurrence of subwords, L *is* recognized by a morphism $\varphi : \{a,b,c\}^* \to M$ where $M \in \mathbf{J}$, i.e., for this φ there is an $F \subseteq M$ such that $L = \varphi^{-1}(F)$. Here is the trick: the program of range n over M given by the sequence of instructions

$$(2,\varphi),(1,\varphi),(3,\varphi),(2,\varphi),(4,\varphi),(3,\varphi),(5,\varphi),(4,\varphi),\cdots,(n,\varphi),(n-1,\varphi),$$

using F as accepting set, defines the set of words of length n in $(a+b)^*ac^+$. 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. (See [Gro20, Lemma 4.1] for a full proof of the fact that $(a+b)^*ac^+ \in \mathcal{P}(\mathbf{J})$.)

The first part of our paper addresses the question of what are the regular languages in $\mathcal{P}(\mathbf{V})$. As mentioned above, this is the key to understanding the expressive power of $\mathcal{P}(\mathbf{V})$.

Observe first that the class $\mathcal{L}(\mathbf{V})$ of all languages recognized by a morphism into a monoid in $\mathbf{V}$ is trivially included in $\mathcal{P}(\mathbf{V})$. It turns out that $\mathcal{P}(\mathbf{V})$ always contains more regular languages. For instance, because a program instruction (i,f) operating on a word w is "aware" of i, the program can have a behavior depending on some arithmetic properties of i. Moreover, a program's behavior can in general depend on the length of the input words it treats. So in particular, as far as regular languages are concerned, a program for a given input length can take into account the length of the input modulo some fixed number k in its acceptance set and each program instruction (i,f) can depend on the value of i modulo k. This can be formalized by assuming without loss of generality that membership can depend on the length of the word w at hand modulo a fixed number k and that each letter in w is tagged with its position modulo k. Regular languages recognized this way are exactly the languages recognized by a stamp (a surjective morphism from Σ^* to M with Σ an alphabet and M a finite monoid) in the variety of stamps $\mathbf{V} * \mathbf{Mod}$, where $*$ is the wreath product of stamps and $\mathbf{Mod}$ the variety of cyclic stamps into groups. In other words, $\mathcal{L}(\mathbf{V} * \mathbf{Mod})$ is always included in $\mathcal{P}(\mathbf{V})$.

A program over a monoid can also recognize regular languages by changing its behavior depending on *bounded-length prefixes* and *suffixes* arbitrarily. To formalize this, we introduce the class $\mathbf{EV}$ of stamps that, modulo the beginning and the end of a word, behave essentially like stamps into monoids from $\mathbf{V}$. It is then not too hard to show that $\mathcal{L}(\mathbf{EV} * \mathbf{Mod})$ is always included in $\mathcal{P}(\mathbf{V})$ when $\mathbf{V}$ does contain non-trivial monoids. Many varieties $\mathbf{V}$ are such that $\mathcal{P}(\mathbf{V})$ cannot recognize more regular languages than those in $\mathcal{L}(\mathbf{EV} * \mathbf{Mod})$. This is the case for example of the variety $\mathbf{DA}$ as we will see below.

Our first result characterizes those varieties $\mathbf{V}$ having the property that $\mathcal{P}(\mathbf{V})$ does not contain "many more" regular languages than does $\mathcal{L}(\mathbf{EV} * \mathbf{Mod})$. To this end we introduce the notion of *tameness* for a variety of finite monoids $\mathbf{V}$ (Definition 3.9) and our first result shows that a variety of finite monoids $\mathbf{V}$ is tame if and only if $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QEV})$. Here, $\mathcal{L}(\mathbf{QV})$ is the class of regular languages recognized by stamps in quasi-$\mathbf{V}$. A stamp φ from Σ^* to M is in quasi-$\mathbf{V}$ if, though M might not be in $\mathbf{V}$, its stable monoid induced by φ is in $\mathbf{V}$, i.e. there is a number k such that $\varphi((\Sigma^k)^*)$ forms a submonoid of M which is in $\mathbf{V}$. For tame varieties $\mathbf{V}$ we do not know when the inclusion of $\mathcal{L}(\mathbf{EV} * \mathbf{Mod})$ in $\mathcal{L}(\mathbf{QEV})$ is strict or not. In particular we do not know when the inclusion in our result is an equality. As usual in this context, we conjecture that equality holds at least for local varieties $\mathbf{V}$.

Our notion of a tame variety differs subtly but fundamentally from the notion of p-variety (program-variety). This notion goes back to Péladeau [Pél90] and can be stated by saying that a variety of finite monoids $\mathbf{V}$ is a p-variety whenever any monoid that can be "simulated" by programs over a monoid in $\mathbf{V}$ belongs itself to $\mathbf{V}$. Equivalently, $\mathbf{V}$ is a p-variety whenever any regular language in $\mathcal{P}(\mathbf{V})$ with a neutral letter (a letter which can be inserted and deleted arbitrarily in words without changing their membership in the language) is in fact morphism-recognized by a monoid in $\mathbf{V}$. (The equivalence between the two definitions is claimed without a proof in [Tes03] and [Gro18], see [PST97] for a proof in one direction, the other direction requiring a standard argument.) While understanding the neutral letter regular languages in $\mathcal{P}(\mathbf{V})$ for $\mathbf{V}$ ranging over all possible varieties of finite monoids would suffice to solve most open questions about the internal structure of NC^1, for $\mathbf{V}$ to be a p-variety does not imply a precise characterization of *all* the regular languages in $\mathcal{P}(\mathbf{V})$. It can be proved that if $\mathbf{V}$ is a p-variety then the regular languages in $\mathcal{P}(\mathbf{V})$ are all in $\mathcal{L}(\mathbf{QLV})$, where $\mathbf{LV}$ is the inclusion-wise largest variety of finite semigroups containing all monoids in $\mathbf{V}$ and only those monoids. For instance, $\mathbf{DA}$ is a p-variety [LTT06], and this implies that $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QLDA})$ as explained above. The latter inclusion is strict and the correct characterization, namely $\mathcal{L}(\mathbf{QEDA})$, requires proving that $\mathbf{DA}$ is also tame in our sense. Furthermore, there exist p-varieties for which unexpected (and interesting) things happen when considering program-recognition of regular languages without neutral letter, and this precisely because they aren't tame. For example, $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QLJ})$ with strict inclusion while it will follow from our result that $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \nsubseteq \mathcal{L}(\mathbf{QEJ})$ (knowing that it is easy to check that $\mathcal{L}(\mathbf{QEJ}) \subseteq \mathcal{L}(\mathbf{QLJ})$).

The situation for programs over finite semigroups of the form $\mathbf{V} * \mathbf{D}$, where $\mathbf{V}$ is a variety of finite monoids and $\mathbf{D}$ is the variety of finite definite (or righty trivial) semigroups, turns out to be much simpler. Indeed, with the necessary adaptations to the notion of p-variety, Péladeau, Straubing and Thérien [PST97] could show that for any p-variety of the form $\mathbf{V} * \mathbf{D}$ we have $\mathcal{P}(\mathbf{V} * \mathbf{D}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{Q}(\mathbf{V} * \mathbf{D}))$. Once our notion of tameness is adapted for finite semigroups, it is possible to show that any p-variety of the form $\mathbf{V} * \mathbf{D}$ is tame. Hence the result of [PST97] mentioned above follows from our result as for varieties of the form $\mathbf{V} * \mathbf{D}$ we have, abusing notation, that $\mathbf{E}(\mathbf{V} * \mathbf{D}) = \mathbf{V} * \mathbf{D}$ and that $\mathcal{L}(\mathbf{Q}(\mathbf{V} * \mathbf{D})) = \mathcal{L}(\mathbf{V} * \mathbf{D} * \mathbf{Mod}) \subseteq \mathcal{P}(\mathbf{V} * \mathbf{D})$. It is to be noted that programs over semigroups in $\mathbf{V} * \mathbf{D}$ correspond to Straubing's k-programs over monoids in $\mathbf{V}$ [Str00, Str01]. It is also possible to prove that regular languages recognized by monoids from a k-program variety $\mathbf{V}$, as for p-varieties, are all in $\mathcal{L}(\mathbf{QLV})$. Interestingly, in order to get a tight characterization for the regular languages recognized by k-programs over commutative monoids in [Str01], Straubing determines not only which monoids can be simulated by such k-programs, but, in our terms, which *stable stamps* those k-programs can "simulate". Our notion of tameness also builds upon stable stamps and, as we have advocated above, subsumes previous definitions of "good behavior" of programs with respect to recognition of regular languages.

Tameness as defined here is also a proper extension of the notion of *sp*-varieties of monoids (Definition 3.3), a concept introduced in [GMS17] as our initial attempt to capture the expected behavior of programs over small varieties. We will for instance see that the variety of finite commutative monoids is tame but not an *sp*-variety.

Showing that a variety is tame can be a difficult task. For instance showing that the variety $\mathbf{A}$ is tame amounts to showing that the regular languages in AC^0 are in $\mathcal{L}(\mathbf{QA})$, which, as shown by Barrington, Compton, Straubing and Thérien [BCST92], follows from the fact that modulo counting cannot be done in AC^0 (the famous result initially proven by Furst, Saxe and Sipser [FSS84] and independently by Ajtai [Ajt83]). Similarly much of the structure of NC^1 would in fact be resolved by showing the tameness of certain varieties (see [MPT91, Corollary 4.13], [Str94, Conjecture IX.3.4]).

The present work is motivated by the need to better understand the subtle behaviors of polynomial-length programs over monoids. We focus in this paper on 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 [TT02b] for an eloquent testimony). In particular $\mathcal{P}(\mathbf{DA})$ corresponds to languages accepted by decision trees of bounded rank [GT03]. It is also known that regular languages with a neutral letter that are in $\mathcal{P}(\mathbf{DA})$ are also in $\mathcal{L}(\mathbf{DA})$ [LTT06].

Our second result shows that the variety $\mathbf{DA}$ is tame. As it is easy to see that $\mathbf{DA}$ is powerful enough to describe prefixes and suffixes of words up to some bounded length, we get that $\mathcal{L}(\mathbf{EDA}) = \mathcal{L}(\mathbf{DA})$. Moreover, because $\mathbf{DA}$ is a local variety, $\mathbf{QDA} = \mathbf{DA} * \mathbf{Mod}$ [DP13]. Altogether the tameness of $\mathbf{DA}$ implies that the regular languages in $\mathcal{P}(\mathbf{DA})$ are precisely the languages in $\mathcal{L}(\mathbf{QDA})$.

Our third result is that, on the other hand, the variety of finite monoids $\mathbf{J}$ is not tame as witnessed by the regular language $(a+b)^*ac^+$ discussed above which is in $\mathcal{P}(\mathbf{J})$ but not in $\mathcal{L}(\mathbf{QEJ})$. Characterizing the regular languages in $\mathcal{P}(\mathbf{J})$ remains an open problem, partially solved in [Gro20].

Our final result concerns $\mathcal{P}(\mathbf{DA})$. With $\mathcal{C}_k$ the class of languages recognized by programs of length $\mathrm{O}(n^k)$ over $\mathbf{DA}$, 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 $\mathbf{DA}$ 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 [TT02a].

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

2. Preliminaries

This section is dedicated to the introduction of 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 [Eil74, Eil76] and Pin [Pin86].

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. Since all our alphabets and words in this article are always finite, we shall not mention it anymore from here on. Given some word $w \in \Sigma$, we denote its length by $|w|$ and, for any $a \in \Sigma$, by $|w|_a$ the number of occurrences of the letter a in w. 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 non-empty 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. With the exception of free monoids and semigroups, all monoids and semigroups considered here are finite. 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; unless otherwise stated, when we say "morphism", we always mean "monoid morphism". Any morphism $\varphi\colon \Sigma^* \to M$ for Σ an alphabet and M some monoid is uniquely determined by the images of the letters of Σ by φ. 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. A semigroup T *divides* a semigroup S if T is the image by a semigroup morphism of a subsemigroup of S. Division of monoids is defined in the same way by replacing any occurrence of "semigroup" by "monoid". 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 finite monoids* is a non-empty class of finite monoids closed under Cartesian product and monoid division. A *variety of finite semigroups* is defined similarly. When dealing with varieties, we consider only varieties of finite monoids or semigroups, so we will drop the adjective "finite" when talking about those.

