The Power of Programs over Monoids in J

Nathan Grosshans[1,2](✉)

[1] DI ENS, ENS, CNRS, PSL University, Paris, France
nathan.grosshans@polytechnique.edu
[2] Inria, Paris, France
https://www.di.ens.fr/~ngrosshans/

Abstract. The model of programs over (finite) monoids, introduced by Barrington and Thérien, gives an interesting way to characterise the circuit complexity class NC^1 and its subclasses and showcases deep connections with algebraic automata theory. In this article, we investigate the computational power of programs over monoids in **J**, a small variety of finite aperiodic monoids. First, we give a fine hierarchy within the class of languages recognised by programs over monoids from **J**, based on the length of programs but also some parametrisation of **J**. Second, and most importantly, we make progress in understanding what regular languages can be recognised by programs over monoids in **J**. We show that those programs actually can recognise all languages from a class of restricted dot-depth one languages, using a non-trivial trick, and conjecture that this class suffices to characterise the regular languages recognised by programs over monoids in **J**.

1 Introduction

In computational complexity theory, many hard still open questions concern relationships between complexity classes that are expected to be quite small in comparison to the mainstream complexity class P of tractable languages. One of the smallest such classes is NC^1, the class of languages decided by Boolean circuits of polynomial length, logarithmic depth and bounded fan-in, a relevant and meaningful class, that has many characterisations but whose internal structure still mostly is a mystery. Indeed, among its most important subclasses, we count AC^0, CC^0 and ACC^0: all of them are conjectured to be different from each other and strictly within NC^1, but despite many efforts for several decades, this could only be proved for the first of those classes.

In the late eighties, Barrington and Thérien [3], building on Barrington's celebrated theorem [2], gave an interesting viewpoint on those conjectures, relying on algebraic automata theory. They defined the notion of a program over a monoid M: a sequence of instructions (i, f), associating through function f some element of M to the letter at position i in the input of fixed length. In that way, the program outputs an element of M for every input word, by multiplying out the elements given by the instructions for that word; acceptance or rejection then depends on that outputted element. A language of words of arbitrary length

© Springer Nature Switzerland AG 2020
A. Leporati et al. (Eds.): LATA 2020, LNCS 12038, pp. 315–327, 2020.
https://doi.org/10.1007/978-3-030-40608-0_22

is consequently recognised in a non-uniform fashion, by a sequence of programs over some fixed monoid, one for each possible input length; when that sequence is of polynomial length, it is said that the monoid p-recognises that language. Barrington and Thérien's discovery is that NC^1 and almost all of its significant subclasses can each be exactly characterised by p-recognition over monoids taken from some suitably chosen variety of finite monoids (a class of finite monoids closed under basic operations on monoids). For instance, NC^1, AC^0, CC^0 and ACC^0 correspond exactly to p-recognition by, respectively, finite monoids, finite aperiodic monoids, finite solvable groups and finite solvable monoids. Understanding the internal structure of NC^1 thus becomes a matter of understanding what finite monoids from some particular variety are able to p-recognise.

It soon became clear that regular languages play a central role in understanding p-recognition: McKenzie, Péladeau and Thérien indeed observed [12] that finite monoids from a variety $\mathbf{V}$ and a variety $\mathbf{W}$ p-recognise the same languages if and only if they p-recognise the same regular languages. Otherwise stated, most conjectures about the internal structure of NC^1 can be reformulated as a statement about where one or several regular languages lie within that structure. This is why a line of previous works got interested into various notions of tameness, capturing the fact that for a given variety of finite monoids, p-recognition does not offer much more power than classical morphism-recognition when it comes to regular languages (see [8,10,11,13,14,20–22]).

This paper is a contribution to an ongoing study of what regular languages can be p-recognised by monoids taken from "small" varieties, started with the author's Ph.D. thesis [7]. In a previous paper by the author with McKenzie and Segoufin [8], a novel notion of tameness was introduced and shown for the "small" variety of finite aperiodic monoids $\mathbf{DA}$. This allowed them to characterise the class of regular languages p-recognised by monoids from $\mathbf{DA}$ as those recognised by so called quasi-$\mathbf{DA}$ morphisms and represented a first small step towards a new proof that the variety $\mathbf{A}$ of finite aperiodic monoids is tame. This is a statement equivalent to Furst's, Saxe's, Sipser's [6] and Ajtai's [1] well-known lower bound result about AC^0. In [8], the authors also observed that, while $\mathbf{DA}$ "behaves well" with respect to p-recognition of regular languages, the variety $\mathbf{J}$, a subclass of $\mathbf{DA}$, does, in contrast, "behave badly" in the sense that monoids from $\mathbf{J}$ do p-recognise regular languages that are not recognised by quasi-$\mathbf{J}$ morphisms.

Now, $\mathbf{J}$ is a well-studied and fundamental variety in algebraic automata theory (see, e.g., [15,16]), corresponding through classical morphism-recognition to the class of regular languages in which membership depends on the presence or absence of a finite set of words as subwords. This paper is a contribution to the understanding of the power of programs over monoids in $\mathbf{J}$, a knowledge that certainly does not bring us closer to a new proof of the tameness of $\mathbf{A}$ (as we are dealing with a strict subvariety of $\mathbf{DA}$), but that is motivated by the importance of $\mathbf{J}$ in algebraic automata theory and the unexpected power of programs over monoids in $\mathbf{J}$. The results we present in this article are twofold: first, we exhibit a fine hierarchy within the class of languages p-recognised by monoids from $\mathbf{J}$, depending on the length of those programs and on a parametrisation of $\mathbf{J}$;

second, we show that a whole class of regular languages, that form a subclass of dot-depth one languages [15], are p-recognised by monoids from $\mathbf{J}$ while, in general, they are not recognised by any quasi-$\mathbf{J}$ morphism. This class roughly corresponds to dot-depth one languages where detection of a given factor does work only when it does not appear too often as a subword. We actually even conjecture that this class of languages with additional positional modular counting (that is, letters can be differentiated according to their position modulo some fixed number) corresponds exactly to all those p-recognised by monoids in $\mathbf{J}$, a statement that is interesting in itself for algebraic automata theory.

