Electronic Colloquium on Computational Complexity, Report No. 5 (2001)

The Computing Power of Programs over Finite Monoids

Pascal Tesson
McGill University

Denis Thérien*
McGill University

Abstract

The formalism of programs over monoids has been studied for its close connection to parallel complexity classes defined by small-depth boolean circuits. We investigate two basic questions about this model. When is a monoid rich enough that it can recognize arbitrary languages (provided no restriction on length is imposed)? When is a monoid weak enough that all its computations can be realized in polynomial length? Surprisingly, these two properties appear to be dual to each other.

1 Introduction

Finite monoids can be used as language recognizers in many different ways. Classically, one would use a morphism $\phi : A^* \to M$ and a subset $F \subseteq M$ to recognize the language $L = \phi^{-1}(F) \subseteq A^*$. It is well-known that this framework characterizes the class of regular languages and the algebraic point of view provides a most powerful set of tools to understand and classify the combinatorial properties of such languages (see [Eil76] and [Pin86] for a detailed description of this approach). In this model, the morphism can be seen as a very uniform way to translate a string $a_1 \dots a_n$ in A^* to a string $\phi(a_1) \dots \phi(a_n)$ of monoid elements which is then evaluated in the monoid to yield the value of the "machine" M on its input.

In [Bar89] and [BT88], a more general device to transform a string in A^* into a string of monoid elements was introduced. An n-input M-program takes as input a word of length n over the alphabet A and it is allowed to query the positions in arbitrary order and each position

*School of Computer Science, McGill University. 3480 University, Montréal, Québec H3A 2A7 Canada. Email: {ptesso,denis}@cs.mcgill.ca. Supported by NSERC and FCAR. This work was completed while the second author was in Germany, supported by the von Humboldt Foundation.

ISSN 1433-8092

can be queried several times. At each query, the letter read in the given position is transformed to a monoid element (precise definition is given in the next section). In this way, the input word $w = a_1 \dots a_n$ gives rise to a string $m_1 \dots m_s$, which again is multiplied out in the monoid to yield the value of the program on w. One recognizes a subset of A^* by considering a sequence of n-input programs, one for each input length. The interest of this model comes from its close relationship to parallel complexity classes defined by boolean circuits of small depth: the circuit classes NC^1, ACC^0, CC^0, AC^0 can all be characterized algebraically by this approach, and many central open questions about the power of circuits can be rephrased in purely algebraic terms: for example, the conjecture that ACC^0 is strictly contained in NC^1 is equivalent to the conjecture that a polynomial-length program over a solvable monoid cannot simulate the multiplication of a non-solvable group.

The combinatorial power of monoids in this new framework is far from being understood: it is of course quite a bit more subtle than in the case of morphisms, largely because the monoid is allowed to "look" at its input many times. In this paper, we will study and compare two natural properties relevant to the model. When is a monoid complicated enough to allow recognition of arbitrary languages via programs (with no restriction on length)? When is a monoid simple enough that any program over it can be replaced by an equivalent program of polynomial length? Our results lead to the surprising conjecture that a monoid M is not universal iff it has this polynomial length property. We prove that the conjecture holds for groups and provide several partial results for aperiodic monoids. We offer a suggestion as to the potential algebraic description of the general case.

2 Definitions

2.1 Finite Monoids

A monoid is a set with a binary associative operation and an identity element. Throughout this paper, M, N will denote finite monoids.

The wreath product $M \circ N$ is the set $M^N \times N$ with an operation defined as

$$(f_1, n_1) \cdot (f_2, n_2) = (f_1 \cdot f_2^{n_1}, n_1 n_2)$$

where $f_2^{n_1}(x) = f_2(xn_1)$, and $\cdot$ is the operation in M.

The monoid M is a group iff it satisfies an equation of the form $x^q = 1$ for some $q \geq 1$. We say that M is aperiodic if no subset of it forms a non-trivial group or, equivalently, if it satisfies the equation $m^{t+1} = m^t$ for some $t \geq 0$. Any finite monoid can be decomposed into a wreath product of group and aperiodic components. The following three aperiodic monoids will bear special importance in our discussion. First, U_1 is the aperiodic monoid with two elements 1 and 0 and multiplication given by $0 \cdot 1 = 1 \cdot 0 = 0 \cdot 0 = 0$ and $1 \cdot 1 = 1$.