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 power of* S, often denoted ω, such that for any element $s \in S$, s^ω is idempotent.

A general result of Reiterman [Rei82] states that each variety of monoids (or semigroups) can be defined as the class of all finite monoids satisfying some set of identities, for an appropriate notion of identity. In our case, we only use a restricted version of this notion of an identity, that we understand as a formal equality of terms built on the basis of variables by using products and ω-powers. A finite monoid is then said to satisfy such an identity whenever the equality is verified for any setting of the variables to elements of the monoid, interpreting the ω-power as the idempotent power of that monoid. For instance, the variety of finite aperiodic monoids $\mathbf{A}$, known as the variety of "group-free" finite monoids (i.e. those verifying that they do not have any non-trivial group as a subsemigroup), is defined by the identity $x^\omega = x^{\omega+1}$. 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 identities $(xy)^\omega = (xy)^\omega x = y(xy)^\omega$. One easily deduces that $\mathbf{J} \subseteq \mathbf{DA} \subseteq \mathbf{A}$.

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

Eilenberg showed [Eil76, 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.

When $\mathbf{V}$ is a variety of semigroups, we will denote by $\mathcal{L}(\mathbf{V})$ the class of languages whose syntactic semigroup belongs to $\mathbf{V}$. There is also an Eilenberg-type correspondence for an appropriate notion of language varieties, that is *ne*-varieties (non-erasing-varieties) of languages, but we won't present it here. (The interested reader may have a look at [Str02] as well as [PS05, Lemma 6.3].)

Quasi and locally $\mathbf{V}$ 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 [PS05, CPS06b]. Let φ be a surjective morphism from Σ^*, for Σ some alphabet, to a finite monoid M: such a morphism is called a *stamp*. 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 and this k is called the *stability index* of φ. If 1 is the identity of M, then $\varphi(\Sigma^k) \cup \{1\}$ 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 stamps 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.

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 instead give a characterization of the languages recognized by such algebraic objects [CPS06a, Til87].

Let $\mathbf{V}$ be a variety of monoids. We say that a language over Σ is in $\mathcal{L}(\mathbf{V} * \mathbf{Mod})$ if it is obtained by a finite combination of unions and intersections of languages over Σ for which membership of each word over Σ only depends on its length modulo some integer $k \in \mathbb{N}_{>0}$ and languages L over Σ for which there is a number $k \in \mathbb{N}_{>0}$ and a language L' over $\Sigma \times \{0, \ldots, k-1\}$ 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. Observe that neither $\mathbf{V} * \mathbf{Mod}$ nor $\mathbf{QV}$ are varieties of monoids or semigroups, but classes of stamps that happen to be varieties of stamps of a certain kind[2], that we won't introduce.

Similarly we say that a language over Σ is in $\mathcal{L}(\mathbf{V} * \mathbf{D})$ if it is obtained by a finite combination of unions and intersections of languages over Σ for which membership of each word over Σ only depends on its $k \in \mathbb{N}$ last letters and languages L over Σ for which 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 of monoids $\mathbf{V}$ 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 [Til87, Theorem 17.3]. One of the consequences of locality that we will use is that $\mathcal{L}(\mathbf{V} * \mathbf{Mod}) = \mathcal{L}(\mathbf{QV})$ when $\mathbf{V}$ is local [DP14, Corollary 18], while $\mathcal{L}(\mathbf{V} * \mathbf{Mod}) \subseteq \mathcal{L}(\mathbf{QV})$ in general (see [Dar14, Pap14]).

Programs over varieties of monoids. Programs over monoids form a non-uniform model of computation, first defined by Barrington and Thérien [BT88], extending Barrington's permutation branching program model [Bar89]. Let M be a finite monoid and Σ an 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 (i, f) verify $1 \leq i \leq n$. A program P of range n defines a function from Σ^n, the words of length n, to M as follows. On input $w \in \Sigma^n$, for $w = w_1 \cdots w_n$, 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. The only program of range 0, the empty one, always outputs the identity of M.

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 recognizes $L \cap \Sigma^n$, that is, there exists a subset

[2]To be precise, both are *lm*-varieties of stamps, as defined in [Str02].

F_n of M such that $L \cap \Sigma^n$ is precisely the set of words w of length n such that $P_n(w) \in F_n$. In that case, we also say that L is p-recognized by M.

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 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 2.1. [MPT91, Corollary 3.5] *For* $\mathbf{V}$ *any variety of monoids,* $\mathcal{P}(\mathbf{V})$ *is closed under Boolean operations, quotients and inverse images of* lm-*morphisms.*

Given two range n programs P, P' over some monoid M using the same input alphabet Σ, we shall say that P' is a *subprogram*, a *prefix* or a *suffix* of P whenever P' is, respectively, a subword, a prefix or a suffix of P, looking at P and P' as words over $[n] \times M^{\Sigma}$.

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 $\mathbf{V}$. We will see in Section 3.2 that programs over some monoid in $\mathbf{V}$ can also p-recognize the regular languages that are "essentially $\mathbf{V}$" i.e. that differ from a language in $\mathcal{L}(\mathbf{V})$ only on the prefix and suffix of the words.

In this section we characterize those varieties $\mathbf{V}$ such that programs over monoids in $\mathbf{V}$ do not recognize more regular languages than those mentioned above.

We first recall the definitions and results around p-varieties developed by Péladeau, Tesson, Straubing and Thérien and then present the definition of *sp*-varieties that was inspired by their work and studied in the conference version of the present paper. In order to deal with the limitation of *sp*-varieties we then define the notion of essentially-$\mathbf{V}$ that will be the last ingredient for our definition of tameness. We then provide an upper bound on the regular languages that can be p-recognized by a sequence of programs over a monoid from a tame variety $\mathbf{V}$.

3.1. *p*- and *sp*-varieties of monoids.

We first recall the definition of p-varieties. These seem to have been originally defined by Péladeau in his Ph.D. thesis [Pél90] and later used by Tesson in his own Ph.D. thesis [Tes03]. The notion of a p-variety has also been defined for semigroups by Péladeau, Straubing and Thérien in [PST97].

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 condition to be a p-variety requires a converse of this observation.

Definition 3.1. An p-*variety of monoids* is a variety $\mathbf{V}$ of monoids such that for any finite monoid M, if $\mathcal{W}(M) \subseteq \mathcal{P}(\mathbf{V})$ then $M \in \mathbf{V}$.

The following result illustrates an important property of p-varieties, when the notion is adapted to varieties of semigroups accordingly.

Proposition 3.2. [PST97] *Let* $\mathbf{V} * \mathbf{D}$ *be a* p*-variety of semigroups, where* $\mathbf{V}$ *is a variety of monoids.*

Then $\mathcal{P}(\mathbf{V} * \mathbf{D}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{V} * \mathbf{D} * \mathbf{Mod})$ *(where the latter class is defined in the same way as* $\mathcal{L}(\mathbf{V} * \mathbf{Mod})$*).*

It is known that $\mathbf{J}$ is a p-variety of monoids [Tes03] but as we have seen in the introduction, $\mathcal{P}(\mathbf{J})$ contains languages that are more complicated than those in $\mathcal{L}(\mathbf{J} * \mathbf{Mod})$ (see the end of this subsection for a proof). In order to capture those varieties for which programs are well behaved we need a restriction of p-varieties and this brings us to the following definition.

Definition 3.3. An sp-*variety of monoids* is a variety $\mathbf{V}$ of monoids such that for any finite semigroup S, if $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$ then $S^1 \in \mathbf{V}$.

Hence any *sp*-variety of monoids is also a p-variety of monoids, but the converse is not always true as we will see in Proposition 3.6 below 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 $|w|_a$ is a multiple of k for $w \in \{a, b\}^*$ cannot be done in $\mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ [FSS84, Ajt83] (we shall denote the corresponding language over the alphabet $\{0, 1\}$ by $\mathrm{MOD_k}$). 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 $x^\omega \neq x^{\omega+1}$, we have $k > 1$, so that $L \notin \mathcal{P}(\mathbf{A})$ by [FSS84, Ajt83]. Let $\eta_S\colon S^* \to S^1$ be the evaluation morphism of S. The morphism $\varphi\colon \Sigma^* \to S^*$ sending each letter $a \in \Sigma$ to $\mu(a)$ verifies that $\mu = \eta_S \circ \varphi$, so that $L = \mu^{-1}(x^\omega) = (\eta_S \circ \varphi)^{-1}(x^\omega) = \varphi^{-1}(\eta_S^{-1}(x^\omega))$. From $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{A})$ it follows that $\eta_S^{-1}(x^\omega) \in \mathcal{P}(\mathbf{A})$, hence since φ sends each letter of Σ to a letter of S, it is an *lm*-morphism and as $\mathcal{P}(\mathbf{A})$ is closed under inverses of *lm*-morphisms by Lemma 2.1, we have $L = \varphi^{-1}(\eta_S^{-1}(x^\omega)) \in \mathcal{P}(\mathbf{A})$, a contradiction.

The following is the desired consequence of being an *sp*-variety of monoids.

Proposition 3.4. *Let* $\mathbf{V}$ *be an* sp*-variety of monoids. Then* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$.

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 2.1.

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 [Pin86, Chapter 2, proof of Lemma 2.6] shows that L'' is a Boolean combination of quotients of L. By Lemma 2.1, 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 2.1. $\square$

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

Proposition 3.5. *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 $\mathcal{L}(\mathbf{QV}) = \mathcal{L}(\mathbf{V} * \mathbf{Mod})$ (see [DP14]). The result can then be derived using Proposition 3.4, as we always have $\mathcal{L}(\mathbf{V} * \mathbf{Mod}) \subseteq \mathcal{P}(\mathbf{V})$. $\square$

As $\mathbf{A}$ is local [Til87, Example 15.5] and an *sp*-variety, it follows from Proposition 3.5 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 [BCST92].

We will see in the next section that $\mathbf{DA}$ is an *sp*-variety. As it is also local [Alm96], we get from Proposition 3.5 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 $\mathcal{L}(\mathbf{QJ})$: just compute the stable monoid of the syntactic morphism of the language, which is equal to the syntactic monoid of the language, that is not in $\mathbf{J}$. Hence, by Proposition 3.4, we have the following result:

Proposition 3.6. $\mathbf{J}$ *is not an* sp*-variety of monoids.*

Despite Proposition 3.6 providing some explanation for the unexpected relative strength of programs over monoids in $\mathbf{J}$, the notion of an *sp*-variety of monoids isn't entirely satisfactory.

We say that a monoid is *trivial* when its underlying set contains a sole element. The class of all trivial monoids, that we will denote by $\mathbf{I}$, forms a variety: it is the sole variety containing only trivial monoids, so we may call it the *trivial variety of monoids*.

One observation to be made is that any non-trivial monoid M *p*-recognizes the language of words over $\{a,b\}$ starting with an a: for the first position in any word, just send a to any element that is not the identity and b to the identity. This means that for any non-trivial variety of monoids $\mathbf{V}$, we have that $a(a+b)^* \in \mathcal{P}(\mathbf{V})$. But since the stable monoid of the syntactic morphism of $a(a+b)^*$ is equal to the syntactic monoid of this language, it follows that for any non-trivial variety of monoids $\mathbf{V}$ not containing the syntactic monoid of $a(a+b)^*$, we have $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \nsubseteq \mathcal{L}(\mathbf{QV})$, hence that $\mathbf{V}$ is not an *sp*-variety of monoids.