Organisation of the Paper. Following the present introduction, Sect. 2 is dedicated to the necessary preliminaries. In Sect. 3, we present the results about the fine hierarchy and in Sect. 4 we expose the results concerning the regular languages p-recognised by monoids from $\mathbf{J}$. Section 5 gives a short conclusion.

Note. This article is based on unpublished parts of the author's Ph.D. thesis [7].

2 Preliminaries

2.1 Various Mathematical Materials

We assume the reader is familiar with the basics of formal language theory, semigroup theory and recognition by morphisms, that we might designate by classical recognition; for those, we only specify some things and refer the reader to the two classical references of the domain by Eilenberg [4,5] and Pin [16].

General Notations and Conventions. Let $i, j \in \mathbb{N}$. We shall denote by $[\![i, j]\!]$ the set of all $n \in \mathbb{N}$ verifying $i \leq n \leq j$. We shall also denote by $[i]$ the set $[\![1, i]\!]$. Given some set E, we shall denote by $\mathfrak{P}(E)$ the powerset of E. All our alphabets and words will always be finite; the empty word will be denoted by ε.

Varieties and Languages. A *variety of monoids* is a class of finite monoids closed under submonoids, Cartesian product and morphic images. A *variety of semigroups* is defined similarly. When dealing with varieties, we consider only finite monoids and semigroups, each having an *idempotent power*, a smallest $\omega \in \mathbb{N}_{>0}$ such that $x^\omega = x^{2\omega}$ for any element x. To give an example, the variety of finite aperiodic monoids, denoted by $\mathbf{A}$, contains all finite monoids M such that, given ω its idempotent power, $x^\omega = x^{\omega+1}$ for all $x \in M$.

To each variety $\mathbf{V}$ of monoids or semigroups we associate the class $\mathcal{L}(\mathbf{V})$ of languages such that, respectively, their syntactic monoid or semigroup belongs to $\mathbf{V}$. For instance, $\mathcal{L}(\mathbf{A})$ is well-known to be the class of star-free languages.

Quasi $\mathbf{V}$ *Languages.* 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 [17]. Let φ be a surjective morphism from Σ^* to a finite monoid M. For all k consider the subset $\varphi(\Sigma^k)$ of M (where

Σ^k is the set of words over Σ of length k). As M is finite there is a k such that $\varphi(\Sigma^{2k}) = \varphi(\Sigma^k)$. This implies that $\varphi(\Sigma^k)$ is a semigroup. The semigroup given by the smallest such k is called the *stable semigroup of* φ. If S is the stable semigroup of φ, S^1 is called *the stable monoid of* φ. If $\mathbf{V}$ is a variety of monoids or semigroups, then we shall denote by $\mathbf{QV}$ the class of such surjective morphisms whose stable monoid or semigroup, respectively, is in $\mathbf{V}$ and by $\mathcal{L}(\mathbf{QV})$ the class of languages whose syntactic morphism is in $\mathbf{QV}$.

Programs over Monoids. Programs over monoids form a non-uniform model of computation, first defined by Barrington and Thérien [3], extending Barrington's permutation branching program model [2]. Let M be a finite monoid and Σ an alphabet. A *program P over M on Σ^n* is a finite sequence of instructions of the form (i, f) where $i \in [n]$ and $f \in M^\Sigma$; said otherwise, it is a word over $([n] \times M^\Sigma)$. The *length* of P, denoted by $|P|$, is the number of its instructions. The program P defines a function from Σ^n to M as follows. On input $w \in \Sigma^n$, each instruction (i, f) outputs the monoid element $f(w_i)$. A sequence of instructions then yields a sequence of elements of M and their product is the output $P(w)$ of the program. A language $L \subseteq \Sigma^n$ is consequently recognised by P whenever there exists $F \subseteq M$ such that $L = P^{-1}(F)$.

A language L over Σ is *recognised* by a sequence of programs $(P_n)_{n\in\mathbb{N}}$ over some finite monoid M if for each n, the program P_n is on Σ^n and recognises $L^{=n} = L \cap \Sigma^n$. We say $(P_n)_{n\in\mathbb{N}}$ is of length $s(n)$ for $s\colon \mathbb{N} \to \mathbb{N}$ whenever $|P_n| = s(n)$ for all $n \in \mathbb{N}$ and that it is of length at most $s(n)$ whenever there exists $\alpha \in \mathbb{R}_{>0}$ verifying $|P_n| \leq \alpha \cdot s(n)$ for all $n \in \mathbb{N}$.

For $s\colon \mathbb{N} \to \mathbb{N}$ and $\mathbf{V}$ a variety of monoids, we denote by $\mathcal{P}(\mathbf{V}, s(n))$ the class of languages recognised by sequences of programs over monoids in $\mathbf{V}$ of length at most $s(n)$. The class $\mathcal{P}(\mathbf{V}) = \bigcup_{k\in\mathbb{N}} \mathcal{P}(\mathbf{V}, n^k)$ is then the class of languages *p*-recognised by a monoid in $\mathbf{V}$, i.e. recognised by sequences of programs over monoids in $\mathbf{V}$ of polynomial length.

The following is an important property of $\mathcal{P}(\mathbf{V})$.