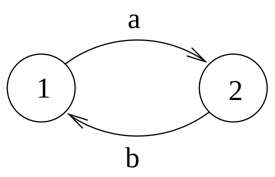

Fig. 1: BA_2 automaton

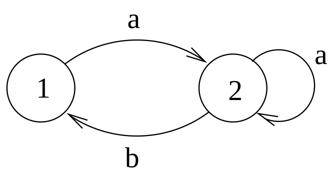

Fig. 2: U automaton

Next, BA_2 and U are the transformation monoids associated with the partial automata of Figures 1 and 2 respectively. Note that missing transitions lead to an implicit sink state. Both monoids hold the six elements $\{1, a, b, ab, ba, 0\}$ but have slightly different multiplication tables which we give here partially:

	a	b	ab	ba
a	0	ab	0	a
b	ba	0	b	0
ab	a	0	ab	0
ba	0	b	0	ba

BA_2's mult. table

	a	b	ab	ba
a	a	ab	ab	a
b	ba	0	b	0
ab	a	0	ab	0
ba	ba	b	b	ba

U's mult. table

The most basic tools of semigroup theory are Green's relations. We define on M four equivalence relations:

- $x\mathcal{R}y$ iff x and y generate the same right ideal, i.e. $xM = yM$.
- $x\mathcal{L}y$ iff x and y generate the same left ideal, i.e. $Mx = My$.
- $x\mathcal{J}y$ iff x and y generate the same two-sided ideal, i.e. $MxM = MyM$.
- $x\mathcal{H}y$ iff $x\mathcal{R}y$ and $x\mathcal{L}y$.

It is known that $\mathcal{L}$ and $\mathcal{R}$ commute. Moreover, when M is finite, we have $\mathcal{R} \circ \mathcal{L} = \mathcal{L} \circ \mathcal{R} = \mathcal{J}$. We will denote by $\mathcal{R}_a$ the $\mathcal{R}$-class of a. An element $e \in M$ is *idempotent* if $e^2 = e$. The next lemma shows that idempotents play a special role in the algebraic structure of M.

Lemma 1 *Let $a\mathcal{J}b$. Then $ab \in \mathcal{R}_a \cap \mathcal{L}_b$ iff $\mathcal{L}_a \cap \mathcal{R}_b$ contains an idempotent.*

Moreover if e is idempotent and $e\mathcal{R}a$ (resp. $e\mathcal{L}a$) then $ea = a$. (resp. $ae = a$)

Two monoid elements $x, y \in M$ are said to be inverse if $xyx = x$ and $yxy = y$. A monoid is inverse if all its elements have a unique inverse. As an example, BA_2 is inverse, whereas U is not since a admits both itself and b as inverses.

We say that N divides M (denoted $N \prec M$) if it is a morphic image of a submonoid of M. A class $\mathcal{C}$ of finite monoids is a (pseudo)-variety if it is closed under direct product and division. The class of finite groups (denoted $\mathbf{G}$) and of finite aperiodic monoids both define varieties. We will also be referring to the following varieties (always denoted in **bold**):

For any prime p, $\mathbf{G_p} = p$-groups $= \{M : \text{ exists } k \text{ s.t. } g^{p^k} = 1\}$

$\mathbf{J_1} = \{M : s^2 = s, mn = nm\}$

$\mathbf{DA} = \{M : \text{ there exists } k \text{ s.t. } (stu)^k t (stu)^k = (stu)^k\}$

If $\mathbf{V}$ and $\mathbf{W}$ are two varieties, we denote by $\mathbf{V} * \mathbf{W}$ the variety generated by all monoids $M \circ N$ for $M \in \mathbf{V}$ and $N \in \mathbf{W}$.

2.2 Programs, Universality and the PLP

Let A be a finite alphabet. An n-input M-program of length s over input alphabet A consists of a sequence of instructions

$$\phi = (i_1, f_1) \dots (i_s, f_s)$$

where, for each j, $1 \leq i_j \leq n$ and $f_j : A \to M$, and an accepting subset $F \subseteq M$. On an input word $x = x_1 \dots x_n \in A^n$, the program produces the monoid element $\phi(x) = f_1(x_{i_1}) \dots f_s(x_{i_s})$: the word is accepted iff $\phi(x)$ is in F and the language $L(\phi) \subseteq A^n$ is the set of all words accepted by the program.