Therefore, many varieties of monoids actually aren't *sp*-varieties of monoids simply because of the built-in capacity of programs over any non-trivial monoid to test the first letter of input words. This is for example true for any non-trivial variety containing only groups and for any non-trivial variety containing only commutative monoids. This built-in capacity, additional to programs' ability to do positional modulo counting that underlies the definition of *sp*-varieties, should be taken into account in the notion we are looking for to capture "good behavior". In order to define our notion of tameness we first study this extra capacity that is built-in for programs over $\mathbf{V}$ and that we call "essentially-$\mathbf{V}$".

3.2. **Essentially-V stamps.** It is easy to extend our reasoning above to show that given any non-trivial monoid M and given some $k \in \mathbb{N}_{>0}$, the language of words over $\{a, b\}$ having an a in position k, that is $(a+b)^{k-1}a(a+b)^*$, is p-recognized by M, and the same goes for $(a+b)^*a(a+b)^{k-1}$. By generalizing, we can quickly conclude that given any non-trivial variety of monoids $\mathbf{V}$, for any alphabet Σ and any $x, y \in \Sigma^*$, we have that $x\Sigma^*y \in \mathcal{P}(\mathbf{V})$ by closure of $\mathcal{P}(\mathbf{V})$ under Boolean operations, Lemma 2.1. Put informally, p-recognition by monoids taken from any fixed non-trivial variety of monoids allows one to check some constant-length beginning or ending of the input words. Moreover, p-recognition by monoids taken from any fixed non-trivial variety of monoids $\mathbf{V}$ also easily allows to test for membership of words in $\mathcal{L}(\mathbf{V})$ after stripping out some constant-length beginning or ending: that is, languages of the form $\Sigma^{k_1}L\Sigma^{k_2}$ for $k_1, k_2 \in \mathbb{N}$ and $L \in \mathcal{L}(\mathbf{V})$.

This motivates the definition of *essentially*-$\mathbf{V}$ stamps.

Definition 3.7. Let $\mathbf{V}$ be a variety of monoids. Let $\varphi\colon \Sigma^* \to M$ be a stamp and let s be its stability index.

We say that φ is *essentially*-$\mathbf{V}$ whenever there exists a stamp $\mu\colon \Sigma^* \to N$ with $N \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu(u) = \mu(v) \Rightarrow \big(\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s\big)\ .$$

We will denote by $\mathbf{EV}$ the class of all essentially-$\mathbf{V}$ stamps[3] and by $\mathcal{L}(\mathbf{EV})$ the class of languages recognized by morphisms in $\mathbf{EV}$.

Informally stated, a stamp $\varphi\colon \Sigma^* \to M$ is essentially-$\mathbf{V}$ when it behaves like a stamp into a monoid of $\mathbf{V}$ as soon as a sufficiently long beginning and ending of any input word has been fixed. The value for "sufficiently long" depends on φ and is adequately given by the stability index s of φ, as by definition of s, any word w of length at least $2s$ can always be made of length between s and $2s-1$ without changing the image by φ.

Let us start by giving some examples.

Consider first the language $a(a+b)^*$ over the alphabet $\{a, b\}$. Let's take $\varphi\colon \{a, b\}^* \to M$ to be its syntactic morphism: its stability index is equal to 1 and it has the property that for any $w \in \{a, b\}^*$, we have $\varphi(aw) = \varphi(a)$ and $\varphi(bw) = \varphi(b)$. Hence, if we define $\mu\colon \{a, b\}^* \to \{1\}$ to be the obvious stamp into the trivial monoid $\{1\}$, we indeed have that for all $u, v \in \{a, b\}^*$, it holds that

$$\mu(u) = \mu(v) \Rightarrow \big(\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \{a, b\}^1\big)\ .$$

In conclusion, the stamp φ is essentially-$\mathbf{V}$ for any variety of monoids, in particular $a(a+b)^* \in \mathcal{L}(\mathbf{EI})$.

Let us now consider the language $a(a+b)^*b(a+b)^*a$ over the alphabet $\{a, b\}$ of words starting and ending with an a and containing some b in between. Let $\varphi'\colon \{a, b\}^* \to M'$ be its syntactic morphism: its stability index is equal to 3 and it has the property that for all $x, y \in \{a, b\}^+$, given any $u, v \in \{a, b\}^*$ verifying that the letter b appears in u if and only if it appears in v, it holds that $\varphi'(xuy) = \varphi'(xvy)$. Hence, if we define $\mu'\colon \{a, b\}^* \to N'$ to be the syntactic morphism of the language $(a+b)^*b(a+b)^*$, it is direct to see that for all $u, v \in \{a, b\}^*$, it holds that

$$\mu'(u) = \mu'(v) \Rightarrow \big(\varphi'(xuy) = \varphi'(xvy) \quad \forall x, y \in \{a, b\}^3\big)\ .$$

[3]This class actually is an *ne*-variety of stamps, as defined in [Str02].

So we can conclude that the stamp φ' is essentially-$\mathbf{V}$ for any variety of monoids containing the syntactic monoid of $(a+b)^*b(a+b)^*$, in particular $a(a+b)^*b(a+b)^*a \in \mathcal{L}(\mathbf{EJ})$. However, note that $\varphi' \notin \mathbf{EI}$ because we have $\varphi'\big((aaa)a(aaa)\big) \neq \varphi'\big((aaa)b(aaa)\big)$.

It is now easy to prove that as long as $\mathbf{V}$ is non-trivial, polynomial-length programs over monoids from $\mathbf{V}$ do have the built-in capacity to recognize any language recognized by an essentially-$\mathbf{V}$ stamp.

Proposition 3.8. *For any non-trivial variety of monoids $\mathbf{V}$, we have $\mathcal{L}(\mathbf{EV}) \subseteq \mathcal{P}(\mathbf{V})$.*

Proof. Let $\varphi\colon \Sigma^* \to M$ be a stamp in $\mathbf{EV}$. By definition, given the stability index s of φ, there exists a stamp $\mu\colon \Sigma^* \to N$ with $N \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu(u) = \mu(v) \Rightarrow \big(\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s\big) \ .$$

Let $F \subseteq M$. By definition of μ, given $m \in N$ and $x, y \in \Sigma^s$, we either have that $x\mu^{-1}(m)y \subseteq \varphi^{-1}(F)$ or that $x\mu^{-1}(m)y \cap \varphi^{-1}(F) = \emptyset$. This entails that there exist $B \subseteq \Sigma^{\leq 2s-1}$ and $I \subseteq \Sigma^s \times N \times \Sigma^s$ such that

$$\varphi^{-1}(F) = B \cup \bigcup_{(x,m,y)\in I} x\mu^{-1}(m)y \ .$$

We claim that $\{w\} \in \mathcal{P}(\mathbf{V})$ for any $w \in \Sigma^{\leq 2s-1}$ and also that $x\mu^{-1}(m)y \in \mathcal{P}(\mathbf{V})$ for any $x, y \in \Sigma^s$ and $m \in N$. So, by closure of $\mathcal{P}(\mathbf{V})$ under Boolean operations, Lemma 2.1, it follows that $\varphi^{-1}(F) \in \mathcal{P}(\mathbf{V})$. Since this is true for any F, we have that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{V})$ and as this is itself true for all φ, we can conclude that $\mathcal{L}(\mathbf{EV}) \subseteq \mathcal{P}(\mathbf{V})$.

The claim remains to be proven.

Let $k \in \mathbb{N}_{>0}$ and $a \in \Sigma$. Since $\mathbf{V}$ is non-trivial, there exists a non-trivial $N' \in \mathbf{V}$: we shall denote its identity by 1 and by z one of its elements distinct from the identity, chosen arbitrarily. It is easy to see that the language $\Sigma^{k-1}a\Sigma^*$ is p-recognized by the sequence of programs $(P_n)_{n\in\mathbb{N}}$ over N' such that for all $n \in \mathbb{N}$, we have

$$P_n = \begin{cases} (k, f) & \text{if } n \geq k \\ \varepsilon & \text{otherwise} \end{cases}$$

where $f\colon \Sigma \to N'$ is defined by $f(b) = \begin{cases} z & \text{if } b = a \\ 1 & \text{otherwise} \end{cases}$ for all $b \in \Sigma$. We prove the same for $\Sigma^*a\Sigma^{k-1}$ symmetrically.

It then follows by closure of $\mathcal{P}(\mathbf{V})$ under Boolean operations, Lemma 2.1, that $\{w\} \in \mathcal{P}(\mathbf{V})$ for any $w \in \Sigma^{\leq 2s-1}$ and that $x\Sigma^*y \in \mathcal{P}(\mathbf{V})$ for any $x, y \in \Sigma^s$.

Finally, let $m \in N$. It is direct to show that there exists $L_m \subseteq \Sigma^*$ in $\mathcal{P}(\mathbf{V})$ verifying that $L_m \cap \Sigma^s\Sigma^*\Sigma^s = \Sigma^s\mu^{-1}(m)\Sigma^s$: just build the sequence of programs $(Q_n)_{n\in\mathbb{N}}$ over N such that for all $n \in \mathbb{N}$, we have

$$Q_n = \begin{cases} (s+1, g)(s+2, g)\cdots(n-s, g) & \text{if } n \geq 2s+1 \\ \varepsilon & \text{otherwise} \end{cases}$$

where $g\colon \Sigma \to N$ is defined by $g(b) = \mu(b)$ for all $b \in \Sigma$. We can then conclude that $x\mu^{-1}(m)y \in \mathcal{P}(\mathbf{V})$ for any $x, y \in \Sigma^s$ by closure of $\mathcal{P}(\mathbf{V})$ under Boolean operations, Lemma 2.1, and this holds for any m. $\square$

3.3. **Tameness.** We are now ready to define tameness.

We will say that a stamp $\varphi\colon \Sigma^* \to M$ is *stable* whenever $\varphi(\Sigma^2) = \varphi(\Sigma)$, i.e. the stability index of φ is 1.

Definition 3.9. A variety of monoids $\mathbf{V}$ is said to be *tame* whenever for any stable stamp $\varphi\colon \Sigma^* \to M$, if $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{V})$ then $\varphi \in \mathbf{EV}$.

Let us first mention that tameness is a generalization of *sp*-varieties of monoids.

Proposition 3.10. *Any* sp*-variety of monoids is tame.*

Proof. Let $\mathbf{V}$ be an *sp*-variety of monoids.

Let $\varphi\colon \Sigma^* \to M$ be a stable stamp such that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{V})$.

Let $S = \varphi(\Sigma^+)$: as φ is stable, we have $S = \varphi(\Sigma)$. Let $\rho\colon S \to \Sigma$ be an arbitrary mapping from S to Σ such that $\varphi(\rho(s)) = s$. Consider $\eta_S\colon S^* \to S^1$ the evaluation morphism of S: the unique morphism $f\colon S^* \to \Sigma^*$ sending each letter $s \in S$ to $\rho(s)$ verifies that $\eta_S = \varphi \circ f$. Now, given any $F \subseteq S^1$, we have $\eta_S^{-1}(F) = f^{-1}(\varphi^{-1}(F))$, but since $\varphi^{-1}(F) \in \mathcal{P}(\mathbf{V})$ and as f is an *lm*-morphism because it sends each letter of S to a letter of Σ, it follows that $\eta_S^{-1}(F) \in \mathcal{P}(\mathbf{V})$ by closure of $\mathcal{P}(\mathbf{V})$ under inverses of *lm*-morphisms, Lemma 2.1. Therefore, $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$.

Since $\mathbf{V}$ is an *sp*-variety of monoids, this entails that $M = S^1$ belongs to $\mathbf{V}$, and therefore $\varphi \in \mathbf{EV}$. As this is true for any stable stamp φ such that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{V})$, we can conclude that $\mathbf{V}$ is tame. □

The notion of essentially-$\mathbf{V}$ stamps can be adapted to varieties of semigroups in a straightforward way. We can then define a notion of tameness for varieties of semigroups accordingly. The exact same proof as the one above then goes through to allow us to show that *p*-varieties of the form $\mathbf{V} * \mathbf{D}$ are tame.