Proposition 1 ([12, Corollary 3.5]). *Let $\mathbf{V}$ be a variety of monoids, then $\mathcal{P}(\mathbf{V})$ is closed under Boolean operations.*

Given two alphabets Σ and Γ, a Γ-program on Σ^n for $n \in \mathbb{N}$ is defined just like a program over some finite monoid M on Σ^n, except that instructions output letters from Γ and thus that the program outputs words over Γ. Let now $L \subseteq \Sigma^*$ and $K \subseteq \Gamma^*$. We say that *L program-reduces to K* if and only if there exists a sequence $(\Psi_n)_{n\in\mathbb{N}}$ of Γ-programs (the program-reduction) such that Ψ_n is on Σ^n and $L^{=n} = \Psi_n^{-1}(K^{=|\Psi_n|})$ for each $n \in \mathbb{N}$. The following proposition shows closure of $\mathcal{P}(\mathbf{V})$ also under program-reductions.

Proposition 2 ([7, Proposition 3.3.12 and Corollary 3.4.3]). *Let Σ and Γ be two alphabets. Let $\mathbf{V}$ be a variety of monoids. Given $K \subseteq \Gamma^*$ in $\mathcal{P}(\mathbf{V}, s(n))$ for $s\colon \mathbb{N} \to \mathbb{N}$ and $L \subseteq \Sigma^*$ from which there exists a program-reduction to K of length $t(n)$, for $t\colon \mathbb{N} \to \mathbb{N}$, we have that $L \in \mathcal{P}(\mathbf{V}, s(t(n)))$. In particular, when K is recognised (classically) by a monoid in $\mathbf{V}$, we have that $L \in \mathcal{P}(\mathbf{V}, t(n))$.*

2.2 Tameness and the Variety J

We won't introduce any of the proposed notions of tameness but will only state that the main consequence for a variety of monoids $\mathbf{V}$ to be tame in the sense of [8] is that $\mathcal{P}(\mathbf{V}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QV})$. This consequence has far-reaching implications from a computational-complexity-theoretic standpoint when $\mathcal{P}(\mathbf{V})$ happens to be equal to a circuit complexity class. For instance, tameness for $\mathbf{A}$ implies that $\mathcal{P}(\mathbf{A}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QA})$, which is equivalent to the fact that AC^0 does not contain the language $\mathrm{MOD_m}$ of words over $\{0, 1\}$ containing a number of 1s not divisible by m for any $m \in \mathbb{N}, m \geq 2$ (a central result in complexity theory [1,6]).

Let us now define the variety of monoids $\mathbf{J}$. A finite monoid M of idempotent power ω belongs to $\mathbf{J}$ if and only if $(xy)^\omega = (xy)^\omega x = y(xy)^\omega$ for all $x, y \in M$. It is a strict subvariety of the variety $\mathbf{DA}$, containing all finite monoids M of idempotent power ω such that $(xy)^\omega = (xy)^\omega x(xy)^\omega$ for all $x, y \in M$, itself a strict subvariety of $\mathbf{A}$. The variety $\mathbf{J}$ is a "small" one, well within $\mathbf{A}$.

We now give some specific definitions and results about $\mathbf{J}$ that we will use, based essentially on [9], but also on [16, Chapter 4, Section 1].

For some alphabet Σ and each $k \in \mathbb{N}$, let us define the equivalence relation $\sim_k$ on Σ^* by $u \sim_k v$ if and only if u and v have the same set of k-subwords (subwords of length at most k), for all $u, v \in \Sigma^*$. The relation $\sim_k$ is a congruence of finite index on Σ^*. For an alphabet Σ and a word $u \in \Sigma^*$, we shall write $u \sqcup\!\sqcup \Sigma^*$ for the language of all words over Σ having u as a subword. In the following, we consider that $\sqcup\!\sqcup$ has precedence over $\cup$ and $\cap$ (but of course not over concatenation).

We define the *class of piecewise testable languages* $\mathcal{PT}$ as the class of regular languages such that for every alphabet Σ, we associate to Σ^* the set $\mathcal{PT}(\Sigma^*)$ of all languages over Σ that are Boolean combinations of languages of the form $u \sqcup\!\sqcup \Sigma^*$ where $u \in \Sigma^*$. In fact, $\mathcal{PT}(\Sigma^*)$ is the set of languages over Σ equal to a union of $\sim_k$-classes for some $k \in \mathbb{N}$ (see [18]). Simon showed [18] that a language is piecewise testable if and only if its syntactic monoid is in $\mathbf{J}$, i.e. $\mathcal{PT} = \mathcal{L}(\mathbf{J})$.

We can define a hierarchy of piecewise testable languages in a natural way. For $k \in \mathbb{N}$, let the *class of k-piecewise testable languages* $\mathcal{PT}_k$ be the class of regular languages such that for every alphabet Σ, we associate to Σ^* the set $\mathcal{PT}_k(\Sigma^*)$ of all languages over Σ that are Boolean combinations of languages of the form $u \sqcup\!\sqcup \Sigma^*$ where $u \in \Sigma^*$ with $|u| \leq k$. We then have that $\mathcal{PT}_k(\Sigma^*)$ is the set of languages over Σ equal to a union of $\sim_k$-classes. Let us define $\mathbf{J_k}$ the inclusion-wise smallest variety of monoids containing the quotients of Σ^* by $\sim_k$ for any alphabet Σ: we have that a language is k-piecewise testable if and only if its syntactic monoid belongs to $\mathbf{J_k}$, i.e. $\mathcal{PT}_k = \mathcal{L}(\mathbf{J_k})$. (See [9, Section 3].)

3 Fine Hierarchy

The first part of our investigation of the computational power of programs over monoids in $\mathbf{J}$ concerns the influence of the length of programs on their computational capabilities.