We say that a monoid M is *universal* if, for each n and for each subset $L \subseteq A^n$, there exists an n-input M-program ϕ such that $L(\phi) = L$. Examples of universal monoids will be given later. We first observe the following:

Lemma 2 *The class of non-universal monoids is closed under division*

Proof. Let N be a non-universal monoid. Let first M be a submonoid of N. Since any M-program is also an N-program, M cannot be universal either. Let next $M = \theta(N)$ for some morphism θ and fix for each $m \in M$ an arbitrary preimage n_m in N. Let $\phi = (i_1, f_1) \dots (i_s, f_s)$ be an n-input M-program with accepting subset F. Construct the N-program $\psi = (i_1, g_1) \dots (i_s, g_s)$, where $g_j : A \to N$ satisfies $g_j(a) = n_{f_j(a)}$ for each letter a, and the accepting subset $G \subseteq N$ is declared to be $\theta^{-1}(F)$. For any word x, we then have $\phi(x)$ is in F iff $\psi(x)$ is in G, and thus $L(\phi) = L(\psi)$: this implies that M

cannot be universal. ∎

On the other hand, we are unable to prove yet that the class of non-universal monoids is closed under direct product, i.e. it is not known if this class forms a variety.

We say that a monoid M has the *polynomial length property* (abbreviated PLP) if there is a constant k such that for each n and for every n-input M-program ϕ, there exists an equivalent n-input M-program ψ of length n^k. As a first example, any commutative monoid has the PLP. Indeed, commutativity allows us to permute the order of the instructions at will and consecutive instructions querying the same position can be coalesced into a single one: hence, any n-input M-program is equivalent to an M-program of length n. It is an easy exercise to establish the following closure property.

Lemma 3 *If M and N have the PLP, then so does $M \times N$.*

On the other hand, it is not clear if the PLP is preserved by taking submonoids or morphic images. Let N have the PLP and let M be a submonoid of N: an M-program is also an N-program and this N-program can be reduced to an equivalent one of polynomial length, but this new N-program may involve in its instructions elements which are outside of M, and hence may not be an M-program. Let next $M = \theta(N)$: for any M-program ϕ, we have seen that we can construct an N-program which recognizes the same set, and this N-program can be reduced to an equivalent one of polynomial length, say ψ. We are unfortunately not guaranteed that the accepting subset of ψ is the preimage of some subset of M, i.e. it may be that $\theta(n_1) = \theta(n_2)$ where n_1 is accepting and n_2 is rejecting: hence, there is no clear way of transforming ψ into an M-program.

In particular, there is a polynomial length program computing the AND function over the group $S_3 \times A_5$, but any program computing AND over the subgroup S_3 has exponential length [BST90]. This does not ruin the possibility that the polynomial length property is preserved under division, as PLP is provably false for $S_3 \times A_5$, but the example shows that an argument to prove the closure property will crucially depend on PLP holding for the larger monoid.

How do the two notions introduced compare? It is easy to see that when M has the PLP, it cannot be universal. This follows since there are doubly exponentially many subsets of A^n, whereas we can construct only singly exponentially many M-programs. This paper wishes to offer, and argue in favor of, the following surprising conjecture.

Conjecture 4 *A monoid M has the polynomial length property iff it is not universal*

The validity of the conjecture would in particular imply that the class comprising those monoids forms a variety. We will prove that the statement holds for groups, and that the property characterizes the well-known class of nilpotent groups. The picture is not complete in the general case but we are able to prove the following:

- any monoid in the variety **DA** has the PLP
- any monoid outside the variety $\mathbf{DA} * \mathbf{G}$ is universal.

Moreover, we have examples of monoids which we can prove are not universal but for which we cannot prove the PLP (although each of them divides a monoid having the PLP). All this evidence enables us to propose a candidate for the exact variety of monoids which correspond to the class defined by the conjecture, namely we suggest that the right answer could be those monoids which divide the wreath product of a monoid in **DA** and a group but do not have any non-nilpotent subgroup. Note that this is *not* the same as $\mathbf{DA} * \mathbf{G_{nil}}$.

3 The case of groups

In this section we will prove that the conjecture holds for the restricted case of groups. This can be seen by putting together results that have implicitly appeared earlier in [BST90], but we present here an alternative proof.