However there exist varieties of monoids that are tame but not *sp*-varieties. We give an example of such a variety in Subsection 3.4.

Programs over monoids taken from tame varieties of monoids have the expected behavior as we show next.

Let $\varphi\colon \Sigma^* \to M$ be a stamp of stability index s. The *stable stamp of* φ is the unique stamp $\varphi'\colon (\Sigma^s)^* \to M'$ such that $\varphi'(u) = \varphi(u)$ for all $u \in \Sigma^s$ and M' is the stable monoid of φ. For any variety of monoids $\mathbf{V}$ we let $\mathbf{QEV}$ be the class of stamps whose stable stamp is essentially-$\mathbf{V}$ and, accordingly, we define $\mathcal{L}(\mathbf{QEV})$ as the class of languages whose syntactic morphism is in $\mathbf{QEV}$.

Proposition 3.11. *A variety of monoids* $\mathbf{V}$ *is tame if and only if* $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QEV})$.

Proof. Let $\mathbf{V}$ be a variety of monoids.

Left-to-right implication. Assume first that $\mathbf{V}$ is tame. For this direction, the proof follows the same lines as those of Proposition 3.4.

Let $L \in \mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg}$ over some alphabet Σ and let $\eta\colon \Sigma^* \to M$ be the syntactic morphism of L. For any $m \in M$, a classical algebraic argument [Pin86, Chapter 2, proof of Lemma 2.6] shows that $\eta^{-1}(m)$ is a Boolean combination of quotients of L, so $\eta^{-1}(m) \in \mathcal{P}(\mathbf{V})$ by Lemma 2.1.

Now let s be the stability index of η, let M' be its stable monoid and take $\eta'\colon (\Sigma^s)^* \to M'$ to be the stable stamp of η. The unique morphism $f\colon (\Sigma^s)^* \to \Sigma^*$ such that $f(u) = u$ for all $u \in \Sigma^s$ is an *lm*-morphism and verifies that $\eta' = \eta \circ f$. Hence, for all $m' \in M'$, we have that

$\eta'^{-1}(m') = f^{-1}(\eta^{-1}(m'))$, so that $\eta'^{-1}(m') \in \mathcal{P}(\mathbf{V})$ by closure of $\mathcal{P}(\mathbf{V})$ under inverses of lm-morphisms, Lemma 2.1. Thus, since inverses of monoid morphisms commute with union and $\mathcal{P}(\mathbf{V})$ is closed under unions (Lemma 2.1), we can conclude that $\eta'^{-1}(F) \in \mathcal{P}(\mathbf{V})$ for all $F \subseteq M'$, i.e. $\mathcal{W}(\eta') \subseteq \mathcal{P}(\mathbf{V})$.

But as η' is stable, by tameness of $\mathbf{V}$, this entails that $\eta' \in \mathbf{EV}$, so that $L \in \mathcal{L}(\mathbf{QEV})$.

Right-to-left implication. Assume now that $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QEV})$. Let $\varphi \colon \Sigma^* \to M$ be a stable stamp verifying $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{V})$.

For any $m \in M$, we therefore have $\varphi^{-1}(m) \in \mathcal{L}(\mathbf{QEV})$. Let $\eta_m \colon \Sigma^* \to M_m$ be the syntactic morphism of the language $\varphi^{-1}(m)$, we thus have $\eta_m \in \mathbf{QEV}$. We first claim that η_m is a stable stamp. To see this notice first that for all $u, v \in \Sigma^*$ we have $\varphi(u) = \varphi(v) \Rightarrow \eta_m(u) = \eta_m(v)$. Indeed assume that $\varphi(u) = \varphi(v)$, then for all $x, y \in \Sigma^*$ we have $\varphi(xuy) = \varphi(xvy)$ hence we have $xuy \in \varphi^{-1}(m)$ iff $xvy \in \varphi^{-1}(m)$ which entails $\eta_m(u) = \eta_m(v)$ by definition of the syntactic morphism. It follows that $\eta_m(\Sigma^2) = \eta_m(\Sigma)$ as $\varphi(\Sigma^2) = \varphi(\Sigma)$.

Since η_m is equal to its stable stamp and $\eta_m \in \mathbf{QEV}$, it follows that $\eta_m \in \mathbf{EV}$. Therefore there exists a stamp $\mu_m \colon \Sigma^* \to N_m$ with $N_m \in \mathbf{V}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu_m(u) = \mu_m(v) \Rightarrow \big(\eta_m(xuy) = \eta_m(xvy) \quad \forall x, y \in \Sigma\big) \ .$$

Now, we define the unique stamp $\mu \colon \Sigma^* \to N$ such that $\mu(a) = \prod_{m \in M} \mu_m(a)$ for all $a \in \Sigma$ and N is the submonoid of $\prod_{m \in M} N_m$ generated by the set $\{\prod_{m \in M} \mu_m(a) \mid a \in \Sigma\}$. As $\mathbf{V}$ is a variety, $N \in \mathbf{V}$. Take $u, v \in \Sigma^*$ and assume $\mu(u) = \mu(v)$: this means that $\mu_m(u) = \mu_m(v)$ for all $m \in M$. Let $x, y \in \Sigma$. We then have in particular $\mu_{\varphi(xuy)}(u) = \mu_{\varphi(xuy)}(v)$. This implies by definition of $\mu_{\varphi(xuy)}$ that $\eta_{\varphi(xuy)}(xuy) = \eta_{\varphi(xuy)}(xvy)$. As $\eta_{\varphi(xuy)}$ is the syntactic morphism of $\varphi^{-1}(\varphi(xuy))$, it follows that $\varphi(xvy) = \varphi(xuy)$. And this is true for any $x, y \in \Sigma$.

In conclusion, μ witnesses the fact that φ is essentially-$\mathbf{V}$. $\square$

As for the case of *sp*-varieties of monoids, we don't know whether it is always true that for a tame non-trivial variety of monoids $\mathbf{V}$, $\mathcal{L}(\mathbf{QEV})$ is included in $\mathcal{P}(\mathbf{V})$. If this were the case then for tame non-trivial varieties of monoids $\mathbf{V}$ we would have $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QEV})$. We conjecture this to be at least true for varieties of monoids that are local.

Conjecture 3.12. Let $\mathbf{V}$ be a local tame variety of monoids. Then $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QEV})$.

We conclude this subsection by showing that $\mathbf{J}$, which is not an *sp*-variety of monoids (Proposition 3.6), isn't tame either.

Proposition 3.13. $\mathbf{J}$ *is not tame.*

Proof. To show this, we show that $(a + b)^* ac^+$, which belongs to $\mathcal{P}(\mathbf{J})$ by the construction of the introduction, does not belong to $\mathcal{L}(\mathbf{QEJ})$.

We first claim that any essentially-$\mathbf{J}$ stamp $\varphi \colon \Sigma^* \to M$ of stability index s verifies that there exists some $k \in \mathbb{N}_{>0}$ such that $\varphi(x(uv)^k y) = \varphi(x(uv)^k uy)$ for all $u, v \in \Sigma^*$ and $x, y \in \Sigma^s$. Indeed, by definition there exists a stamp $\mu \colon \Sigma^* \to N$ with $N \in \mathbf{J}$ such that for all $u, v \in \Sigma^*$, we have

$$\mu(u) = \mu(v) \Rightarrow \big(\varphi(xuy) = \varphi(xvy) \quad \forall x, y \in \Sigma^s\big) \ .$$

If we set ω to be the idempotent power of N, we have that for all $u, v \in \Sigma^*$,

$$\mu\big((uv)^\omega\big) = \big(\mu(u)\mu(v)\big)^\omega = \big(\mu(u)\mu(v)\big)^\omega \mu(u) = \mu\big((uv)^\omega u\big)$$

by the identities for **J**. Hence, we have that $\varphi(x(uv)^\omega y) = \varphi(x(uv)^\omega uy)$ for all $u, v \in \Sigma^*$ and $x, y \in \Sigma^s$.

Let us now consider the syntactic morphism $\eta\colon \{a,b,c\}^* \to M$ of the language $(a+b)^*ac^+$. As already mentioned for Proposition 3.6, the stable monoid of η is equal to the syntactic monoid M. Moreover, the stability index of η is 2. Therefore, the stable stamp of η is the unique stamp $\eta'\colon (\{a,b,c\}^2)^* \to M$ such that $\eta'(u) = \eta(u)$ for all $u \in \{a,b,c\}^2$. By what we have shown just above, since the stability index of η' is 1, if η' were essentially-**J**, there should exist some $k \in \mathbb{N}_{>0}$ such that $\eta'(x(uv)^k y) = \eta'(x(uv)^k uy)$ for all $u, v \in (\{a,b,c\}^2)^*$ and $x, y \in \{a,b,c\}^2$. However, for all $k \in \mathbb{N}_{>0}$, we do have that $(aa)\big((bb)(aa)\big)^k(cc) \in (a+b)^*ac^+$ while $(aa)\big((bb)(aa)\big)^k(bb)(cc) \notin (a+b)^*ac^+$, which implies that

$$\eta'\Big((aa)\big((bb)(aa)\big)^k(cc)\Big) \neq \eta'\Big((aa)\big((bb)(aa)\big)^k(bb)(cc)\Big)$$

for all $k \in \mathbb{N}_{>0}$. Therefore, it follows that the stable stamp η' of η is not essentially-**J**, so we can conclude that $(a+b)^*ac^+ \notin \mathcal{L}(\mathbf{QEJ})$. □

3.4. **The example of finite commutative monoids.** The variety **Com** of finite commutative monoids is defined by the identity $xy = yx$ and $\mathcal{L}(\mathbf{Com})$ is the class of languages that are Boolean combinations of languages of the form $\{w \in \Sigma^* \mid |w|_a \equiv k \mod p\}$ for $k \in [\![0, p-1]\!]$ and p prime or $\{w \in \Sigma^* \mid |w|_a = k\}$ for $k \in \mathbb{N}$ with Σ any alphabet and $a \in \Sigma$ (see [Eil76, Chapter VIII, Example 3.5]).

Since the syntactic monoid of the language $a(a+b)^*$ is not commutative, by the discussion at the end of Subsection 3.1, we know that **Com** is not an *sp*-variety of monoids. It is, however, tame, as we are going to prove now.

We first give a sufficient equational characterization for any stable stamp φ to be essentially-**Com**.

Lemma 3.14. *Let $\varphi\colon \Sigma^* \to M$ be a stable stamp verifying that for any $x, y, e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents, we have*

$$\varphi(exyf) = \varphi(eyxf)\ .$$

Then, $\varphi \in \mathbf{ECom}$.

Proof. Let us define the equivalence relation $\sim$ on Σ^* by $u \sim v$ for $u, v \in \Sigma^*$ whenever $\varphi(xuy) = \varphi(xvy)$ for all $x, y \in \Sigma$. This equivalence relation is actually a congruence, because given $u, v \in \Sigma^*$ verifying $u \sim v$, for all $s, t \in \Sigma^*$ we have $sut \sim svt$ since for any $x, y \in \Sigma$, it holds that

$$\varphi(xsuty) = \varphi(x'uy') = \varphi(x'vy') = \varphi(xsvty)$$

where $x', y' \in \Sigma$ verify $\varphi(xs) = \varphi(x')$ and $\varphi(ty) = \varphi(y')$.

