Programs over semigroups of dot-depth one

Alexis Maciel [1]

Department of Mathematics and Computer Science, Clarkson University, Potsdam, NY 13699-5815, U.S.A.

Pierre Péladeau

Booz Allen & Hamilton Inc., 112 Avenue Kléber, 75116, Paris, France

Denis Thérien [2]

School of Computer Science, McGill University, 3480 University Street, Montréal, Québec, H3A 2A7, Canada

Abstract

The notion of a p-variety arises in the algebraic approach to Boolean circuit complexity. It has great significance, since many known and conjectured lower bounds on circuits are equivalent to the assertion that certain classes of semigroups form p-varieties. In this paper, we prove that semigroups of dot-depth one form a p-variety. This example has the following implication: if a Boolean combination of Σ_1 formulas, using arbitrary numerical predicates, defines a regular language, one can then find an equivalent Σ_1 formula all of whose numerical predicates are regular.

1 Introduction

This paper is a contribution to the ongoing research on the relationships between small-depth families of Boolean circuits, finite semigroups and logical formulas.

[1] Email: alexis@sun.mcs.clarkson.edu. Part of this work was done while this author was at the departments of Mathematics and Computer Science, University of Pittsburgh and at the Department of Computer Science and UMIACS, University of Maryland, College Park. Supported in part by NSF grants CCR–9457782 and CCR–9522084.

[2] Email: denis@cs.mcgill.ca. Supported by NSERC and by FCAR.

In [2], Barrington showed that computations in NC^1, i.e., realized by $O(\log n)$-depth, bounded fan-in Boolean circuit families, could be viewed as taking place over finite monoids. His result was considerably refined in [5] where it was proved that several natural subclasses of NC^1 could be characterized by restricting the algebraic structure of the monoids being considered: for example, the class AC^0 of constant-depth polynomial-size circuits constructed with OR and AND gates of unbounded fan-in corresponds to computations over aperiodic monoids. These theorems can be trivially modified to be phrased in terms of semigroups instead of monoids. More results along those lines were presented in [4] and [18].

The central notion in these investigations is that of a *program over a finite semigroup*. A program, similarly to a morphism, is used to translate an input string from some alphabet A into a sequence of semigroup elements: the difference is that the program may query each input bit several times and produce different semigroup elements each time it does. We require that the number of input queries be limited by a polynomial in the length of the input string. Acceptance or rejection of the input is determined by multiplying out the output sequence of the program in the semigroup. Precise definitions will be given in Section 2.

In [12], McKenzie, Péladeau and Thérien showed that the computing power of programs over a class of semigroups is essentially captured by the regular languages that can be recognized: in fact only word problems for finite semigroups need to be considered. It is thus natural to ask the following question: for which varieties of semigroups is it the case that recognition of word problems via morphisms and recognition of word problems via programs are equivalent? This property defines the notion of a *p-variety* and it has great significance when considered in the context of Boolean circuits. Indeed many known and conjectured lower bounds for circuits are equivalent to the assertion that certain classes of finite semigroups form p-varieties.

On the other hand, from a logical perspective, several classes of constant-depth circuits can be characterized by first-order formulas using arbitrary numerical predicates, where the quantification depends on the depth and on the type of gates used: for example, it is known that such formulas with existential and universal quantifiers correspond exactly to the class AC^0 [10]. Circuit lower bounds are then equivalent to the following statement: if L is a regular language defined by such a formula, then one can also express L by a formula using only regular numerical predicates, with essentially the same quantifier complexity. This is indeed the case for regular languages in AC^0 [3] and this is equivalent to the fact that these circuits cannot determine if the sum of their inputs is divisible by p [8,1]. Also each of these two statements is equivalent to the fact that aperiodic semigroups form a p-variety. For a beautiful exposition of the algebraic and logical theory of regular languages, and its extension to

Boolean circuits, the reader may consult [19].