Let G be a group and g, h be elements of G: the group element $g^{-1}h^{-1}gh$ is said to be a commutator of weight 2 and is denoted by $[g, h]$. More generally, all elements of the group will be said to be commutators of weight 1, and commutators of weight k will be those elements which can be expressed as $[g, h]$, where g and h are commutators for which the sum of the weights is at least k. G is nilpotent of class k iff the only commutator of weight $k+1$ is the identity, G is nilpotent iff it is nilpotent of class k for some k.

Lemma 5 *Let G be an arbitrary group and let G_c be the subgroup generated by all commutators of weight c. Then any function $f : A^c \to G_c$ can be realized by a G-program of length $(d_G)^c$, where d_G depends on G.*

Proof. The proof is by induction on c. If $c = 1$, the program is simply $\phi_f = (1, f)$. For $c > 1$, let g be a commutator of weight c_1 and h be a commutator of weight c_2, where $c_1 + c_2 = c$. By induction hypothesis, for any fixed x in A^{c_1}, there is a program $\phi_{x,g}$ that takes the value g on input x and the value identity on any input different from x. Similarly, for any fixed y in A^{c_2}, there is a program $\phi_{y,h}$ that takes the value h on input y and the value identity on any other input. Such programs also exist for g^{-1} and h^{-1} since these are also in G_{c_1} and

G_{c_2} respectively. Let now z be in A^c, with $z = xy$, where x is in A^{c_1} and y is in A^{c_2}: we construct $\phi_{z,[g,h]} = \phi_{x,g^{-1}}\phi_{y,h^{-1}}\phi_{x,g}\phi_{y,h}$, where it is understood that the first and third segments query the prefix of length c_1 of the input while the second and fourth segments query the suffix of length c_2. This program is easily seen to have the property that it yields $[g,h]$ on input z and the identity element on any other input. We finally get the desired program ϕ_f as the concatenation of the various $\phi_{z,f(z)}$, for all z in A^c. Note that the program has length exponential in c. ∎

Lemma 6 *If G is nilpotent, it has the polynomial length property.*

Proof. Earlier, we argued that commutative monoids have the PLP using the fact that blocks of instructions could be permuted at will in the program. This is obviously not possible in general but nilpotent groups allow a similar construction. Formally, let G be nilpotent of class k and ϕ be an n-input G-program. For any function $f : A \to G$, denote by f^{-1} the function defined by $f^{-1}(a) = (f(a))^{-1}$ for every a. Similarly, for any G-program $\phi = (i_1, f_1)\dots(i_s, f_s)$, we define $\phi^{-1} = (i_s, f_s^{-1})\dots(i_1, f_1^{-1})$. Note that for all x, $\phi(x)\cdot\phi^{-1}(x) = 1$. Note that just as $gh = hg[g,h]$, for any two program segments ϕ_1, ϕ_2 we have:

$$\phi_1\phi_2 = \phi_2\phi_1[\phi_1,\phi_2] = \phi_2\phi_1\phi_1^{-1}\phi_2^{-1}\phi_1\phi_2 \tag{1}$$

We will say that a program segment ϕ_1 is a t-block if it depends on at most t variables and always has output in G_t. In particular, any single instruction can be viewed as a 1-block and if ϕ_1, ϕ_2 are t_1- and t_2-blocks respectively, $[\phi_1,\phi_2]$ is a t-block for some $t \geq t_1 + t_2$. Because G has nilpotency class k, any t-block with $t > k$ outputs the identity and can thus be deleted from the program.

Let $I_1, \dots, I_m$ be some enumeration of the subsets of $\{1, \dots, n\}$ of cardinality $\leq k$, such that the sequence of cardinalities of the I_j is non-decreasing. Note that $m = O(n^k)$. We claim that ϕ is equivalent to a program $\phi = \psi_1\psi_2\dots\psi_m$ where the segment ψ_j is a product of $|I_j|$-blocks depending only on variables in I_j. Observe that we can achieve this by repeatedly using (1): we can push any misplaced block further left by introducing a new block of strictly heavier weight on the right. This process terminates since we can delete the $(k+1)$-blocks.