We observe that, since the stability index of φ is equal to 1, φ verifies that $\varphi(euvf) = \varphi(evuf)$ for all $u, v \in \Sigma^*$ and $e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents. Now take $u, v \in \Sigma^*$ and $x, y \in \Sigma$. Since $\varphi(\Sigma)$ is a finite semigroup and verifies that $\varphi(\Sigma) = \varphi(\Sigma)^2$, by a classical result in finite semigroup theory (see e.g. [Pin86, Chapter 1, Proposition 1.12]), we have that there exist $x_1, e, x_2, y_1, f, y_2 \in \Sigma$ such that $\varphi(x_1 e x_2) = \varphi(x)$ and $\varphi(y_1 f y_2) = \varphi(y)$

with $\varphi(e)$ and $\varphi(f)$ idempotents. Therefore, it follows that

$$\begin{aligned}\varphi(xuvy) &= \varphi(x_1ex_2uvy_1fy_2)\\ &= \varphi(x_1euvy_1x_2fy_2)\\ &= \varphi(x_1euvy_1x_2ffy_2)\\ &= \varphi(x_1ey_1x_2fuvfy_2)\\ &= \varphi(x_1ey_1x_2fvufy_2)\\ &= \varphi(x_1evuy_1x_2ffy_2)\\ &= \varphi(x_1evuy_1x_2fy_2)\\ &= \varphi(x_1ex_2vuy_1fy_2)\\ &= \varphi(xvuy)\ .\end{aligned}$$

Thus, we have that $uv \sim vu$ for all $u, v \in \Sigma^*$, implying that $\Sigma^*/{\sim} \in \mathbf{Com}$. We can eventually conclude that the stamp $\mu\colon \Sigma^* \to \Sigma^*/{\sim}$ defined by $\mu(w) = [w]_\sim$ for all $w \in \Sigma^*$ witnesses, by construction, the fact that φ is essentially-**Com**. □

The following lemma then asserts that any stable stamp φ such that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{Com})$ actually verifies the equation of the previous lemma, which allows us to conclude that **Com** is tame by combining those two lemmas.

Lemma 3.15. *Let $\varphi\colon \Sigma^* \to M$ be a stable stamp such that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{Com})$. Then, for any $x, y, e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents, we have*

$$\varphi(exyf) = \varphi(eyxf)\ .$$

Proof. Let us first observe that for any program P over some finite commutative monoid N using the input alphabet Σ and of range $n \in \mathbb{N}$, there exist a program P' over N using the same input alphabet and of same range such that $P' = \prod_{i=1}^{n}(i, h_i)$ verifying $P(w) = P'(w)$ for all $w \in \Sigma^n$ [Tes03, Example 3.4]. We call P' a single-scan program.

The assumption that $\mathcal{W}(\varphi) \subseteq \mathcal{P}(\mathbf{Com})$ thus means that for all $F \subseteq M$, there exists a sequence $(P_{F,n})_{n\in\mathbb{N}}$ of single-scan programs over some $N_F \in \mathbf{Com}$ that recognizes $\varphi^{-1}(F)$.

For all $x, y, e, f, g \in \Sigma$ such that $\varphi(e)$, $\varphi(f)$ and $\varphi(g)$ are idempotents, we claim that

$$\varphi(exfyg) = \varphi(eyfxg) \tag{3.1}$$
$$\varphi(efef) = \varphi(ef) \tag{3.2}$$
$$\varphi(exyf) = \varphi(eyefxf). \tag{3.3}$$

Assuming the claim, take $x, y, e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents. Equation (3.2) implies that $\varphi(ef)$ is an idempotent. As $\varphi(\Sigma^2) = \varphi(\Sigma)$, we have that there exists $g \in \Sigma$ such that $\varphi(ef) = \varphi(g)$, hence

$$\varphi(exyf) = \varphi(eyefxf) = \varphi(eygxf) = \varphi(exgyf) = \varphi(exefyf) = \varphi(eyxf)\ ,$$

the first and last equalities being from (3.3) and the middle one from (3.1).

Thus, the lemma will be proven once we will have proven that (3.1), (3.2) and (3.3) hold for all $x, y, e, f, g \in \Sigma$ such that $\varphi(e)$, $\varphi(f)$ and $\varphi(g)$ are idempotents.

(3.1) **holds.** Let $x, y, e, f, g \in \Sigma$ such that $\varphi(e)$, $\varphi(f)$ and $\varphi(g)$ are idempotents. Set $F = \{\varphi(exfyg)\}$ and $n = 2(|N_F|^2 + 1) + 1$ and assume $P_{F,n} = \prod_{i=1}^{n}(i, h_i)$. Then, since the function

$$\begin{aligned} \Delta\colon [\![1, |N_F|^2 + 1]\!] &\to {N_F}^2 \\ j &\mapsto (h_{2j}(x), h_{2j}(y)) \end{aligned}$$

cannot be injective, there must exist $j_1, j_2 \in [\![1, |N_F|^2+1]\!], j_1 < j_2$ such that $h_{2j_1}(x) = h_{2j_2}(x)$ and $h_{2j_1}(y) = h_{2j_2}(y)$. So

$$P_{F,n}(e^{2j_1-1}xf^{2j_2-1-2j_1}yg^{n-2j_2}) = P_{F,n}(e^{2j_1-1}yf^{2j_2-1-2j_1}xg^{n-2j_2}) \ ,$$

hence as $e^{2j_1-1}xf^{2j_2-1-2j_1}yg^{n-2j_2} \in \varphi^{-1}(F)$ because

$$\varphi(e^{2j_1-1}xf^{2j_2-1-2j_1}yg^{n-2j_2}) = \varphi(exfyg) \ ,$$

we must have $e^{2j_1-1}yf^{2j_2-1-2j_1}xg^{n-2j_2} \in \varphi^{-1}(F)$. Thus, we have

$$\varphi(e^{2j_1-1}yf^{2j_2-1-2j_1}xg^{n-2j_2}) = \varphi(eyfxg) = \varphi(exfyg) \ .$$

So for all $x, y, e, f, g \in \Sigma$ such that $\varphi(e)$, $\varphi(f)$ and $\varphi(g)$ are idempotents, we have that (3.1) holds.

(3.2) **holds.** Let $e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents. We have that

$$\varphi(efef) = \varphi(effef) = \varphi(eefff) = \varphi(ef) \ ,$$

the middle equality being from (3.1).

So for all $e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents, we have that (3.2) holds.

(3.3) **holds.** Let $x, y, e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents. Set $F = \{\varphi(exyf)\}$ and $n = 4(2\,|N_F|^4 + 1)$ and assume $P_{F,n} = \prod_{i=1}^{n}(i, h_i)$. Then, we have that the function

$$\begin{aligned} \Delta\colon [\![1, 2\,|N_F|^4 + 1]\!] &\to {N_F}^4 \\ j &\mapsto (h_{4j-2}(x), h_{4j-2}(f), h_{4j-1}(y), h_{4j-1}(e)) \end{aligned}$$

verifies that there exists $(m_1, m_2, m_3, m_4) \in {N_F}^4$ such that

$$\left|\Delta^{-1}((m_1, m_2, m_3, m_4))\right| \geq 3 \ .$$

Therefore, there exist $j_1, j_2, j_3 \in [\![1, 2\,|N_F|^4 + 1]\!], j_1 < j_2 < j_3$ such that $h_{4j_3-2}(x) = h_{4j_2-2}(x)$, $h_{4j_3-2}(f) = h_{4j_2-2}(f)$, $h_{4j_2-1}(y) = h_{4j_1-1}(y)$ and $h_{4j_2-1}(e) = h_{4j_1-1}(e)$. So

$$P_{F,n}(e^{4j_2-3}xyf^{n-4j_2+1}) = P_{F,n}(e^{4j_1-2}ye^{4(j_2-j_1)-2}fef^{4(j_3-j_2)-2}xf^{n-4j_3+2}) \ ,$$

hence as $e^{4j_2-3}xyf^{n-4j_2+1} \in \varphi^{-1}(F)$ because

$$\varphi(e^{4j_2-3}xyf^{n-4j_2+1}) = \varphi(exyf) \ ,$$

we must have $e^{4j_1-2}ye^{4(j_2-j_1)-2}fef^{4(j_3-j_2)-2}xf^{n-4j_3+2} \in \varphi^{-1}(F)$. Thus, we have

$$\begin{aligned} \varphi(exyf) &= \varphi(e^{4j_1-2}ye^{4(j_2-j_1)-2}fef^{4(j_3-j_2)-2}xf^{n-4j_3+2}) \\ &= \varphi(eyefefxf) \\ &= \varphi(eyefxf) \end{aligned}$$

where the last equality uses (3.2).

So for all $x, y, e, f \in \Sigma$ such that $\varphi(e)$ and $\varphi(f)$ are idempotents, we have that (3.3) holds. $\square$

4. The case of DA

In this section, we prove that **DA** is an *sp*-variety of monoids, which implies that it is tame. Combined with the fact that **DA** is local [Alm96], we obtain the following result by Proposition 3.5.

Theorem 4.1. $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA})$.

The result follows from the following main technical contribution:

Proposition 4.2. *$(c+ab)^*$, $(b+ab)^*$ and $b^*((ab^*)^k)^*$ for any integer $k \geq 2$ are regular languages not in $\mathcal{P}(\mathbf{DA})$.*

Observe that for any $k \in \mathbb{N}, k \geq 2$, the fact that $b^*((ab^*)^k)^* \notin \mathcal{P}(\mathbf{DA})$ is an immediate corollary of the classical result that $\mathrm{MOD_k} \notin \mathsf{AC}^0 = \mathcal{P}(\mathbf{A})$ [FSS84, Ajt83]. However, we propose a direct semigroup-theoretic proof of the first result without resorting to the involved proof techniques of the latter result.

Before proving the proposition we first show that it implies that **DA** is an *sp*-variety of monoids. This implication is a consequence of the following lemma, which is a result inspired by an observation in [TT02b] stating that non-membership of a given finite monoid M in **DA** implies non-aperiodicity of M or division of it by (at least) one of two specific finite monoids.

Lemma 4.3. *Let S be a finite semigroup such that $S^1 \notin \mathbf{DA}$. Then, one of $(c+ab)^*$, $(b+ab)^*$ or $b^*((ab^*)^k)^*$ for some $k \in \mathbb{N}, k \geq 2$ is recognized by a morphism $\mu\colon \Sigma^* \to S^1$, for Σ the appropriate alphabet, such that $\mu(\Sigma^+) \subseteq S$.*

Proof. We distinguish two cases: the aperiodic and the non-aperiodic one.

Aperiodic case. Assume first that S^1 is aperiodic. Then, since $S^1 \notin \mathbf{DA}$, by Lemma 3.2.4 in [Tes98], we have that S^1 is divided by the syntactic monoid of $(c^*ac^*bc^*)^*$, denoted by B_2, or by the syntactic monoid of $\big((b+c)^*a(b+c)^*b(b+c)^*\big)^*$, denoted by U. We treat those two not necessarily distinct subcases separately.

Subcase B_2 divides S^1. It is easily proven that $(c+ab)^*$ is recognized by B_2 (actually, its syntactic monoid is isomorphic to B_2): just consider the syntactic morphism $\eta\colon \{a,b,c\}^* \to B_2$ of $(c^*ac^*bc^*)^*$ and build the morphism $\varphi\colon \{a,b,c\}^* \to B_2$ sending a to $\eta(a)$, b to $\eta(b)$ and c to $\eta(ab)$.

By a classical result in algebraic automata theory [Pin86, Chapter 1, Proposition 2.7], this implies that S^1 recognizes $(c+ab)^*$. Thus there exists a morphism $\mu\colon \{a,b,c\}^* \to S^1$ recognizing $(c+ab)^*$, that has the property that $\mu(\{a,b,c\}^+) \subseteq S$, otherwise there would exist $w \in \{a,b,c\}^+$ such that either $w \notin (c+ab)^*$ while $\mu(\varepsilon) = \mu(w)$ or $w \in (c+ab)^+$ while $\mu(ab) = \mu(awb)$.

Subcase U divides S^1. The proof goes the same way as for the first subcase.