We say two programs over a same monoid on the same set of input words are *equivalent* if and only if they recognise the same languages. Tesson and Thérien proved in [23] that for any monoid M in $\mathbf{DA}$, there exists some $k \in \mathbb{N}$ such that for any alphabet Σ there is a constant $c \in \mathbb{N}_{>0}$ verifying that any program over M on Σ^n for $n \in \mathbb{N}$ is equivalent to a program over M on Σ^n of length at most $c \cdot n^k$. Since $\mathbf{J} \subset \mathbf{DA}$, any monoid in $\mathbf{J}$ does also have this property. However, this does not imply that there exists some $k \in \mathbb{N}$ working for all monoids in $\mathbf{J}$, i.e. that $\mathcal{P}(\mathbf{J})$ collapses to $\mathcal{P}(\mathbf{J}, n^k)$.

In this section, we show on the one hand that, as for $\mathbf{DA}$, while $\mathcal{P}(\mathbf{J}, s(n))$ collapses to $\mathcal{P}(\mathbf{J})$ for any super-polynomial function $s \colon \mathbb{N} \to \mathbb{N}$, there does not exist any $k \in \mathbb{N}$ such that $\mathcal{P}(\mathbf{J})$ collapses to $\mathcal{P}(\mathbf{J}, n^k)$; and on the other hand that $\mathcal{P}(\mathbf{J_k})$ does optimally collapse to $\mathcal{P}(\mathbf{J_k}, n^{\lceil k/2 \rceil})$ for each $k \in \mathbb{N}$.

3.1 Strict Hierarchy

Given $k, n \in \mathbb{N}$, we say that σ is a *k-selector over n* if σ is a function of $\mathfrak{P}([n])^{[n]^k}$ that associates a subset of $[n]$ to each vector in $[n]^k$. For any sequence $\Delta = (\sigma_n)_{n \in \mathbb{N}}$ such that σ_n is a k-selector over n for each $n \in \mathbb{N}$—a sequence we will call a *sequence of k-selectors*—, we set $L_\Delta = \bigcup_{n \in \mathbb{N}} K_{n,\sigma_n}$, where for each $n \in \mathbb{N}$, the language K_{n,σ_n} is the set of words over $\{0,1\}$ of length $(k+1) \cdot n$ that can be decomposed into $k+1$ consecutive blocks $u^{(1)}, u^{(2)}, \ldots, u^{(k)}, v$ of n letters where the first k blocks each contain 1 exactly once and uniquely define a vector ρ in $[n]^k$, where for all $i \in [k]$, ρ_i is given by the position of the only 1 in $u^{(i)}$ (i.e. $u^{(i)}_{\rho_i} = 1$) and v is such that there exists $j \in \sigma_n(\rho)$ verifying that v_j is 1. Observe that for any k-selector σ_0 over 0, we have $K_{0,\sigma_0} = \emptyset$.

We now proceed similarly to what has been done in Subsection 5.1 in [8] to show, on one hand, that for all $k \in \mathbb{N}$, there is a monoid M_k in $\mathbf{J_{2k+1}}$ such that for any sequence of k-selectors Δ, the language L_Δ is recognised by a sequence of programs over M_k of length at most n^{k+1}; and, on the other hand, that for all $k \in \mathbb{N}$ there is a sequence of k-selectors Δ such that for any finite monoid M and any sequence of programs $(P_n)_{n \in \mathbb{N}}$ over M of length at most n^k, the language L_Δ is not recognised by $(P_n)_{n \in \mathbb{N}}$.

We obtain the following proposition.

Proposition 3. *For all $k \in \mathbb{N}$, we have $\mathcal{P}(\mathbf{J}, n^k) \subset \mathcal{P}(\mathbf{J}, n^{k+1})$. More precisely, for all $k \in \mathbb{N}$ and $d \in \mathbb{N}, d \leq \lceil \frac{k}{2} \rceil - 1$, we have $\mathcal{P}(\mathbf{J_k}, n^d) \subset \mathcal{P}(\mathbf{J_k}, n^{d+1})$.*

3.2 Collapse

Looking at Proposition 3, it looks at first glance rather strange that, for each $k \in \mathbb{N}$, we can only prove strictness of the hierarchy inside $\mathcal{P}(\mathbf{J_k})$ up to exponent $\lceil \frac{k}{2} \rceil$. We now show, in a way similar to Subsection 5.2 in [8], that in fact $\mathcal{P}(\mathbf{J_k})$ does collapse to $\mathcal{P}(\mathbf{J_k}, n^{\lceil k/2 \rceil})$ for all $k \in \mathbb{N}$, showing Proposition 3 to be optimal in some sense.

Proposition 4. *Let $k \in \mathbb{N}$. Let $M \in \mathbf{J_k}$ and Σ be an alphabet. Then there exists a constant $c \in \mathbb{N}_{>0}$ such that any program over M on Σ^n for $n \in \mathbb{N}$ is equivalent to a program over M on Σ^n of length at most $c \cdot n^{\lceil k/2 \rceil}$.*

In particular, $\mathcal{P}(\mathbf{J_k}) = \mathcal{P}(\mathbf{J_k}, n^{\lceil k/2 \rceil})$ for all $k \in \mathbb{N}$.

4 Regular Languages in $\mathcal{P}(\mathbf{J})$

The second part of our investigation of the computational power of programs over monoids in **J** is dedicated to understanding exactly what regular languages can be p-recognised by monoids in **J**.

4.1 Non-tameness of J

It is shown in [8] that $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \nsubseteq \mathcal{L}(\mathbf{QJ})$, thus giving an example of a well-known subvariety of **A** for which p-recognition allows to do unexpected things when recognising a regular language. How far does this unexpected power go?

The first thing to notice is that, though none of them is in $\mathcal{L}(\mathbf{QJ})$, all languages of the form $\Sigma^* u$ and $u\Sigma^*$ for Σ an alphabet and $u \in \Sigma^+$ are in $\mathcal{P}(\mathbf{J})$. Indeed, each of them can be recognised by a sequence of constant-length programs over the syntactic monoid of $u \sqcup\!\sqcup \Sigma^*$: for every input length, just output the image, through the syntactic morphism of $u \sqcup\!\sqcup \Sigma^*$, of the word made of the $|u|$ first or last letters. So, informally stated, programs over monoids in **J** can check for some constant-length beginning or ending of their input words.