Now, the segment ψ_j is, by definition, computing a function from $A^{|I_j|}$ into $G_{|I_j|}$ so by Lemma 5 it can be replaced by an equivalent program of length bounded by a constant $(d_G)^{|I_j|} \leq (d_G)^k$. ∎

Lemma 7 *If G is not nilpotent, then it is universal.*

Proof. If G is not nilpotent, then for every n there is a commutator $h \in G_n$ that is not the identity. Fix such an h arbitrarily. Let L be an arbitrary subset of A^n. Let $f : A^n \to G_n$ be defined by $f(w) = h$ if $w \in L$ and $f(w) = 1$ otherwise. By Lemma 5, f can be realized by a G-program and so L is G-recognized. ∎

The two lemmas combine into:

Theorem 8 *If G is a group, then G has the polynomial length property iff it is not universal.*

4 The General Case

In this section, we study the general case, first focusing on aperiodic monoids. Although we are not able to settle completely the main conjecture, several interesting partial results will be proved. It turns out that, as is often the case, the aperiodic variety **DA** is a convenient case to deal with. We will use the following two results about **DA**.

Lemma 9 ([Sch76]) *Let M be a monoid in* **DA** *and consider the natural evaluation morphism $Eval : M^* \to M$. Then for any $F \subseteq M$, the language $Eval^{-1}(F)$ can be expressed as a finite disjoint union of languages of the form $M_0^* m_1 M_1^* \ldots m_t M_t^*$, where each M_i is a subset of M and the concatenation is unambiguous (i.e. any given word w in M^* may have at most one factorization as $w = w_0 m_1 w_1 \ldots m_t w_t$ with each w_i in M_i^*.)*

Lemma 10 ([BMT92]) *If M is an aperiodic monoid not in* **DA**, *then M is divided either by U or by BA_2.*

It is now possible to show the following:

Theorem 11 *If M is in* **DA**, *then M has the polynomial length property.*

Proof. Consider an n-input M-program ϕ with accepting subset F. Suppose w is accepted: by Lemma 9 the sequence of monoid elements produced by ϕ spells out a word $\phi(w) = w_0 m_1 w_1 \ldots m_t w_t$ in $L = M_0^* m_1 M_1^* \ldots m_t M_t^* \subseteq Eval^{-1}(F) \subseteq M^*$. Let us refer to these occurrences of the m_i's in $\phi(w)$ as the *bookmarks* of $\phi(w)$. The t instructions producing these bookmarks together query a set I of at most t positions in the input word. The key observation is that whenever another word y agrees with w on the positions I and is such that $\phi(y)$ is also in L, the t bookmarks of $\phi(y)$ must be produced by exactly the same instructions as those producing the bookmarks of $\phi(w)$: to see this, construct the program ψ by deleting all instructions of ϕ that query positions outside of I. It is clear that $\psi(w) = \psi(y) \in L$. Moreover, this word can be factorized using either the bookmarks used for $\phi(w)$ or for $\phi(y)$, but the unambiguity condition then forces these bookmarks to be at the same location. This observation implies that only polynomially many instructions, say $O(n^r)$ can ever be responsible for producing some bookmark in an accepted word. Construct a new program by taking these instructions, plus, for each pair of consecutive instructions in this sequence, exactly one occurrence of each instruction that appears in the segment separating the two bookmark-producing instructions in the original program. Since there are only $O(n)$ possible instructions, this will produce a program of length $O(n^{r+1})$. We claim that the new program is equivalent to the original one. Any word accepted by ϕ is still accepted, since all bookmark-producing instructions have been preserved, and the new program is obtained from the old one by deleting instructions. Conversely, any word accepted by the new program must have its output factorized using some unique set of bookmarks: the same valid bookmarks exist in the original program since all instructions appearing between any two consecutive ones also appear in the reduced program. ∎

We next investigate what happens for aperiodics outside of **DA**. The following result, originally proved in [Thé89], deals with part of this situation.

Theorem 12 *U is universal.*

Proof. Fix some word $w \in A^n$ and consider the program $\phi = (1, f_1) \ldots (n, f_n)$, where $f_i(c) = b$ if $c = w_i$ and $f_i(c) = e$ otherwise. Then $\phi(x)$ fixes state 1 in the automaton for U iff $x = w$, otherwise $\phi(x)$ sends state 1 to state 2. Concatenating programs of this form for each word in the language to be recognized, with in-between instructions that produce b, we get a program such that $\phi(x)$ sends state 1 to the sink state iff x belongs to the desired language. ∎

This implies that any monoid divided by U is universal. When restricted to aperiodic monoids, our conjecture states that the converse is also true. Aperiodic monoids outside of **DA** that are not divided by U include, in particular, all aperiodic inverse monoids that are not in **DA**, BA_2 being a prime example. We can show that BA_2 is not universal, but we are so far unable to prove that it has the PLP.