It is again easily proven that $(b+ab)^*$ is recognized by U: here we consider the syntactic morphism $\eta\colon \{a,b\}^* \to U$ of $\big((b+c)^*a(b+c)^*b(b+c)^*\big)^*$ and build the morphism $\varphi\colon \{a,b\}^* \to U$ sending a to $\eta(a)$ and b to $\eta(b)$.

Thus there exists a morphism $\mu\colon \{a,b\}^* \to S^1$ recognizing $(b+ab)^*$, that also verifies $\mu(\{a,b\}^+) \subseteq S$, otherwise there would exist $w \in \{a,b\}^+$ such that $w \notin (b+ab)^*$ while $\mu(\varepsilon) = \mu(w)$ or $w \in b(b+ab)^*$ while $\mu(a) = \mu(aw)$ or $w \in ab(b+ab)^*$ while $\mu(ab) = \mu(awb)$.

Non-aperiodic case. Assume now that S^1 is not aperiodic. Then there is an x in S such that $x^\omega \neq x^{\omega+1}$ for $\omega \in \mathbb{N}_{>0}$ the idempotent power of S^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 $L = \mu^{-1}(x^\omega)$. Let $k \in \mathbb{N}, k \geq 2$ be the smallest positive integer such that $x^{\omega+k} = x^\omega$, that cannot be 1 because $x^\omega \neq x^{\omega+1}$. Using this, for all $w \in \{a,b\}^*$, we have

$$\mu(w) = x^{|w|\cdot\omega+|w|_a} = x^{\omega+(|w|_a \mod k)} ,$$

where $|w|$ indicates the length of w and $|w|_a$ the number of a's it contains, so that w belongs to L if and only if $|w|_a = 0 \mod k$. Hence, L is the language of all words with a number of a's divisible by k, $b^*((ab^*)^k)^*$. In conclusion, $b^*((ab^*)^k)^*$ is recognized by μ verifying $\mu(\{a,b\}^+) \subseteq S$. □

Let now S be any 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. To show that S^1 is in $\mathbf{DA}$, we assume for the sake of contradiction that it is not the case. Then Lemma 4.3 tells us that one of $(c+ab)^*$, $(b+ab)^*$ or $b^*((ab^*)^k)^*$ for some $k \in \mathbb{N}, k \geq 2$ is recognized by a morphism $\mu\colon \Sigma^* \to S^1$, for Σ the appropriate alphabet, such that $\mu(\Sigma^+) \subseteq S$.

In all cases, we thus have a language $L \subseteq \Sigma^*$ equal to $\mu^{-1}(Q)$ for some subset Q of S^1 with the morphism μ sending letters of Σ to elements of S. Consider then the morphism $\varphi\colon \Sigma^* \to S^*$ sending each letter $a \in \Sigma$ to $\mu(a)$, a letter of S: we have $\mu = \eta_S \circ \varphi$, so that $L = \varphi^{-1}(\eta_S^{-1}(Q))$. As $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{DA})$, we have that $\eta_S^{-1}(Q) \in \mathcal{P}(\mathbf{DA})$, hence since φ is an *lm*-morphism and $\mathcal{P}(\mathbf{DA})$ is closed under inverses of *lm*-morphisms by Lemma 2.1, we have $L = \varphi^{-1}(\eta_S^{-1}(Q)) \in \mathcal{P}(\mathbf{DA})$: a contradiction to Proposition 4.2.

In the remaining part of this section we prove Proposition 4.2.

Proof of Proposition 4.2. 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 L. 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 set Δ of words such that $L \subseteq \Delta^*$ (for instance take $\Delta = \{c, ab\}$ for $L = (c+ab)^*$). 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 outputs the same value and there are completions of the entries in Δ^*. Hence, if Δ was chosen so that there is actually a completion of the fixed entries in L and one outside of L, P_n cannot recognize the restriction of L to words of length n. 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 spell out the technical details.

Let Δ be a finite non-empty set of non-empty words over an alphabet Σ. 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 (possibly zero) 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* and is necessarily free.

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.

Note that it is important in the definition of safe for Δ that we fix only safe positions, i.e. positions far apart and far from the beginning and the end of the mask. Indeed, depending on the chosen Δ, there might be words that never appear as factors in any word of Δ^*, such as bb when $\Delta = \{c, ab\}$ or aa when $\Delta = \{b, ab\}$, so that fixing a position near an already fixed position to an arbitrary letter in a Δ-compatible mask may result in a mask that has no completion in Δ^*. This is why we make sure that safe positions are far from those already fixed and from the beginning and the end of the mask, where far depends on the length of the words of Δ.

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.

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

We define a version of Green's relations for decomposing monoids that will be used, as often in this setting, to prove the main technical lemma in the current proof. 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 $u' \not\leq_J u$. 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 $u' \not\leq_R u$. 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 $u' \not\leq_L u$. Finally, given two elements u, u' of M, we write $u \sim_H u'$ if $u \sim_R u'$ and $u \sim_L u'$.

We shall use the following well-known fact about these preorders and equivalence relations (see [Pin86, Chapter 3, Proposition 1.4]).

Lemma 4.4. *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 $u \sim_L v$.*

From the definition it follows that for all elements u, v, r of M, we have $u \leq_R ur$ and $v \leq_L rv$. When the inequality is strict in the first case, i.e. $u <_R ur$, we say that r is R-bad for u. Similarly r is L-bad for v if $v <_L rv$. It follows from $M \in$ **DA** that being R-bad or L-bad only depends on the $\sim_R$ or $\sim_L$ class, respectively. This is formalized in the following lemma, that is folklore and used at least implicitly in many proofs involving **DA** (see for instance [TT02b, proof of Theorem 3]). Since we didn't manage to find the lemma stated and proven in the form below, we include a proof for completeness.

Lemma 4.5. *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 $ru' \sim_L u$.*

Proof. Let $u, u', r \in M$ such that $u \sim_R u'$ and $ur \sim_R u$. This means that there exist $v, v', s \in M$ such that $u = u'v'$, $u' = uv$ and $urs = u$.

This implies that

$$u' = uv = ursv = u'v'rsv = u'(v'rsv)^2 = \cdots = u'(v'rsv)^\omega$$

where ω is the idempotent power of M. Hence, we have that

$$u'r(v'rsv)^\omega v' = u'(v'rsv)^\omega r(v'rsv)^\omega v' \ .$$

But, by [TT02b, Theorem 2], since $M \in \mathbf{DA}$, we have that $(xyz)^\omega y(xyz)^\omega = (xyz)^\omega$ for all $x, y, z \in M$, so that

$$u'r(v'rsv)^\omega v' = u'(v'rsv)^\omega v' = u'v' = u \ .$$

Therefore, we have $u'r \leq_R u$ and since $uvr = u'r$, we also have $u \leq_R u'r$, so that $u'r \sim_R u$ as claimed.

The proof goes through symmetrically for $\sim_L$. □

Let Δ be a finite set of words and Σ be the corresponding 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 of M. 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 and only 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 $u_1P_1(w)v_1 = u_2P_2(w)v_2$ 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 $u_1P_1(w)v_1 = u_2P_2(w)v_2$ 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 $u_1P_1(w)v_1 <_J u_2P_2(w)v_2$ 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 $u_1P_1(w)v_1 <_J u_2P_2(w)v_2$ 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 can be upper bounded by $2 \cdot |M|^2$, that does only depend on M. For a given quadruplet μ, we shall also call its height the biggest $i \in \mathbb{N}$ such that there exists a decreasing chain $\mu_i \prec \mu_{i-1} \prec \cdots \prec \mu_0 = \mu$.

The following lemma is the key to the proof. It shows that modulo fixing a few entries, one can fix the output: to count the number of fixed positions for a given mask λ, we denote by $|\lambda|_\Sigma$ the number of letters in λ belonging to Σ, that is to say, the number of fixed positions in λ.

Lemma 4.6. *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 such that (λ, P, u, v) is of height h. Then there is an element t of M and a Δ-compatible submask λ' of λ verifying $|\lambda'|_\Sigma \leq (2^h 6l)^{2^h} \cdot \max\{|\lambda|_\Sigma, 1\}$ such that any safe completion w of λ' verifies $uP(w)v = t$.*

Proof. The proof goes by induction on the height h.

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 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 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, and including, instruction $i-1$. Let w be a safe completion of λ. For any instruction (y, g) of P', as $y < i$, $g(w_y)$ cannot be R-bad for u, so $u \sim_R ug(w_y)$. Hence, by Lemma 4.5, $u \sim_R uP'(w)$ for all safe completions w of λ.

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 4.5, hence $uP'(w) <_J uP(w)v$ by Lemma 4.4. 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 4.5) 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, 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.

Thus, any safe completion w of λ_2 is such that

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

Therefore λ_2 and t form the desired couple of a Δ-compatible submask of λ and an element of M. We still have to show that $|\lambda_2|_\Sigma$ satisfies the desired upper bound.

By induction, since $(\lambda', P', u, 1)$ is of height $h' \leq h-1$, we have

$$|\lambda_1|_\Sigma \leq (2^{h'} 6l)^{2^{h'}} \cdot |\lambda'|_\Sigma \leq (2^{h-1} 6l)^{2^{h-1}} \cdot |\lambda'|_\Sigma \ .$$

Consequently, by induction again, as $(\lambda_1, P'', t_1 f(a), v)$ is of height $h'' \leq h-1$, we have

$$\begin{aligned} |\lambda_2|_\Sigma &\leq (2^{h''} 6l)^{2^{h''}} \cdot |\lambda_1|_\Sigma \\ &\leq (2^{h-1} 6l)^{2^{h-1}} \cdot |\lambda_1|_\Sigma \\ &\leq (2^{h-1} 6l)^{2^{h-1}} \cdot (2^{h-1} 6l)^{2^{h-1}} \cdot |\lambda'|_\Sigma \\ &= (2^{h-1} 6l)^{2^{h}} \cdot |\lambda'|_\Sigma \ . \end{aligned}$$

Moreover, it holds that $|\lambda'|_\Sigma \leq |\lambda|_\Sigma + 1$, so that

$$\begin{aligned}|\lambda_2|_\Sigma &\leq (2^{h-1}6l)^{2^h} \cdot (|\lambda|_\Sigma + 1)\\ &\leq (2^{h-1}6l)^{2^h} \cdot 2^{2^h} \cdot \max\{|\lambda|_\Sigma, 1\}\\ &= (2^h 6l)^{2^h} \cdot \max\{|\lambda|_\Sigma, 1\}\ .\end{aligned}$$

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 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 4.5, $u \sim_R uP(w)$ for all safe completions w of λ. Notice then that $u \sim_R uP(w) <_R uP(w)v$ (by Lemma 4.5), hence $uP(w) <_J uP(w)v$ (by Lemma 4.4) for all safe completions w of λ. So $(\lambda, P, u, 1) \prec (\lambda, P, u, v)$, therefore we obtain by induction a monoid element t_1 and a Δ-compatible submask λ' of λ such that $uP(w) = t_1$ for all completions w of λ'. If we set $t = t_1 v$, we get that any safe completion w of λ' is such that $uP(w)v = t_1 v = t$. Therefore λ' and t form the desired couple of a Δ-compatible submask of λ and an element of M.

Moreover, by induction, since $(\lambda, P, u, 1)$ is of height $h' \leq h - 1$, we have

$$\left|\lambda'\right|_\Sigma \leq (2^{h'}6l)^{2^{h'}} \cdot \max\{|\lambda|_\Sigma, 1\} \leq (2^h 6l)^{2^h} \cdot \max\{|\lambda|_\Sigma, 1\}\ ,$$

the desired upper bound.

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 was in Case 2 and Case 4, using Lemma 4.5, the fact that condition (a) and condition (c) are verified implies 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 4.5, we get that $uP(w)v \sim_R u$ and $uP(w)v \sim_L v$ for all safe completions w of λ. This implies that (λ, P, u, v) is minimal for $\prec$ and that $h = 0$.