As is commonly done, we use the same notation for a class of circuits and the family of languages that they recognize. AC^0 stands for the class of constant-depth, polynomial-size circuits constructed with OR and AND gates of unbounded fan-in. We define AC^0_k to consist of fixed-size Boolean combinations of languages recognized by AC^0 circuits of depth exactly k, and we let $\widehat{\mathrm{AC}}^0_k$ be similarly defined, except that input gates are allowed to compute functions that depend on at most t positions of the input, for some fixed integer $t \geq 0$. Our main result deals with the class $\widehat{\mathrm{AC}}^0_1$: we observe that these languages are exactly those recognized by programs over semigroups of dot-depth one, and we prove that this class forms a p-variety. Along the way, we will give a simple direct argument that the hierarchy defined by $\widehat{\mathrm{AC}}^0_k$ interleaves strictly with the one defined by AC^0_k, i.e.,

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

thus refining a result of [16]. This result can also be derived from a special case of a more general theorem in [7]. The fact that semigroups of dot-depth one form a p-variety allows us to generalize a result of Straubing [19] and Péladeau [13] on regular languages expressible by Σ_1 formulas with arbitrary numerical predicates. They showed that one may restrict to regular predicates without losing any regular language: we prove that this remains true for Boolean combinations of Σ_1 formulas.

2 Definitions and basic results

2.1 Semigroups and varieties

A semigroup is a set S equipped with an associative binary operation; a monoid is a semigroup that has a (necessarily unique) two-sided identity element, denoted by 1. We will write $S^\bullet$ for the smallest monoid containing S, i.e., $S^\bullet = S$ if S is a monoid and $S^\bullet = S \cup \{1\}$ otherwise. Any element of S satisfying $e = e^2$ is called an idempotent. When S is finite, for any s in S, we denote by s^ω the unique smallest positive power of s that is an idempotent. We define the following equivalence relation on S: $s\mathcal{J}t$ iff $S^\bullet s S^\bullet = S^\bullet t S^\bullet$; equivalently $s\mathcal{J}t$ iff there exist u, v, x, y in $S^\bullet$ such that $s = utv$ and $t = xsy$. We say that S divides T iff S is a morphic image of a subsemigroup of T; this is clearly a partial order on semigroups. Finally, we define the wreath product $S \circ T$ of two semigroups: this is defined as the set $S^{T^\bullet} \times T$, with the operation $(f_1, t_1)(f_2, t_2) = (f, t_1 t_2)$, where, for any t in $T^\bullet$, $f(t) = f_1(t) f_2(t t_1)$.

Let A^+ (A^*) be the free semigroup (monoid) generated by the alphabet A.

Classically, a semigroup has been viewed as a language recognizer in the following way: S recognizes $L \subseteq A^+$ iff there exist a morphism $\phi : A^+ \to S$ and a subset F of S such that $L = \phi^{-1}(F)$. Equivalently, one may also view ϕ as a morphism from A^+ to S^+; the morphism thus translates an input string x in A^+ into a sequence $\phi(x)$ of semigroup elements and this sequence is multiplied out in S; the resulting value determines if x is in L or not. It is well-known that a subset of A^+ can be described by a regular expression iff it can be recognized by a finite semigroup. The smallest semigroup (in the division ordering) that recognizes L is called the syntactic semigroup of L, denoted by $S(L)$: it can be explicitly defined as $A^+/\equiv_L$ where $x \equiv_L y$ iff for all u, v in A^* uxv is in L iff uyv is in L. We will write $\mathcal{M}_A(S)$ to represent the class of subsets of A^+ that can be recognized, via morphisms, by S; $\mathcal{M}(S)$ will denote the union of these classes over all alphabets A and, for a class $\mathbf{V}$ of semigroups, we will write $\mathcal{M}(\mathbf{V})$ for the union of the $\mathcal{M}(S)$, taken over all S in $\mathbf{V}$.

A natural question is to investigate the relationship between the combinatorial properties of a language and the algebraic structure of its syntactic semigroup. For regular languages, several such connections are known: for example, a language can be denoted by a star-free regular expression (allowing complement as an operator) iff its syntactic semigroup is finite and aperiodic, i.e., for any element s, $s^\omega = s^{\omega+1}$ (see [15]).

The notion of a variety has emerged as the natural unit for classifying finite semigroups in terms of their recognition power. In our context, a variety of semigroups is a class of finite semigroups that is closed under division and finite direct product. For example, the class $\mathbf{A}$ of finite aperiodic semigroups is easily seen to form a variety. Other examples of varieties will be discussed in subsequent sections.

Let S be a finite semigroup and $\eta_S : S^+ \to S$ be the natural morphism: we define the word problems of S, denoted by $\mathcal{W}(S)$, to be the family of languages that are of the form $\eta^{-1}(Q)$ for some $Q \subseteq S$. We have the following

Fact 1 *If* $\mathbf{V}$ *is a variety, then* $\mathcal{W}(S) \subseteq \mathcal{M}(\mathbf{V})$ *iff* $S \in \mathbf{V}$.