Lemma 13 *The monoid BA_2 cannot recognize the language PARITY.*

Proof. A proof of this fact was first given in [Thé89] using representation of languages by polynomials and it can also be obtained as a corollary to theorem 14. It is unlikely that these arguments will generalize to more complicated cases and we provide here a new proof whose ideas could help resolve the case of all inverse aperiodic monoids.

Suppose that the n-input BA_2-program ϕ is computing PARITY for some $n \geq 5$. W.L.O.G we can assume[1] that $\phi(0^n) = 0$ and that every instruction produces one of $\{1, a, b\}$. Note that a string in $\{1, a, b\}^*$ evaluates to 0 in BA_2 iff it contains either two a's not separated by a b or two b's not separated by an a. Thus if $\phi(0^n) = 0$, we can find instructions j, k querying (not necessarily distinct) bits i_j, i_k and producing two a's (or b's) while all instructions in between them output the identity.

Let ψ be the subprogram of ϕ consisting of instructions

$$(i_{j+1}, f_{j+1}) \dots (i_{k-1}, f_{k-1})$$

. Since ϕ is computing PARITY, $\phi(w)$ cannot be 0 for any w of weight 1. Hence, if a bit other than i_j or i_k is turned on, the number of instructions of ψ that now produce a b exceeds the number of those now producing an a by 1 in order to separate the two a's still output by instructions j and k. Thus, for an input of weight three, ψ outputs a string with 3 more b's than a's so it must evaluate to 0 in BA_2, a contradiction. ∎

As we mentioned earlier, it is unclear whether the PLP is preserved under the formation of submonoids or morphic images. In particular, it was observed in [Thé89] that for all $m \in \mathbf{N}$, BA_2 is a divisor of $U_1 \circ C_m$, where C_m denotes the cyclic group of order m. It is puzzling to note that, as we will show next, the latter has the PLP whenever m is prime, even though we can not restrict the arguments to the case of BA_2.

Theorem 14 *Let $M = (U_1)^k \circ G$, where $(U_1)^k$ is a k-fold direct product of U_1 and G is a p-group. Then M has the PLP.*

Proof. For simplicity, we will assume that $M = U_1 \circ G$ and that the input alphabet[2] is $\{0, 1\}$.

By results of [BST90], we know that for any n-input program ψ over a p-group G, there exists a polynomial r in $Z_p[X_1, \dots X_n]$ of degree at most d_G such that $r = 1$ whenever ψ outputs $h \in F \subseteq G$ and $r = 0$ otherwise.

[1]In fact, if 0 is not in the image of ϕ, we have to appeal to an easy variation of our argument

[2]This a priori seems to be a strong restriction. In particular, representing G_p-programs as bounded-degree polynomials in Z_p is more tricky. We include this construction in the appendix for completeness.

Let us denote by the pair $(\overline{f}_i, \overline{g}_i) \in (U_1^G, G)$ the product of the first i instructions of ϕ on some input. If (f_i, g_i) is the result of the i^{th} instruction, we have

$$(\overline{f}_i, \overline{g}_i) = (\overline{f}_{i-1} f_i^{\overline{g}_{i-1}}, \overline{g}_{i-1} g_i)$$

We will say that instruction i is an "h-crash site" for $x \in \{0,1\}^n$ if on x we have $\overline{f}_{i-1}(h) = 1$ but $\overline{f}_i(h) = 0$. In particular, this implies that $f_i^{\overline{g}_{i-1}}(h) = 0$.

Our main claim is that for all h in G there can only be polynomially h-crash sites querying bit X_1. Consider in particular all such instructions such that a crash occurs on an input where $X_1 = 1$. Since the value $\overline{g}_{i-1}$ is computed by a G program, we know by preceding remarks that there exists a fixed degree Z_p polynomial r_i associated with this instruction such that $f_i^{\overline{g}_{i-1}}(h) = 0$ if and only if $r_i = 1$. There are only $\binom{n}{d_G}$ many linearly independent such r's, hence if we have more than $\binom{n}{d_G}$ crash sites it must be the case that some r_i can be expressed as a linear combination of r_j's with $j < i$. Hence if $r_i = 1$, there must be $j < i$ with $r_j = 1$. This shows that i is actually not a crash site since whenever $X_1 = 1$ and $f_i^{\overline{g}_{i-1}}(h) = 0$ we already had $\overline{f}_j(h) = 0$.