Let w_0 be a completion of λ that is in Δ^*. Let λ' be the submask of λ fixing all free dangerous positions of λ using w_0 and let $t = uP(w_0)v$. Then, for any completion w of λ', which is a safe completion of λ by construction, we have that $uP(w)v \sim_R u \sim_R t$ and $uP(w)v \sim_L v \sim_L t$. Thus, $uP(w)v \sim_H t$ for any completion w of λ'. As M is aperiodic, this implies that $uP(w)v = t$ for all completions w of λ' (see [Pin86, Chapter 3, Proposition 4.2]). Therefore λ' and t form the desired couple of a Δ-compatible submask of λ and an element of M.

Now, since the number of free positions of λ fixed in λ', i.e. $|\lambda'|_\Sigma - |\lambda|_\Sigma$, is exactly the number of free dangerous positions in λ, and as a position in λ is dangerous if it is within distance $2l - 2$ of a fixed position or within distance $l - 1$ of the beginning or the end of

λ, we have

$$\left|\lambda'\right|_\Sigma \leq 2 \cdot |\lambda|_\Sigma \cdot (2l-2) + 2 \cdot (l-1) + |\lambda|_\Sigma = |\lambda|_\Sigma \cdot (4l-3) + 2l - 2$$
$$\leq (2^0 6l)^{2^0} \cdot \max\{|\lambda|_\Sigma, 1\} ,$$

the desired upper bound.

This concludes the proof of the lemma. □

Setting $\Delta = \{c, ab\}$ or $\Delta = \{b, ab\}$ with Σ the associated alphabet, when applying Lemma 4.6 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 equal to 1, 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 has a number of fixed positions upper-bounded by $(2^h 6l)^{2^h}$ where h is the height of (λ, P, u, v), itself upper-bounded by $2 \cdot |M|^2$, 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 be part of any sequence of programs p-recognizing 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 4.2 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 superfluous: any program over a monoid in **DA** is equivalent to one of polynomial length over the same monoid [TT02a] (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 $\mathrm{O}(n^{k+1})$ express strictly more than those of length $\mathrm{O}(n^k)$.

Following [GT03], 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 alphabet A. A language L is in $\mathcal{SUM}_k$ for $k \in \mathbb{N}_{>0}$ 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 stated without proof that a language L is recognized by a monoid in **DA** iff there is a $k \in \mathbb{N}$ such that L is a Boolean combination of languages in $\mathcal{SUM}_k$ [GT03] (see [Gro18, Theorem 4.1.9] for a proof). 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 $\mathcal{SUM}_k$: this is what we do in the first subsection.

In the two following subsections, we then give a fine program-length-based hierarchy within $\mathcal{P}(\mathbf{DA})$ for this parametrization of **DA**.

5.1. **A parametrization of DA.** For each $k \in \mathbb{N}$, we denote by $\mathcal{SUL}_k$ the class of regular languages that are Boolean combinations of languages in $\mathcal{SUM}_k$; it is a variety of languages as shown just below. But 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 $\mathcal{SUL}_k$.

Back to the fact that $\mathcal{SUL}_k$ is a variety of languages for any $k \in \mathbb{N}$. 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 alphabet Σ, we will denote by $\mathrm{alph}(u)$ the set of letters of Σ that appear in u.

Lemma 5.1. *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k$ over an alphabet Σ and $u \in \Sigma^*$, $u^{-1}L$ and Lu^{-1} both are unions 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 an 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 unions 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 an 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 unions 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

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

As before, 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 5.2. *For all $k \in \mathbb{N}$, for all $L \in \mathcal{SUM}_k$ over an alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism where Γ is another 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 an alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism where Γ is another 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 an alphabet Σ and $\varphi\colon \Gamma^* \to \Sigma^*$ a morphism where Γ is another 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 $B = \{b \in \Gamma \mid a \in \mathrm{alph}(\varphi(b))\}$ 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 $\varphi(b) = u_{b,1} a u_{b,2}$ with $u_{b,1}, u_{b,2} \in \Sigma^*$ and $a \notin \mathrm{alph}(u_{b,1})$. 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 5.1 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. □

5.2. **Strict hierarchy.** For each k we show there exists 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 $\mathbf{DA_k}$ but cannot be recognized by any sequence of programs of length $\mathrm{O}(n^{k-1})$ over any monoid in $\mathbf{DA}$.

For a given $k \in \mathbb{N}_{>0}$, 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* for some $n \in \mathbb{N}$ 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 for each $n \in \mathbb{N}$, 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 $\mathbf{DA_k}$ 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 for all k 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 $\alpha \cdot n^{k-1}$ for α some constant 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 some $k \in \mathbb{N}_{>0}$ and $\Delta = (S_n)_{n\in\mathbb{N}}$, 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 $\mathbf{DA_k}$ of length $\mathrm{O}(n^k)$ is a consequence of the following proposition.

Proposition 5.3. *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 k-sets S_n 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 $Y_k = \{\perp_l, \top_l \mid 1 \leq l \leq k\}$. 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 $\perp_k$ to its left, and the sequence between the first occurrence of $\top_k$ and the first occurrence of $\perp_k$ 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 elements of $Y_k \cup \{\varepsilon\}$ instead of outputting elements of M_k.

For any $k > 0$, $1 \leq j \leq n$ 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, $f_{j,S}(1) = \perp_k$ otherwise. We also let g_k be the function with $g_k(0) = \varepsilon$ and $g_k(1) = \perp_k$. If S is a k-set over n and $1 \leq j \leq n$ 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 $\perp_k$ 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 5.4. *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 $\{0,1\}^n$ through programs. □

If for some $k \in \mathbb{N}_{>0}$ and $1 \leq i \leq \alpha$, $\alpha \in \mathbb{N}_{>0}$, we apply Claim 5.4 for all $n \in \mathbb{N}$, $l = \alpha \cdot n^{k-1}$, we get a number $\mu_i(n)$ 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 $2^{\binom{n}{k}}$, i.e. $\mu_i(n)$ is in $\mathrm{o}\big(2^{\binom{n}{k}}\big)$.

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 $1 \leq i \leq j$, of range n_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 5.5. *For all $k \in \mathbb{N}$, $\mathcal{P}\big(\mathbf{DA}, n^k\big) \subsetneq \mathcal{P}\big(\mathbf{DA}, n^{k+1}\big)$. More precisely, for all $k \in \mathbb{N}$ and $d \in \mathbb{N}, d \leq \max\{k-1, 0\}$, $\mathcal{P}\big(\mathbf{DA_k}, n^d\big) \subsetneq \mathcal{P}\big(\mathbf{DA_k}, n^{d+1}\big)$.*

To prove this proposition, we use two facts. First, that for all $k \in \mathbb{N}$ and all $d \in \mathbb{N}, d \leq \max\{k-1, 0\}$, any monoid from $\mathbf{DA_d}$ is also a monoid from $\mathbf{DA_k}$. And second, that $a^* \in \mathcal{P}(\mathbf{DA_0}, n) \setminus \mathcal{P}(\mathbf{DA_0}, 1)$ simply because any program over some finite monoid of range n for $n \in \mathbb{N}$ recognizing a^n must have at least n instructions, one for each input letter.

5.3. **Collapse.** Tesson and Thérien showed that any program over a monoid M in $\mathbf{DA}$ is equivalent to one of polynomial length [TT02a]. We now show that if we further assume that M is in $\mathbf{DA_k}$ then the length can be assumed to be $\mathrm{O}(n^{\max\{k,1\}})$.

Proposition 5.6. *Let $k \geq 0$. Let $M \in \mathbf{DA_k}$. Then any program over M is equivalent to a program over M of length $\mathrm{O}(n^{\max\{k,1\}})$ which is a subprogram of the initial one.*

For each possible acceptance set, an input word to the program is accepted if and only if 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.

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 $\mathit{EP}(w)$.

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 $\mathit{EP}(w) \in L$ where L is a language in $\mathcal{SUL}_k$, a Boolean combination of languages in $\mathcal{SUM}_k$.

Lemma 5.7. *Let Σ be an alphabet, M a finite monoid, and n, k natural numbers.*

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

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

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 positive 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 subprogram of P 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.

The intuition behind the proof for a program P on inputs of length n and some $K_1 \gamma K_2 \in \mathcal{SUM}_k$ when $k \geq 2$ is as follows. We assume that K_1 does not contain any word with the letter γ, the other case is done symmetrically. Consider the subset of all indices $I_\gamma \subseteq [l]$ that correspond, for a fixed letter a and a fixed position p in the input, to the first instruction of P that would output the element γ when reading a at position p. We then have that, given some w as input, $\mathit{EP}(w) \in K_1 \gamma K_2$ if and only if there exists $i \in I_\gamma$ verifying that the element at position i of $\mathit{EP}(w)$ is γ, $\mathit{EP}[1, i-1](w) \in K_1$ and $\mathit{EP}[i+1, l](w) \in K_2$. The idea is then that if we set I to contain I_γ as well as all indices obtained by induction for $P[1, i-1]$ and K_1 and for $P[i+1, l]$ and K_2, we would have that for all w, $\mathit{EP}(w) \in K_1 \gamma K_2$ if and only if $\mathit{EP}[I](w) \in K_1 \gamma K_2$, that is $\mathit{EP}(w)$ where only the elements at indices in I have been kept.

The intuition behind the proof when $k < 2$ is essentially the same, but without induction.

We now spell out the details of the proof, starting with the inductive step.

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 and any language K over M in $\mathcal{SUM}_k$. If $K \in \mathcal{SUM}_{k-1}$, by the inductive hypothesis, we are done. Otherwise, by definition, $K = K_1\gamma K_2$ for $\gamma \in M$ and some languages $K_1 \in \mathcal{SUM}_{k_1}$ and $K_2 \in \mathcal{SUM}_{k_2}$ over M with $k_1 + k_2 = k - 1$. 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.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $1 \leq p \leq n$ and each $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \gamma$, if it exists. We let I_γ be the set of indices of these instructions for all a and p. Notice that the size of I_γ 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 $P[1, i-1]$ appearing in its subprogram obtained by induction for $P[1, i-1]$ and 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 $Q = P[I]$ 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.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

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 $J \subseteq [i-1]$ containing $J_{i,1}$ and $EP[i+1, l][J](w) \in K_2$ for any set $J \subseteq [l-i]$ 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 $EP[I'](w) \in K$. This means that 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 $EP[I'](w) = EP[1, i-1][I'_1](w)\gamma EP[i+1, l][I'_2](w)$ with $EP[1, i-1][I'_1](w) \in K_1$ and $EP[i+1, l][I'_2](w) \in K_2$. If $i \in I_\gamma$, then it means that $I'_1 \subseteq [i-1]$ contains $J_{i,1}$ and that $I'_2 \subseteq [l-i]$ contains $J_{i,2}$ by construction, so that, by induction,

$$EP(w) = EP[1, i-1](w)\gamma EP[i+1, l](w) \in K_1\gamma K_2 = K\ .$$

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. So we have $EP(w) \in K$ as desired.

Base case. There are two subcases to consider.

Subcase $k = 1$. Let n be a natural number, P a program over M of range n and length l and any language K over M in $\mathcal{SUM}_1$.

If $K \in \mathcal{SUM}_0$, we can conclude by referring to the subcase $k = 0$.

Otherwise $K = A_1^* \gamma A_2^*$ for $\gamma \in M$ and some alphabets $A_1 \subseteq M$ and $A_2 \subseteq M$. 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.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $1 \leq p \leq n$, each $\alpha \in M$ and $a \in \Sigma$ consider within the sequence of instructions of P the first and last instruction of the form (p, f) with $f(a) = \alpha$, if they exist. 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 $Q = P[I]$ has the desired properties. We just showed that it has the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

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 $i \in I \subseteq I'$. From $EP[1, i-1](w) \in A_1^*$ and $EP[i+1, l](w) \in A_2^*$ 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 $EP[I'](w) \in K$. Let i be the index in I' whose instruction provides the letter γ witnessing the fact that $EP[I'](w) \in K$. This means that $EP[I' \cap [\![1, i-1]\!]](w) \in A_1^*$ and $EP[I' \cap [\![i+1, l]\!]](w) \in A_2^*$. 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 $j' \in I \subseteq I'$ 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.