2.2 Boolean circuits

An n-input circuit C_n is given by a directed acyclic graph: vertices of fan-in 0 (the input gates) are labeled by elements of $\{0, 1, X_1, \ldots, X_n, \overline{X}_1, \ldots, \overline{X}_n\}$; all other vertices are labeled OR or AND, have arbitrary fan-in, and we assume a unique node of fan-out 0 (the output gate). Such a graph naturally determines a function $C_n : \{0,1\}^n \to \{0,1\}$; given $x \in \{0,1\}^n$, an input gate returns the appropriate constant (if the label is 0 or 1), the i^{th} bit of x (if the label is X_i) or its negation (if the label is $\overline{X}_i$); an inner vertex returns the value of the function by which it is labeled applied to its entries and the value of $C_n(x)$ is the value returned by the output gate.

By considering sequences $C = (C_n)_{n>0}$, where each C_n is an n-input circuit as defined above, we can compute Boolean functions from $\{0,1\}^+$ into $\{0,1\}$. In this paper, no uniformity conditions will be imposed on the sequences (C_n) being used. We will look at computations over arbitrary alphabets by allowing input gates to be labeled "$X_i = a$?" for any letter a. Our interest will be in circuit families of constant depth and polynomial size. More precisely, for each $k \geq 0$, the class AC^0_k consists of the languages that can be expressed as Boolean combinations of sets recognizable by polynomial-size circuits of depth k. We will also define the class $\widehat{\mathrm{AC}}^0_k$ in a similar manner, the only difference being that input gates in each C_n are allowed to be labeled by any function which, for some fixed t, depends on at most t positions of the input string.

2.3 Programs

Let S be a finite semigroup: an n-input S-program over the alphabet A is a sequence $\phi_n = (i_1, f_1) \dots (i_r, f_r)$ where for each j, $1 \leq i_j \leq n$ and $f_j : A \to S$. Such a program computes a function $\phi_n : A^n \to S$ by setting, for any $x = x_1 \dots x_n$ in A^n, $\phi_n(x) = f_1(x_{i_1}) \dots f_r(x_{i_r})$. An S-program is a family $\phi = (\phi_n)_{n>0}$ where each ϕ_n is an n-input S-program: it naturally defines a function $\phi : A^+ \to S$. We say that a language L is *p-recognized* by S if there exist an S-program ϕ and a family of subsets $(F_n)_{n>0}$ of S such that for each n, $L \cap A^n = \phi_n^{-1}(F_n)$. Alternatively we may view $\phi_n(x)$ as a word over the alphabet S, and then we require that, for each n, $L \cap A^n = \phi_n^{-1}(K_n)$ for some K_n in $\mathrm{S}\mathcal{M}(S)$. We write $\mathcal{P}(S)$ for the class of languages recognized by S-programs of polynomial length, and, for any variety $\mathbf{V}$, $\mathcal{P}(\mathbf{V})$ is the union of the $\mathcal{P}(S)$ for all S in $\mathbf{V}$.

Our interest in such families of languages is stimulated by the fact that several natural complexity classes defined by Boolean circuits correspond to classes of the form $\mathcal{P}(\mathbf{V})$. Two significant examples are the correspondence between NC^1 and programs over arbitrary semigroups, and the correspondence between AC^0 and programs over aperiodic semigroups: these connections were originally stated for monoids but the formulation in terms of semigroups is trivially obtained from the other one. Several examples of this circuit-semigroup relationship can be found in [12].

Because of Fact 1 we know that, for any two varieties, $\mathbf{V}$ and $\mathbf{W}$ are distinct iff $\mathcal{M}(\mathbf{V})$ and $\mathcal{M}(\mathbf{W})$ are distinct. In contrast, it is quite possible to have $\mathbf{V} \neq \mathbf{W}$ and yet $\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$. For example, the argument of Barrington [2] concerning the computing power of the class NC^1 can be used to show that

$\mathcal{P}(\mathbf{V}) = \mathcal{P}(\mathbf{W})$ whenever each contains at least one simple non-abelian group.