Therefore, we have at most $|G| \cdot 2 \cdot n \cdot \binom{n}{d_G}$ instructions which are crash sites, i.e. where the U_1^G part of the computation is truly active. For all but polynomially many instructions in ϕ, we can thus replace the U_1^G component of the instruction by the identity without affecting the result of our computation. In between any two potential crash sites, we are thus left with subprograms over the subgroup G but these can be made to have polynomially bounded length using Lemma 6. ∎

It is known that a monoid M belongs to the variety $\mathbf{J}_1$ if and only if it divides the direct product of k copies of U_1. One corollary of our theorem is that any monoid dividing some $(U_1)^k \circ G$ with $G \in \mathbf{G_p}$ is non-universal. It had in fact already been shown in [Thé89] that programs over $M \in \mathbf{J_1} \circ \mathbf{G_p}$ can not recognize the function MOD$_q$ for any primes $p \neq q$.

Finally, we show that any monoid which does not divide the wreath product of a monoid in **DA** and a group is universal.

Lemma 15 *If M does not belong to the variety $(\mathbf{DA} * \mathbf{G})$ then M is universal.*

Proof. It has been proved in [ST] that if $M \notin (\mathbf{DA} * \mathbf{G})$ then there exist two idempotents $e, f \in M$ such that $e\mathcal{J}f\mathcal{J}(ef)$ but ef is not idempotent. Lemma 1 insures the presence of an idempotent s in the $\mathcal{H}$-class $\mathcal{L}_e \cap \mathcal{R}_f$.

Suppose first that there is no idempotent in $\mathcal{H}_{ef} = \mathcal{R}_e \cap \mathcal{L}_f$. Then it is easy to verify that there is a surjective morphism from the submonoid generated by $\{1, e, f, s\}$ into U. Since U is universal, M must also be.

Suppose now that there exists an idempotent $d\mathcal{H}(ef)$. Note that Lemma 1 can be used to show that $ed = d$, $de = e$ and $df = d$. Also, since $efef \neq ef$ we have $efe \neq e$.

Let L be an arbitrary subset of A^n. Fix a word $w \in L$ and consider the program $\phi = e \cdot (1, f_1) \ldots (n, f_n) \cdot fe$ where, for any $c \in A$, $f_i(c) = 1$ if $c = w_i$ and $f_i(c) = d$ otherwise. For any $x \neq w$, at least one instruction outputs a d and, since $d^2 = d$, $\phi(x) = edfe = e$. On the other hand $\phi(w) = efe$. Concatenating such programs for all elements of L, we get a program ψ with the property that $\psi(x) = e$ for $x \notin L$. On the other hand, if x does belong to L then exactly one of the segments will output efe and so we will get $\psi(x) = efe \neq e$. ∎

5 Conclusion

As we mentioned in the introduction, we conjecture that a finite monoid is non-universal if and only if it has the PLP if and only if it belongs to the variety $(\mathbf{DA} * \mathbf{G}) \cap \mathbf{M_{nil}}$, where $\mathbf{M_{nil}}$ denotes the variety of monoids in which every subgroup is nilpotent. We have shown in the paper that PLP implies non-universality which in turn implies membership in $(\mathbf{DA} * \mathbf{G}) \cap \mathbf{M_{nil}}$ by Lemmas 7 and 15. This variety is closely linked to the one studied in a descriptive complexity context in [ST].

We have more or less implicitly indicated some possible steps to take towards a proof of our main conjecture. Theorem 14 in particular raises many questions. For one, it is reasonable to hope that similar arguments could show that a monoid of the form $(U_1)^k \circ G$ has the PLP if G is nilpotent. Indeed, nilpotent groups are the direct product of p-groups, the case covered by our theorem, but even the apparently simple case of $U_1 \circ C_6$ has so far eluded proof. It is not even known whether this monoid is non-universal.

As we mentioned, Theorem 14 does imply non-universality for the whole variety $\mathbf{J_1} \circ \mathbf{G_p}$, but not the PLP. An important step would be to show that BA_2 for instance does have the PLP but it is unclear how or even if the proof of our theorem can help in achieving such a goal.