Subcase $k = 0$. Let n be a natural number, P a program over M of range n and length l and any language K over M in $\mathcal{SUM}_0$.

Then $K = A^*$ for some alphabet $A \subseteq M$.

We again use the same idea as before.

Observe that when $n = 0$, we necessarily have $P = \varepsilon$, so that the lemma is trivially proven in that case. So we now assume $n > 0$.

For each $1 \leq p \leq n$, each $\alpha \in M$ and $a \in \Sigma$ consider within the sequence of instructions of P the first instruction of the form (p, f) with $f(a) = \alpha$, if it exists. 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 $Q = P[I]$ has the desired properties. We just showed that it has the required length.

Let Q' be a subprogram of P that has Q as a subprogram: it means that there exists some set $I' \subseteq [l]$ containing I such that $Q' = P[I']$.

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 $j' \in I \subseteq I'$ 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. □

6. Conclusion

We introduced a notion of tameness, particularly relevant to the analysis of programs over monoids from "small" varieties. The main source of interest in tameness is Proposition 3.11, stating that a variety of monoids $\mathbf{V}$ is tame if and only if the class of regular languages p-recognized by programs over monoids from $\mathbf{V}$ is included in the class $\mathcal{L}(\mathbf{QEV})$. A first question that arises is for which $\mathbf{V}$ those two classes of regular languages are equal. We could not rule out the possibility that for some tame non-trivial $\mathbf{V}$, $\mathcal{L}(\mathbf{QEV}) \setminus \mathcal{P}(\mathbf{V}) \neq \emptyset$. We conjecture that if $\mathbf{V}$ is local, abusing notation, $\mathbf{QEV} = \mathbf{EV} * \mathbf{Mod}$, by analogy with $\mathbf{QV}$ equating $\mathbf{V} * \mathbf{Mod}$ in that case; as $\mathcal{L}(\mathbf{EV} * \mathbf{Mod}) \subseteq \mathcal{P}(\mathbf{V})$ holds unconditionally (because $\mathbf{V}$ cannot be trivial if it is local [Til87, p. 134]), under our conjecture $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QEV})$ would hold for local tame varieties $\mathbf{V}$ (this is Conjecture 3.12).

Concretely, we have obtained the technical result that $\mathbf{DA}$ is a tame variety using semigroup-theoretic arguments. We have given $\mathbf{A}$ and $\mathbf{Com}$ as further examples of tame varieties. Our proof that $\mathbf{A}$ is tame needed the fact that $\mathrm{MOD_m} \notin \mathsf{AC}^0$ for all $m \geq 2$, so it would be interesting to prove $\mathbf{A}$ tame "purely algebraically", independently from the known combinatorial arguments [Ajt83, FSS84, Hås86] and those based on approximating circuits by polynomials over some finite field [Raz87, Smo87]. But tameness of $\mathbf{A}$ is actually equivalent to $\mathrm{MOD_m} \notin \mathsf{AC}^0$ for all $m \geq 2$ by Proposition 3.11, thus confronting us with the challenging task to reprove significant circuit complexity results by relying mainly on new semigroup-theoretic arguments. Such a breakthrough is still to be made.

By contrast, we have shown that $\mathbf{J}$ is not tame. So programs over monoids from $\mathbf{J}$ p-recognize "more regular languages than expected". A natural question to ask is what these regular languages in $\mathcal{P}(\mathbf{J})$ are. Partial results in that direction were obtained in [Gro20].

To conclude we should add, in fairness, that the progress reported here does not in any obvious way bring us closer to major NC^1 complexity subclasses separations. Our concrete contributions here largely concern $\mathcal{P}(\mathbf{DA})$ and $\mathcal{P}(\mathbf{J})$, classes that are well within AC^0. But this work does uncover new ways in which a program can or cannot circumvent the limitations imposed by the underlying monoid algebraic structure available to it.

Acknowledgments

The authors would like to thank the anonymous referees for their detailed reports as well as for their constructive criticism and the many suggestions and comments they made. This helped us to really improve our paper.

References

[Ajt83] Miklós Ajtai. Σ_1^1-formulae on finite structures. *Ann. Pure Appl. Log.*, 24(1):1–48, 1983. doi:10.1016/0168-0072(83)90038-6.

[Alm96] Jorge Almeida. A syntactical proof of locality of DA. *Int. J. Algebra Comput.*, 6(2):165–178, 1996. doi:10.1142/S021819679600009X.

[Bar89] 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. doi:10.1016/0022-0000(89)90037-8.

[BCST92] 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. doi:10.1016/0022-0000(92)90014-A.

[BT88] David A. Mix Barrington and Denis Thérien. Finite monoids and the fine structure of NC^1. *J. ACM*, 35(4):941–952, 1988. doi:10.1145/48014.63138.

[CPS06a] Laura Chaubard, Jean-Éric Pin, and Howard Straubing. Actions, wreath products of $\mathcal{C}$-varieties and concatenation product. *Theor. Comput. Sci.*, 356(1-2):73–89, 2006. doi:10.1016/j.tcs.2006.01.039.

[CPS06b] Laura Chaubard, Jean-Eric Pin, and Howard Straubing. First order formulas with modular predicates. In *21th IEEE Symposium on Logic in Computer Science (LICS 2006), 12-15 August 2006, Seattle, WA, USA, Proceedings*, pages 211–220, 2006. doi:10.1109/LICS.2006.24.

[Dar14] Luc Dartois. *Méthodes algébriques pour la théorie des automates*. PhD thesis, Université Paris Diderot, Paris, 2014. URL: http://lacl.fr/~ldartois/these.pdf.

[DP13] Luc Dartois and Charles Paperman. Two-variable first order logic with modular predicates over words. In *30th International Symposium on Theoretical Aspects of Computer Science, STACS 2013, February 27 - March 2, 2013, Kiel, Germany*, pages 329–340, 2013. doi:10.4230/LIPIcs.STACS.2013.329.

[DP14] Luc Dartois and Charles Paperman. Adding modular predicates. *CoRR*, abs/1401.6576, 2014. URL: http://arxiv.org/abs/1401.6576.

[Eil74] Samuel Eilenberg. *Automata, Languages, and Machines. Volume A*. Pure and applied mathematics. Academic Press, New York, 1974. URL: https://www.sciencedirect.com/bookseries/pure-and-applied-mathematics/vol/59/part/PA.

[Eil76] Samuel Eilenberg. *Automata, Languages, and Machines. Volume B*. Pure and applied mathematics. Academic Press, New York, 1976. URL: https://www.sciencedirect.com/bookseries/pure-and-applied-mathematics/vol/59/part/PB.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984. doi:10.1007/BF01744431.

[GMS17] Nathan Grosshans, Pierre McKenzie, and Luc Segoufin. 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*, pages 2:1–2:20, 2017. doi:10.4230/LIPIcs.MFCS.2017.2.

[Gro18] Nathan Grosshans. *The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids*. PhD thesis, University of Paris-Saclay, France, 2018. URL: https://tel.archives-ouvertes.fr/tel-01935719.

[Gro20] Nathan Grosshans. The power of programs over monoids in J. In *Language and Automata Theory and Applications - 14th International Conference, LATA 2020, Milan, Italy, March 4-6, 2020, Proceedings*, pages 315–327, 2020. URL: https://hal.archives-ouvertes.fr/hal-02414771, doi:10.1007/978-3-030-40608-0_22.

[GT03] Ricard Gavaldà and Denis Thérien. Algebraic characterizations of small classes of Boolean functions. In *STACS 2003, 20th Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, February 27 - March 1, 2003, Proceedings*, pages 331–342, 2003. doi:10.1007/3-540-36494-3_30.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986. doi:10.1145/12130.12132.

[LTT06] Clemens Lautemann, Pascal Tesson, and Denis Thérien. An algebraic point of view on the Crane Beach property. In Zoltán Ésik, editor, *Computer Science Logic, 20th International Workshop,* *CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, pages 426–440, 2006. doi:10.1007/11874683_28.

[MPT91] Pierre McKenzie, Pierre Péladeau, and Denis Thérien. NC^1: The automata-theoretic viewpoint. *Comput. Complex.*, 1:330–359, 1991. doi:10.1007/BF01212963.

[MR65] Ward D. Maurer and John L. Rhodes. A property of finite simple non-abelian groups. *Proc. Amer. Math. Soc.*, 16(3):552–554, 1965. doi:10.1090/s0002-9939-1965-0175971-0.

[Pap14] Charles Paperman. *Circuits booléens, prédicats modulaires et langages réguliers*. PhD thesis, Université Paris Diderot, Paris, 2014. URL: https://paperman.name/data/these.pdf.

[Pél90] Pierre Péladeau. *Classes de circuits booléens et variétés de monoïdes*. PhD thesis, Université Pierre-et-Marie-Curie (Paris-VI), Paris, France, 1990.

[Pin86] Jean-Éric Pin. *Varieties of Formal Languages*. North Oxford, London and Plenum, New-York, 1986. (Traduction de Variétés de langages formels).

[PS05] Jean-Éric Pin and Howard Straubing. Some results on $\mathcal{C}$-varieties. *RAIRO Theor. Informatics Appl.*, 39(1):239–262, 2005. doi:10.1051/ita:2005014.

[PST97] Pierre Péladeau, Howard Straubing, and Denis Thérien. Finite semigroup varieties defined by programs. *Theor. Comput. Sci.*, 180(1-2):325–339, 1997. doi:10.1016/S0304-3975(96)00297-6.

[Raz87] Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987. doi:10.1007/bf01137685.

[Rei82] Jan Reiterman. The Birkhoff theorem for finite algebras. *Algebra Universalis*, 14(1):1–10, 1982. doi:10.1007/bf02483902.

[Sim75] Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975*, pages 214–222, 1975. doi:10.1007/3-540-07407-4_23.

[Smo87] 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, 1987, New York, New York, USA*, pages 77–82, 1987. doi:10.1145/28395.28404.

[Str94] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhauser, Boston, 1994. doi:10.1007/978-1-4612-0289-9.

[Str00] Howard Straubing. When can one finite monoid simulate another. In *Algorithmic Problems in Groups and Semigroups*, pages 267–288. Birkhäuser Boston, 2000. doi:10.1007/978-1-4612-1388-8_15.

[Str01] Howard Straubing. Languages defined with modular counting quantifiers. *Inf. Comput.*, 166(2):112–132, 2001. doi:10.1006/inco.2000.2923.

[Str02] Howard Straubing. On logical descriptions of regular languages. In *LATIN 2002: Theoretical Informatics, 5th Latin American Symposium, Cancun, Mexico, April 3-6, 2002, Proceedings*, pages 528–538, 2002. doi:10.1007/3-540-45995-2_46.

[Tes98] Pascal Tesson. An algebraic approach to communication complexity. Master's thesis, McGill University, Montreal, Canada, 1998. URL: https://escholarship.mcgill.ca/concern/theses/9g54xk72r.

[Tes03] Pascal Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, McGill University, Montreal, Canada, 2003. URL: https://escholarship.mcgill.ca/concern/theses/j9602115z.

[Til87] Bret Tilson. Categories as algebra: An essential ingredient in the theory of monoids. *J. Pure Appl. Algebra*, 48(1-2):83–198, 1987. doi:10.1016/0022-4049(87)90108-3.

[TT02a] Pascal Tesson and Denis Thérien. The computing power of programs over finite monoids. *J. Autom. Lang. Comb.*, 7(2):247–258, 2002. doi:10.25596/jalc-2002-247.

[TT02b] Pascal Tesson and Denis Thérien. Diamonds are forever: the variety DA. In *Semigroups, Algorithms, Automata and Languages*, pages 475–499, 2002. doi:10.1142/9789812776884_0021.