It follows from Lemma 3.6 in [12] that classes $\mathcal{P}(\mathbf{V})$ (hence the corresponding circuit classes) are characterized by the word problems they contain. This suggests the following definition: a semigroup variety $\mathbf{V}$ is a *p-variety* iff, for any semigroup S, $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$ iff $S \in \mathbf{V}$. Intuitively this says that, for a p-variety $\mathbf{V}$, programs of polynomial length over a semigroup in $\mathbf{V}$ can simulate the multiplication of a semigroup S iff S itself belongs to $\mathbf{V}$. The importance of determining if a class of semigroups is a p-variety comes from the direct impact this has on separating complexity classes defined by Boolean circuits. For example suppose circuit classes $\mathcal{C}$ and $\mathcal{D}$ correspond to $\mathcal{P}(\mathbf{V})$ and $\mathcal{P}(\mathbf{W})$ respectively: then $\mathcal{C} \subseteq \mathcal{D}$ is equivalent to $\mathcal{P}(\mathbf{V}) \subseteq \mathcal{P}(\mathbf{W})$ and knowing that $\mathbf{V}$ is a p-variety implies that the inclusion must be strict, unless $\mathbf{W} = \mathbf{V}$.

2.4 Logical descriptions of languages

Consider a first-order logical system in which formulas consist of variables, usually denoted $i, i_1, i_2, \ldots$, *letter* predicates Q_a, for $a \in A$, *numerical* predicates, the logical connectives $\neg$, $\vee$ and $\wedge$, the existential quantifier $\exists$, and a special symbol written length. These formulas will be interpreted as statements about words over the alphabet A. Given a word w, variables will represent positions in w and numerical predicates of arity $k \geq 0$ simply assign *true* or *false* to k-tuples of positions, independently of the letters that occur in those positions in w. Examples of numerical predicates that will be used later are the binary predicate $i_2 = i_1 + 1$ and the 0-ary predicate length $\equiv 0 \pmod 2$. The letter predicate $Q_a(i)$ is true if and only if the letter in position i in w is an a. Finally, the symbol length is interpreted as the length of w. This naturally extends to a truth value for any formula ϕ and we say that w satisfies ϕ if ϕ is true when interpreted over w. The language L *defined* by ϕ is the set of all words that satisfy ϕ.

Within this logical framework, in particular by varying the type of quantifiers allowed, it is possible to characterize a variety of Boolean circuit complexity classes. For example, with only existential quantification, as above, formulas using arbitrary numerical predicates define exactly the languages in the class AC^0 [10]. Further results and a more detailed presentation can be found in [19].

Following [19], we say that a numerical predicate is *regular* if it can be defined by a formula over the predicates "$<$" and "$\equiv 0 \pmod q$". For example, the predicate length $\equiv 0 \pmod 2$ is clearly regular and the binary predicate $i_2 = i_1 + 1$ is regular since it can be defined by

$$(i_1 < i_2) \wedge \neg\exists i_3[(i_1 < i_3) \wedge (i_3 < i_2)].$$

The term regular is motivated by

Fact 2 ([19]) *If a language L is defined by a formula in which all numerical predicates are regular, then L is regular.*

Moreover, regular predicates can be used to give logical characterizations of the regular languages contained in several Boolean circuit complexity classes. For example, the regular languages in AC^0 are precisely those defined by formulas using regular numerical predicates [3]. Further results can again be found in [19].

3 The two AC^0 hierarchies interleave strictly

For each $k \geq 0$, let BC_k^0 denote the class of languages that are recognized by AC^0 circuits of depth k. Define $\widehat{\mathrm{BC}}_k^0$ similarly by allowing input gates to compute any function of t positions of the input string, for some fixed t. By using the fact that any function of a constant number of input positions can be written as either a constant-size OR of AND's or as a constant-size AND of OR's, it is easy to see that $\mathrm{BC}_k^0 \subseteq \widehat{\mathrm{BC}}_k^0 \subseteq \mathrm{BC}_{k+1}^0$.

In [16], Sipser showed that the BC_k^0 hierarchy is infinite and Håstad [9] later strengthened the result by showing that the separation is exponential. In fact, Håstad's result even shows that $\widehat{\mathrm{BC}}_k^0 \subset \mathrm{BC}_{k+1}^0$. This implies that the $\widehat{\mathrm{BC}}_k^0$ hierarchy is infinite, and the separation is exponential.