Finally, a slightly different line of progress could be the case of inverse aperiodic monoids. All such monoids lie in $\mathbf{DA} * \mathbf{G}$ but only those in $\mathbf{J_1} * \mathbf{G_p}$ are known to be non-universal. The proof of Lemma 13, on the other hand, argues that if the program leads one string to the automaton's sink state, it leads almost all input strings to the sink state. Similar phenomena seem to occur in all inverse aperiodic monoids and this intuition could help in proving non-universality for this class.

We wish to thank Peter Kadau of the University of Tübingen for his helpful comments and suggestions concerning this paper.

References

[Bar89] David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *Journal of Computer and System Sciences*, (1):150–164, 1989.

[BMT92] M. Beaudry, P. McKenzie, and D. Thérien. The membership problem in aperiodic transformation monoids. *Journal of the ACM*, 39(3):599–616, 1992.

[BST90] David A. Mix Barrington, Howard Straubing, and Denis Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, December 1990.

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

[Eil76] S. Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.

[Pin86] J.-E. Pin. *Varieties of formal languages.* North Oxford Academic Publishers Ltd, London, 1986.

[Sch76] M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, (13):47–75, 1976.

[Smo87] R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19th Annual ACM STOC*, pages 77–82, 1987.

[ST] Howard Straubing and Denis Thérien. Two-variable definability.

[Thé83] Denis Thérien. Subword counting and nilpotent groups. In L.J. Cummings, editor, *Combinatorics on Words: Progress and Perspectives*, pages 195–208. Academic Press, 1983.

[Thé89] Denis Thérien. Programs over aperiodic monoids. *Theoretical Computer Science*, 64(3):271–280, 29 May 1989.

Appendix

The technical arguments given in this appendix are a generalization of the ones presented in [BST90]. Let $A = \{a_1, \ldots, a_s\}$ be some finite alphabet. Generalizing the construction of [Smo87], we will represent subsets of A^n by polynomials over Z_p in the $k \cdot n$ boolean variables $x_1^{a_1}, x_1^{a_2}, \dots, x_1^{a_s}, x_2^{a_1}, \dots, x_n^{a_s}$. The intended meaning of these variables, of course, is that $x_i^{a_j}$ is equal to 1 if the i^{th} letter of the input is a_j and is 0 otherwise. For this reason we will in fact be working over the semi-ring $Z_p[x_1^{a_1}, \dots, x_n^{a_s}]$ modulo the identities $(x_i^{a_j})^2 = x_i^{a_j}$ for all i, j and $x_i^{a_j} \cdot x_i^{a_l} = 0$ for all i and all $j \neq l$.

We say that $L \subseteq A^n$ is *recognized* by the polynomial r if for all $x \in A^n$, $r(x) = \chi_L(x)$.

Lemma 16 *Let G be a p-group. For every n-input G-program ϕ over the alphabet A and any accepting subset $F \subseteq G$, the language $L(\phi)$ is recognized over Z_p by a polynomial of degree at most d_G.*

Proof.
We will use the following subword characterization of languages recognized by p-groups presented in [Thé83]. Let G be a p-group of order p^k for some prime p. For strings $w, u \in G^*$, we say that u occurs as a subword of w if there is a sequence of indices $1 \leq i_1 \leq \dots \leq i_{|u|} \leq |w|$ with $u = w_{i_1} w_{i_2} \dots w_{i_{|u|}}$ and we define $\binom{w}{u}$ as the number of occurrences of u as a subword of w. Then the set $\{w \in G^* : w_1 \cdot w_2 \cdot \ldots \cdot w_n = g\}$ is a boolean combination of sets $\{w \in G^* : \binom{w}{u} = j \pmod{p}\}$ for $|u| \leq k$.

Let ϕ be an n-input G-program over the alphabet $A = \{a_1, \dots, a_s\}$. Note that if the polynomials r_1, r_2 recognize $L_1, L_2 \in A^n$ respectively, then $(1 - r_1)$ recognizes $A^n - L_1$ and $r_1 r_2$ recognizes $L_1 \cap L_2$. Thus, in light of the previous remarks, it is sufficient to show that for all $u \in G^k$ and all $0 < i < p - 1$, the set $\{x \in A^n : \binom{\phi(x)}{u} = i \pmod{p}\}$ is recognized over Z_p by a polynomial of bounded degree. To see this, note that for each possible occurrence of u as a subword of $\phi(x)$, there are k instructions and thus at most k input values responsible for its presence or absence. We can thus count the number of occurrences of u modulo p using a polynomial of degree bounded by k. ∎