But they can do much more. Indeed, the language $(a+b)^* ac^+$ does not belong to $\mathcal{L}(\mathbf{QJ})$ (compute the stable monoid), yet it is in $\mathcal{P}(\mathbf{J})$. The crucial insight is that it can be program-reduced in linear length to the piecewise testable language of all words over $\{a, b, c\}$ having ca as a subword but not the subwords cca, caa and cb by using the following trick (that we shall call "feedback-sweeping") for input length $n \in \mathbb{N}$: read the input letters in the order $2, 1, 3, 2, 4, 3, 5, 4, \ldots, n, n-1$, output the letters read. This has already been observed in [8, Proposition 5].

Lemma 1. $(a+b)^* ac^+ \in \mathcal{P}(\mathbf{J}, n)$.

Using variants of the "feedback-sweeping" reading technique, we can prove that the phenomenon just described is not an isolated case.

Lemma 2. *The languages* $(a+b)^* ac^+$, $(a+b)^* ac^+ a(a+b)^*$, $c^+ a(a+b)^* ac^+$, $(a+b)^* bac^+$ *and* $(a+b)^* ac^+ (a+b)^* ac^+$ *do all belong to* $\mathcal{P}(\mathbf{J}) \setminus \mathcal{L}(\mathbf{QJ})$.

Hence, we are tempted to say that there are "much more" regular languages in $\mathcal{P}(\mathbf{J})$ than just those in $\mathcal{L}(\mathbf{QJ})$, even though it is not clear to us whether $\mathcal{L}(\mathbf{QJ}) \subseteq \mathcal{P}(\mathbf{J})$ or not. But can we show any upper bound on $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg}$? It turns out that we can, relying on two known results.

First, since $\mathbf{J} \subseteq \mathbf{DA}$, we have $\mathcal{P}(\mathbf{J}) \subseteq \mathcal{P}(\mathbf{DA})$, so Theorem 6 in [8], that states $\mathcal{P}(\mathbf{DA}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA})$, implies that $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QDA})$.

Second, let us define an important superclass of the class of piecewise testable languages. Let Σ be an alphabet and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$); we define $[u_1, \ldots, u_k] = \Sigma^* u_1 \Sigma^* \cdots \Sigma^* u_k \Sigma^*$. The *class of dot-depth one languages* is the class of Boolean combinations of languages of the form $\Sigma^* u$, $u\Sigma^*$ and $[u_1, \ldots, u_k]$ for Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u, u_1, \ldots, u_k \in \Sigma^+$. The inclusion-wise smallest variety of semigroups containing all syntactic semigroups of dot-depth one languages is denoted by $\mathbf{J} * \mathbf{D}$ and verifies that $\mathcal{L}(\mathbf{J} * \mathbf{D})$ is exactly the class of dot-depth one languages. (See [11,15,19].) It has been shown in [11, Corollary 8] that $\mathcal{P}(\mathbf{J} * \mathbf{D}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$ (if we extend the program-over-monoid formalism in the obvious way to finite semigroups). Now, we have $\mathbf{J} \subseteq \mathbf{J} * \mathbf{D}$, so that $\mathcal{P}(\mathbf{J}) \subseteq \mathcal{P}(\mathbf{J} * \mathbf{D})$ and hence $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$.

To summarise, we have the following.

Proposition 5. $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} \subseteq \mathcal{L}(\mathbf{QDA}) \cap \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$.

In fact, we conjecture that the inverse inclusion does also hold.

Conjecture 1. $\mathcal{P}(\mathbf{J}) \cap \mathcal{R}\mathsf{eg} = \mathcal{L}(\mathbf{QDA}) \cap \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$.

Why do we think this should be true? Though, for a given alphabet Σ, we cannot decide whether some word $u \in \Sigma^+$ of length at least 2 appears as a factor of any given word w in Σ^* with programs over monoids in $\mathbf{J}$ (because $\Sigma^* u \Sigma^* \notin \mathcal{L}(\mathbf{QDA})$), Lemma 2 and the possibilities offered by the "feedback-sweeping" technique give the impression that we can do it when we are guaranteed that u appears at most a fixed number of times in w, which seems somehow to be what dot-depth one languages become when restricted to belong to $\mathcal{L}(\mathbf{QDA})$. This intuition motivates the definition of *threshold dot-depth one languages.*

4.2 Threshold Dot-Depth One Languages

The idea behind the definition of threshold dot-depth one languages is that we take the basic building blocks of dot-depth one languages, of the form $[u_1, \ldots, u_k]$ for an alphabet Σ, for $k \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$, and restrict them so that, given $l \in \mathbb{N}_{>0}$, membership of a word does really depend on the presence of a given word u_i as a factor if and only if it appears less than l times as a subword.

Definition 1. *Let Σ be an alphabet. For all $u \in \Sigma^+$ and $l \in \mathbb{N}_{>0}$, we define $[u]_l$ to be the language of words over Σ containing u^l as a subword or u as a factor, i.e. $[u]_l = \Sigma^* u \Sigma^* \cup u^l \sqcup\!\sqcup\, \Sigma^*$. Then, for all $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}, k \geq 2$) and $l \in \mathbb{N}_{>0}$, we define $[u_1, \ldots, u_k]_l = [u_1]_l \cdots [u_k]_l$.*

Obviously, for each Σ an alphabet, $k \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$, the language $[u_1, \ldots, u_k]_1$ equals $u_1 \cdots u_k \sqcup\!\sqcup\, \Sigma^*$. Over $\{a, b, c\}$, the language $[ab, c]_3$ contains all words containing a letter c verifying that in the prefix up to that letter, $ababab$ appears as a subword or ab appears as a factor. Finally, the language $(a + b)^* ac^+$ over $\{a, b, c\}$ of Lemma 1 is equal to $[c, a]_2^{\complement} \cap [c, b]_2^{\complement} \cap [ac]_2$.