In this section, by a simple direct argument using Håstad's separation result, we will prove that $\mathrm{BC}_k^0 \subset \widehat{\mathrm{BC}}_k^0$, which implies that the two hierarchies interleave strictly. We will then show how this translates to the AC_k^0 and $\widehat{\mathrm{AC}}_k^0$ hierarchies and prove that $\mathrm{AC}_k^0 \subset \widehat{\mathrm{AC}}_k^0 \subset \mathrm{AC}_{k+1}^0$.

We first need a more precise consequence of Håstad's lower bound result.

Fact 3 ([9]) *For every k, there is a function computable by linear-size depth-k circuits with level 1 fan-in $\sqrt{n^{1/k}\log n}$ that cannot be computed by depth-k circuits with level 1 fan-in $(1/20)\sqrt{n^{1/k}/\log n}$ and size $2^{(1/20)\sqrt{n^{1/k}/\log n}}$.*

Theorem 1 $\mathrm{BC}_k^0 \subset \widehat{\mathrm{BC}}_k^0 \subset \mathrm{BC}_{k+1}^0$.

Proof. Suppose that $\mathrm{BC}_{k-1}^0 = \widehat{\mathrm{BC}}_{k-1}^0$. Let $C_{k,n}$ be the canonical circuit of depth k with level 1 fan-in 2, in the sense that (1) the internal gates of $C_{k,n}$ form a full tree of height $k-1$ and arity n, (2) the leaves of this tree are level 1 gates of fan-in 2, (3) gate type in $C_{k,n}$ alternates from level to level, and

(4) the output gate of $C_{k,n}$ is an OR gate. Since $\widehat{\mathrm{BC}}^0_{k-1} \subseteq \mathrm{BC}^0_{k-1}$, there is a constant c such that $C_{k,n}$ can be simulated by a depth-$(k-1)$ circuit of size n^c. Now any depth-k circuit of size s with level 1 fan-in 2 can be rearranged so that it is identical to $C_{k,s}$, or the negation of $C_{k,s}$, except for its input gates. This implies that every depth-k circuit of size s with level 1 fan-in 2 can be simulated by a depth-$(k-1)$ circuit of size s^c.

Let C be a size-s, $s \in O(n)$, depth-k circuit with level 1 fan-in $\sqrt{n^{1/k}\log n}$ computing the function in the statement of Fact 3. Replace each of the level 1 gates by a binary tree of height $5 + \log\log n$ followed by gates of fan-in $(1/32)\sqrt{n^{1/k}/\log n}$. Replace the subcircuit formed by levels 2 to k of the original circuit plus one level of binary gates by an equivalent depth-$(k-1)$ circuit of size s^c. Recursively repeat this procedure until all the binary gates are gone. The resulting circuit has size $s^{(c+1)^{5+\log\log n}}$, depth k and level 1 fan-in $(1/32)\sqrt{n^{1/k}/\log n}$. Since $s \in O(n)$ and c is a constant, the size of the circuit is in $n^{O(\log n)}$, which contradicts Fact 3. $\Box$

Corollary 2 $\mathrm{AC}^0_k \subset \widehat{\mathrm{AC}}^0_k \subset \mathrm{AC}^0_{k+1}$.

Proof. Suppose that $\widehat{\mathrm{AC}}^0_k = \mathrm{AC}^0_{k+1}$. Then, by an argument similar to the one used in the proof of Theorem 1, there is a constant c such that any BC^0_{k+1} circuit of size s can be simulated by an $\widehat{\mathrm{AC}}^0_k$ circuit of size s^c. Consider a BC^0_{k+2} circuit. The inputs to the output gate of this circuit are BC^0_{k+1} circuits. Replace them by equivalent $\widehat{\mathrm{AC}}^0_k$ circuits. The constant-size Boolean combinations introduced between the output gate and new gates at level k can be eliminated by using, in particular, the fact that any function of a constant number of input positions can be written as either a constant-size OR of AND's or as a constant-size AND of OR's. The result is a $\widehat{\mathrm{BC}}^0_{k+1}$ circuit that simulates the original BC^0_{k+2} circuit. This contradicts Fact 3.