We then define a *threshold dot-depth one language* as any Boolean combination of languages of the form $\Sigma^* u$, $u \Sigma^*$ and $[u_1, \ldots, u_k]_l$ for Σ an alphabet, for $k, l \in \mathbb{N}_{>0}$ and $u, u_1, \ldots, u_k \in \Sigma^+$.

Confirming the intuition briefly given above, the technique of "feedback-sweeping" can indeed be pushed further to prove that the whole class of threshold dot-depth one languages is contained in $\mathcal{P}(\mathbf{J})$, and we dedicate the remainder of this section to prove it. Concerning Conjecture 1, our intuition leads us to believe that, in fact, the class of threshold dot-depth one languages with additional positional modular counting is exactly $\mathcal{L}(\mathbf{QDA}) \cap \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$. We simply refer the interested reader to Section 5.4 of the author's Ph.D. thesis [7], that contains a partial result supporting this belief, too technical and long to be presented here.

Let us now move on to the proof of the following theorem.

Theorem 1. *Every threshold dot-depth one language belongs to* $\mathcal{P}(\mathbf{J})$.

As $\mathcal{P}(\mathbf{J})$ is closed under Boolean operations (Proposition 1), our goal is to prove, given an alphabet Σ, given $l \in \mathbb{N}_{>0}$ and $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), that $[u_1, \dots, u_k]_l$ is in $\mathcal{P}(\mathbf{J})$; the case of $\Sigma^* u$ and $u\Sigma^*$ for $u \in \Sigma^+$ is easily handled (see the discussion at the beginning of Subsect. 4.1). To do this, we need to put $[u_1, \dots, u_k]_l$ in some normal form. It is readily seen that $[u_1, \dots, u_k]_l = \bigcup_{q_1,\dots,q_k \in \{1,l\}} L^{(l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)}$ where the $L^{(l)}_{(u_i,q_i)}$'s are defined thereafter.

Definition 2. *Let* Σ *be an alphabet.*

For all $u \in \Sigma^+$, $l \in \mathbb{N}_{>0}$ *and* $\alpha \in [l]$, *set* $L^{(l)}_{(u,\alpha)} = \begin{cases} \Sigma^* u \Sigma^* & \text{if } \alpha < l \\ u^l \sqcup\!\sqcup \Sigma^* & \text{otherwise} \end{cases}$.

Building directly a sequence of programs over a monoid in **J** that decides $L^{(l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)}$ for some alphabet Σ and $q_1, \dots, q_k \in \{1, l\}$ seems however tricky. We need to split things further by controlling precisely how many times each u_i for $i \in [k]$ appears in the right place when it does less than l times. To do this, we consider, for each $\alpha \in [l]^k$, the language $R^\alpha_l(u_1, \dots, u_k)$ defined below.

Definition 3. *Let* Σ *be an alphabet.*

For all $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), $l \in \mathbb{N}_{>0}$, $\alpha \in [l]^k$, *we set*

$$\begin{aligned} R^\alpha_l(u_1, \dots, u_k) =& ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \\ & \bigcap_{i \in [k], \alpha_i < l} \left(({u_1}^{\alpha_1} \cdots {u_i}^{\alpha_i + 1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\right)^{\complement} . \end{aligned}$$

Now, for a given $\alpha \in [l]^k$, we are interested in the words of $R^\alpha_l(u_1, \dots, u_k)$ such that for each $i \in [k]$ verifying $\alpha_i < l$, the word u_i indeed appears as a factor in the right place. We thus introduce a last language $S^\alpha_l(u_1, \dots, u_k)$ defined as follows.

Definition 4. *Let* Σ *be an alphabet.*

For all $u_1, \dots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$), $l \in \mathbb{N}_{>0}$, $\alpha \in [l]^k$, *we set*

$$S^\alpha_l(u_1, \dots, u_k) = \bigcap_{i \in [k], \alpha_i < l} \left(({u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}) \sqcup\!\sqcup \Sigma^*\right) u_i \left(({u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^*\right).$$

We now have the normal form we were looking for to prove Theorem 1: $[u_1, \dots, u_k]_l$ is equal to the union, over all $\alpha \in [l]^k$, of the intersection of $R^\alpha_l(u_1, \dots, u_k)$ and $S^\alpha_l(u_1, \dots, u_k)$. Though rather intuitive, the correctness of this decomposition is not so straightforward to prove and, actually, we can only prove it when for each $i \in [k]$, the letters in u_i are all distinct.

Lemma 3. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. Then,*

$$\bigcup_{q_1,\ldots,q_k \in \{1,l\}} L^{(l)}_{(u_1,q_1)} \cdots L^{(l)}_{(u_k,q_k)} = \bigcup_{\alpha \in [l]^k} \big(R^\alpha_l(u_1,\ldots,u_k) \cap S^\alpha_l(u_1,\ldots,u_k)\big).$$