Suppose that $\mathrm{AC}^0_k = \widehat{\mathrm{AC}}^0_k$. Then, for every constant t, there is a constant c such that any $\widehat{\mathrm{BC}}^0_k$ circuit of size s and level 1 fan-in t can be simulated by an AC^0_k circuit of size s^c. Consider a $\widehat{\mathrm{BC}}^0_{k+1}$ circuit. The inputs to the output gate of this circuit are $\widehat{\mathrm{BC}}^0_k$ circuits. Replace them by equivalent AC^0_k circuits. The constant-size Boolean combinations introduced between the output gate and new gates at level k can again be eliminated, resulting in a BC^0_{k+1} circuit that simulates the original $\widehat{\mathrm{BC}}^0_{k+1}$ circuit. This contradicts Theorem 1. $\Box$

Note that the separation between BC^0_k and $\widehat{\mathrm{BC}}^0_k$, as well as that between AC^0_k and $\widehat{\mathrm{AC}}^0_k$, is superpolynomial. An exponential separation between BC^0_k and $\widehat{\mathrm{BC}}^0_k$ has been obtained recently by Cai, Chen and Håstad [7] by using random restriction techniques as in [9]. This implies an exponential separation between AC^0_k and $\widehat{\mathrm{AC}}^0_k$, by the same argument as in the proof of Corollary 2.

4 The p-variety corresponding to $\widehat{\mathbf{AC}}^0_1$

4.1 *Algebraic characterizations of* $\widehat{\mathrm{AC}}^0_1$

A subset of A^+ is a *locally testable* language iff it is a Boolean combination of sets of the form A^*v, vA^* and A^*vA^*, where v is in A^+. Let **LT** be the variety generated by the syntactic semigroups of locally testable languages. A subset of A^+ has *dot-depth one* iff it is a Boolean combination of sets of the form A^*v, vA^* and $A^*v_1A^* \cdots A^*v_kA^*$, where $k \geq 1$ and each v_i is in A^+. Let $\mathbf{DD}_1$ be the variety generated by the syntactic semigroups of dot-depth one languages. These varieties have been extensively studied [6,11]; in particular, known algebraic decompositions for **LT** and $\mathbf{DD}_1$, together with Corollary 3.3 of [14], imply that $\mathcal{P}(\mathbf{LT})$ and $\mathcal{P}(\mathbf{DD}_1)$ are closed under Boolean operations. In fact the next theorem shows that the two classes are equal.

Theorem 3 $\widehat{\mathrm{AC}}^0_1 = \mathcal{P}(\mathbf{LT}) = \mathcal{P}(\mathbf{DD}_1)$.

Proof. (a) $\widehat{\mathrm{AC}}^0_1 \subseteq \mathcal{P}(\mathbf{LT})$. Without loss of generality, a circuit in $\widehat{\mathrm{AC}}^0_1$ is a Boolean combination of unbounded fan-in OR's of AND's of fan-in at most t, for some constant t. Since $\mathcal{P}(\mathbf{LT})$ is closed under Boolean combination, it suffices to consider a single OR. Let D be an AND-gate querying bits $i_1, \dots, i_s$ for some $s \leq t$. Let $w = w_1 \dots w_s$ be the setting of these s bits that force D to 1. We construct a program $\phi_D = (i_1, f_1) \dots (i_s, f_s)b^{t-s}$, which maps A^n into $\{b, c\}^+$, where

$$f_k(a) = \begin{cases} b \text{ if } a = w_{j_k} \\ c \text{ otherwise} \end{cases}$$

Thus for any $x \in A^n, \phi_D(x) = b^t$ iff $x_{i_1} \dots x_{i_s} = w$, i.e., iff $D(x) = 1$. Concatenate all ϕ_D's together with in-between \$-producing instructions. The resulting program $\phi : A^n \to \{b, c, \$\}^+$ has the property that $\phi(x) \in \{b, c, \$\}^* b^t \$ \{b, c, \$\}^*$ iff $C(x) = 1$. Since $\{b, c, \$\}^* b^t \$ \{b, c, \$\}^*$ is in $\mathcal{M}(\mathbf{LT})$, we are done.

(b) $\mathcal{P}(\mathbf{LT}) \subseteq \mathcal{P}(\mathbf{DD}_1)$. This is obvious since $\mathbf{LT} \subseteq \mathbf{DD}_1$.

(c) $\mathcal{P}(\mathbf{DD}_1) \subseteq \widehat{\mathrm{AC}}^0_1$. Suppose L is in $\mathcal{P}(\mathbf{DD}_1)$: for each n, there exists a program ϕ and a language $K \in \mathcal{M}(\mathbf{DD_1})$ such that $x \in L \cap A^n$ iff $\phi(x) \in K$. K is a Boolean combination of languages of the form vB^*, B^*v

and $B^*v_1B^* \dots B^*v_sB^*$. Since $\widehat{\mathrm{AC}}_1^0$ is closed under Boolean combinations, we can assume K itself is one of these languages. We treat only the last possibility, the others being similar. An occurrence of the sequence of segments $v_1, \dots, v_s$ in $\phi(x)$ depends on at most $t = |v_1| + \dots + |v_s|$ instructions of ϕ, hence on at most t positions in x. We can thus construct C, an OR of AND's of fan-in t, one AND for each possible combination of at most t positions in the input that can create the required sequence of segments. This yields a circuit in $\widehat{\mathrm{AC}}_1^0$ such that $\phi(x) \in K$ iff $C(x) = 1$. □

It clearly follows that we also have $\widehat{\mathrm{AC}}_1^0 = \mathcal{P}(\mathbf{V})$ for any semigroup variety $\mathbf{V}$ such that $\mathbf{LT} \subseteq \mathbf{V} \subseteq \mathbf{DD}_1$.

4.2 $\mathbf{DD}_1$ *is a p-variety*

In order to show that $\mathbf{DD}_1$ is a p-variety we need to prove that $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{DD}_1)$ iff $S \in \mathbf{DD}_1$. Since $S \in \mathbf{V}$ always implies $\mathcal{W}(S) \subseteq \mathcal{P}(\mathbf{V})$, it suffices to show only the first implication; that is, we want to prove that whenever S is not in $\mathbf{DD}_1$, there is some language in $\mathcal{W}(S)$ that cannot be recognized by programs over semigroups in $\mathbf{DD}_1$, i.e., by circuits in $\widehat{\mathrm{AC}}_1^0$.

The following effective characterization for testing membership in $\mathbf{DD}_1$ will be used.

Lemma 4 ([11]) *$S \in \mathbf{DD}_1$ iff for all $e = e^2, f = f^2, s, t, u, v \in S$, the following equation holds:*

$$(esfte)^\omega (eufve)^\omega = (esfte)^\omega esfve(eufve)^\omega.$$

The key lemma in the argument is the following.

Lemma 5 *$L_1 = (e^*af^*be^*)^+(e^*cf^*de^*)^+$ is not in $\widehat{\mathrm{AC}}_1^0$.*

Proof. It is convenient to work with the following normal form for circuits in $\widehat{\mathrm{AC}}_1^0$. We will assume that our circuit has the form $C = OR(C_1, \dots, C_k)$, with $C_i = AND(C_{i1}, \dots, C_{ik}, D_i)$, where each C_{ij} is an unbounded fan-in OR of AND's of fan-in t, and each D_i is an unbounded fan-in AND of OR's of fan-in t. Suppose that such an n-input circuit accepts $L_1 \cap A^n$. Consider the input word $w = abe^{n-4}cd$: since w is in L_1, some C_i accepts it, say $C_1(w) = 1$. We can then find kt positions in w that force the C_{1j}'s to 1; hence there must be a segment of $m = \frac{n-4}{kt}$ consecutive positions in e^{n-4} that can be modified without changing the values of the C_{1j}'s. Set these m positions to $af^{m-2}d$, the resulting word, call it z, is not in L_1, hence $D_1(z) = 0$ (since

$C_{11}(z) = \ldots = C_{1k}(z) = 1)$. We can then find t positions in z that force D_1 to 0. Hence we must have a segment of $n' = \frac{m-2}{t}$ consecutive positions in f^{m-2} which can be modified without affecting the value of D_1. Set these n' positions to $be^{n'-2}c$. The resulting word is in L_1, hence has to be accepted by $C' = OR(C_2, \ldots, C_k)$ (since C_1 yields 0 on it). If n was initially chosen large enough, we can continue this process until the original circuit C is forced to 0 on a word that belongs to L_1, a contradiction. □

In fact, the above proof shows the following

Corollary 6 *Any circuit in* $\widehat{\mathrm{AC}}_1^0$ *that accepts all the words in* L_1 *must also accept some word in* $L_2 = (e^*af^*be^*)^*e^*af^*de^*(e^*cf^*de^*)^*$.

We are now ready to prove our main result.