Our goal now is to prove, given an alphabet Σ, given $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct, that for any $\alpha \in [l]^k$, the language $R^\alpha_l(u_1,\ldots,u_k) \cap S^\alpha_l(u_1,\ldots,u_k)$ is in $\mathcal{P}(\mathbf{J})$; closure of $\mathcal{P}(\mathbf{J})$ under union (Proposition 1) consequently entails that $[u_1,\ldots,u_k]_l \in \mathcal{P}(\mathbf{J})$. The way $R^\alpha_l(u_1,\ldots,u_k)$ and $S^\alpha_l(u_1,\ldots,u_k)$ are defined allows us to reason as follows. For each $i \in [k]$ verifying $\alpha_i < l$, let L_i be the language of words w over Σ containing $x_{i,1}{u_i}^{\alpha_i}x_{i,2}$ as a subword but not $x_{i,1}{u_i}^{\alpha_i+1}x_{i,2}$ and such that $w = y_1 u_i y_2$ with $y_1 \in x_{i,1} \sqcup\!\sqcup \Sigma^*$ and $y_2 \in x_{i,2} \sqcup\!\sqcup \Sigma^*$, where $x_{i,1} = {u_1}^{\alpha_1} \cdots {u_{i-1}}^{\alpha_{i-1}}$ and $x_{i,2} = {u_{i+1}}^{\alpha_{i+1}} \cdots {u_k}^{\alpha_k}$. If we manage to prove that for each $i \in [k]$ verifying $\alpha_i < l$ we have $L_i \in \mathcal{P}(\mathbf{J})$, we can conclude that $R^\alpha_l(u_1,\ldots,u_k) \cap S^\alpha_l(u_1,\ldots,u_k) = ({u_1}^{\alpha_1} \cdots {u_k}^{\alpha_k}) \sqcup\!\sqcup \Sigma^* \cap \bigcap_{i \in [k], \alpha_i < l} L_i$ does belong to $\mathcal{P}(\mathbf{J})$ by closure of $\mathcal{P}(\mathbf{J})$ under intersection, Proposition 1. The lemma that follows, the main lemma in the proof of Theorem 1, exactly shows this. The proof crucially uses the "feedback sweeping" technique, but note that we actually don't know how to prove it when we do not enforce that for each $i \in [k]$, the letters in u_i are all distinct.

Lemma 4. *Let Σ be an alphabet and $u \in \Sigma^+$ such that its letters are all distinct. For all $\alpha \in \mathbb{N}_{>0}$ and $x_1, x_2 \in \Sigma^*$, we have*

$$(x_1 u^\alpha x_2) \sqcup\!\sqcup \Sigma^* \cap \big((x_1 u^{\alpha+1} x_2) \sqcup\!\sqcup \Sigma^*\big)^{\complement} \cap (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*) \in \mathcal{P}(\mathbf{J}) \ .$$

Proof (Sketch). Let Σ be an alphabet and $u \in \Sigma^+$ such that its letters are all distinct. Let $\alpha \in \mathbb{N}_{>0}$ and $x_1, x_2 \in \Sigma^*$. We let

$$L = (x_1 u^\alpha x_2) \sqcup\!\sqcup \Sigma^* \cap \big((x_1 u^{\alpha+1} x_2) \sqcup\!\sqcup \Sigma^*\big)^{\complement} \cap (x_1 \sqcup\!\sqcup \Sigma^*)u(x_2 \sqcup\!\sqcup \Sigma^*) \ .$$

If $|u| = 1$, the lemma follows trivially because L is piecewise testable and hence belongs to $\mathcal{L}(\mathbf{J})$, so we assume $|u| > 1$.

For each letter $a \in \Sigma$, we shall use $2|u| - 1$ distinct decorated letters of the form $a^{(i)}$ for some $i \in [\![0, 2|u| - 2]\!]$, using the convention that $a^{(0)} = a$; of course, for two distinct letters $a, b \in \Sigma$, we have that $a^{(i)}$ and $b^{(j)}$ are distinct for all $i, j \in [\![0, 2|u| - 2]\!]$. We denote by A the alphabet of these decorated letters. The main idea of the proof is, for a given input length $n \in \mathbb{N}$, to build an A-program Ψ_n over Σ^n such that, given an input word $w \in \Sigma^n$, it first ouputs the $|u| - 1$ first letters of w and then, for each i going from $|u|$ to n, outputs w_i, followed by $w^{(1)}_{i-1} \cdots w^{(|u|-1)}_{i-|u|+1}$ (a "sweep" of $|u| - 1$ letters backwards down to position $i - |u| + 1$, decorating the letters incrementally) and finally by $w^{(|u|)}_{i-|u|+2} \cdots w^{(2|u|-2)}_i$ (a "sweep" forwards up to position i, continuing the incremental decoration of the letters). The idea behind this way of rearranging and decorating letters is that, given an input word $w \in \Sigma^n$, as long as we make sure that w and thus

$\Psi_n(w)$ do contain $x_1 u^\alpha x_2$ as a subword but not $x_1 u^{\alpha+1} x_2$, then $\Psi_n(w)$ can be decomposed as $\Psi_n(w) = y_1 z y_2$ where $y_1 \in x_1 \sqcup\!\sqcup \Sigma^*$, $y_2 \in x_2 \sqcup\!\sqcup \Sigma^*$, and $|y_1|, |y_2|$ are minimal, with z containing $u^\beta u^{(1)}_{|u|-1} \cdots u^{(|u|-1)}_1 u^{(|u|)}_2 \cdots u^{(2|u|-2)}_{|u|} u^{\alpha-\beta}$ as a subword for some $\beta \in [\alpha]$ if and only if $w \in (x_1 \sqcup\!\sqcup \Sigma^*) u (x_2 \sqcup\!\sqcup \Sigma^*)$. This means we can check whether $w \in L$ by testing whether w belongs to some fixed piecewise testable language over A.

As explained before stating the previous lemma, we can now use it to prove the result we were aiming for.

Proposition 6. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. For all $\alpha \in [l]^k$, we have $R^\alpha_l(u_1, \ldots, u_k) \cap S^\alpha_l(u_1, \ldots, u_k) \in \mathcal{P}(\mathbf{J})$.*

We thus derive the awaited corollary.

Corollary 1. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$) such that for each $i \in [k]$, the letters in u_i are all distinct. Then, $[u_1, \ldots, u_k]_l \in \mathcal{P}(\mathbf{J})$.*

However, what we really want to obtain is that $[u_1, \ldots, u_k]_l \in \mathcal{P}(\mathbf{J})$ without putting any restriction on the u_i's. But, in fact, to remove the constraint that the letters must be all distinct in each of the u_i's, we simply have to decorate each of the input letters with its position minus 1 modulo a big enough $d \in \mathbb{N}_{>0}$. This finally leads to the following proposition.

Proposition 7. *Let Σ be an alphabet, $l \in \mathbb{N}_{>0}$ and $u_1, \ldots, u_k \in \Sigma^+$ ($k \in \mathbb{N}_{>0}$). Then $[u_1, \ldots, u_k]_l \in \mathcal{P}(\mathbf{J})$.*

This finishes to prove Theorem 1 by closure of $\mathcal{P}(\mathbf{J})$ under Boolean combinations (Proposition 1) and by the discussion at the beginning of Subsect. 4.1.

5 Conclusion

Although $\mathcal{P}(\mathbf{J})$ is very small compared to AC^0, we have shown that programs over monoids in **J** are an interesting subject of study in that they allow to do quite unexpected things. The "feedback-sweeping" technique allows one to detect presence of a factor thanks to such programs as long as this factor does not appear too often as a subword: this is the basic principle behind threshold dot-depth one languages, that our article shows to belong wholly to $\mathcal{P}(\mathbf{J})$.

Whether threshold dot-depth one languages with additional positional modular counting do correspond exactly to the languages in $\mathcal{L}(\mathbf{QDA}) \cap \mathcal{L}(\mathbf{Q}(\mathbf{J} * \mathbf{D}))$ seems to be a challenging question, that we leave open. In his Ph.D. thesis [7], the author proved that all strongly unambiguous monomials (the basic building blocks in $\mathcal{L}(\mathbf{DA})$) that are imposed to belong to $\mathcal{L}(\mathbf{J} * \mathbf{D})$ at the same time are in fact threshold dot-depth one languages. However, the proof looks much too complex and technical to be extended to, say, all languages in $\mathcal{L}(\mathbf{DA}) \cap \mathcal{L}(\mathbf{J} * \mathbf{D})$. New techniques are probably needed, and we might conclude by saying that proving (or disproving) this conjecture could be a nice research goal in algebraic automata theory.

Acknowledgements. The author thanks the anonymous referees for their helpful comments and suggestions.

References

1. Ajtai, M.: Σ_1^1-formulae on finite structures. Ann. Pure Appl. Logic **24**(1), 1–48 (1983)
2. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. J. Comput. Syst. Sci. **38**(1), 150–164 (1989)
3. Barrington, D.A.M., Thérien, D.: Finite monoids and the fine structure of NC^1. J. ACM **35**(4), 941–952 (1988)
4. Eilenberg, S.: Automata, Languages, and Machines, vol. A. Academic Press, New York (1974)
5. Eilenberg, S.: Automata, Languages, and Machines, vol. B. Academic Press, New York (1976)
6. Furst, M.L., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Math. Syst. Theory **17**(1), 13–27 (1984)
7. Grosshans, N.: The limits of Nečiporuk's method and the power of programs over monoids taken from small varieties of finite monoids. Ph.D. thesis, University of Paris-Saclay, France (2018)
8. Grosshans, N., McKenzie, P., Segoufin, L.: The power of programs over monoids in DA. In: MFCS 2017, Aalborg, Denmark, 21–25 August 2017, pp. 2:1–2:20 (2017)
9. Klíma, O., Polák, L.: Hierarchies of piecewise testable languages. Int. J. Found. Comput. Sci. **21**(4), 517–533 (2010)
10. Lautemann, C., Tesson, P., Thérien, D.: An algebraic point of view on the Crane Beach property. In: Ésik, Z. (ed.) CSL 2006. LNCS, vol. 4207, pp. 426–440. Springer, Heidelberg (2006). https://doi.org/10.1007/11874683_28
11. Maciel, A., Péladeau, P., Thérien, D.: Programs over semigroups of dot-depth one. Theor. Comput. Sci. **245**(1), 135–148 (2000)
12. McKenzie, P., Péladeau, P., Thérien, D.: NC^1: the automata-theoretic viewpoint. Comput. Complex. **1**, 330–359 (1991)
13. Péladeau, P.: Classes de circuits booléens et variétés de monoïdes. Ph.D. thesis, Université Pierre-et-Marie-Curie (Paris-VI), Paris, France (1990)
14. Péladeau, P., Straubing, H., Thérien, D.: Finite semigroup varieties defined by programs. Theor. Comput. Sci. **180**(1–2), 325–339 (1997)
15. Pin, J.: The dot-depth hierarchy, 45 years later. In: The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski, pp. 177–202 (2017)
16. Pin, J.: Varieties of Formal Languages. Plenum Publishing Co., New York (1986)
17. Pin, J., Straubing, H.: Some results on $\mathcal{C}$-varieties. ITA **39**(1), 239–262 (2005)
18. Simon, I.: Piecewise testable events. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 214–222. Springer, Heidelberg (1975). https://doi.org/10.1007/3-540-07407-4_23
19. Straubing, H.: Finite semigroup varieties of the form $V * D$. J. Pure Appl. Algebra **36**, 53–94 (1985)
20. Straubing, H.: When can one finite monoid simulate another? In: Birget, J.C., Margolis, S., Meakin, J., Sapir, M. (eds.) Algorithmic Problems in Groups and Semigroups, pp. 267–288. Springer, Boston (2000). https://doi.org/10.1007/978-1-4612-1388-8_15
21. Straubing, H.: Languages defined with modular counting quantifiers. Inf. Comput. **166**(2), 112–132 (2001)

22. Tesson, P.: Computational complexity questions related to finite monoids and semigroups. Ph.D. thesis, McGill University, Montreal (2003)
23. Tesson, P., Thérien, D.: The computing power of programs over finite monoids. J. Autom. Lang. Comb. **7**(2), 247–258 (2001)