Theorem 7 $\mathbf{DD}_1$ *is a p-variety.*

Proof. Suppose S is not in $\mathbf{DD}_1$, but that all the word problems of S are in $\widehat{\mathrm{AC}}_1^0$. By Lemma 4, there exist $e = e^2, f = f^2, s, t, u, v$ in S such that $(esfte)^\omega(eufve)^\omega \neq (esfte)^\omega esfve(eufve)^\omega$. Let $a = (esfte)^\omega esf$, $b = fte(esfte)^\omega$, $c = (eufve)^\omega euf$, $d = fve(eufve)^\omega$. Then, for any $i, j \geq 1$, we have

$$(e^*af^*be^*)^i(e^*cf^*de^*)^j = (esfte)^\omega(eufve)^\omega = abcd$$

and, for all $i, j \geq 0$, we have

$$(e^*af^*be^*)^ie^*af^*de^*(e^*cf^*de^*)^j = (esfte)^\omega esfve(eufve)^\omega = ad.$$

Consider the languages L_1, L_2 introduced above. Let θ be the morphism from A^+ into S^+, where $A = \{a, b, c, d, e, f\}$, defined by $\theta(a) = a$, $\theta(b) = b$, etc. Let η be the natural morphism from S^+ onto S. Then $\eta(\theta(L_1)) = \{abcd\}$ and $\eta(\theta(L_2)) = \{ad\}$.

By hypothesis, $\eta^{-1}(abcd) \in \widehat{\mathrm{AC}}_1^0$. This implies that $\overline{\theta^{-1}(\eta^{-1}(abcd))} \in \widehat{\mathrm{AC}}_1^0$. Since $L_1 \subseteq \theta^{-1}(\eta^{-1}(abcd))$ and $L_2 \subseteq \theta^{-1}(\eta^{-1}(ad)) \subseteq \overline{\theta^{-1}(\eta^{-1}(abcd))}$, there must be a circuit in $\widehat{\mathrm{AC}}_1^0$ that can separate the words in L_1 from those in L_2, contradicting Corollary 6. □

Let $\mathbf{J}$ be the variety of finite monoids M satisfying the property that $s\mathcal{J}t$ iff $s = t$, for all $s, t \in M$. Let $\mathbf{LI}$ be the variety of finite semigroups T satisfying $ete = e$, for all $e = e^2, t \in T$. It is known [11] that a semigroup S belongs to $\mathbf{DD}_1$ iff it divides a wreath product $M \circ T$, where M is in $\mathbf{J}$ and T is in $\mathbf{LI}$.

In [14], a general theorem about p-varieties of that form has, in our context, the following consequence.

Corollary 8 *The following are equivalent:*

a) $L \in \widehat{\mathrm{AC}}^0_1 \cap \mathrm{Reg}$.
b) *$S(L)$ is finite and for all $t \geq 1$, S is a subsemigroup of $\eta_L(A^t)$ implies $S \in \mathbf{DD}_1$.*
c) *There is some $q \geq 1$ such that L is recognized by a morphism $\phi : A^+ \to S \circ Z_q$ where $S \in \mathbf{DD}_1$ and for all $a \in A$, the projection onto Z_q of $\phi(a)$ is 1.*
d) *L is regular and is recognized by a single-scan program (i.e., a program that makes a single left-to-right scan over its input) over some semigroup in $\mathbf{DD}_1$.*

In the next section, it will be convenient to have a more precise combinatorial description of the regular languages in $\widehat{\mathrm{AC}}^0_1$. To obtain this characterization, we will use the so-called *wreath product principle*, which can be found, for example, in [17].

Lemma 9 *Let $L \subseteq A^+$ be a language recognized by a morphism $\phi : A^+ \to S \circ T$. Denote the projection of this morphism onto T by π. Let $B = T^\bullet \times A$ and $\sigma : A^+ \to B^+$ be the sequential function defined by $\sigma(a_1 \dots a_n) = (1, a_1)(\pi(a_1), a_2) \dots (\pi(a_1 \dots a_{n-1}), a_n)$. Then L is a Boolean combination of languages of the form $\pi^{-1}(t)$ for some $t \in T$ and $\sigma^{-1}(K)$, where $K \subseteq B^+$ is recognized by S.*

We can now prove the following

Corollary 10 *Let $L \subseteq A^+$. Then $L \in \widehat{\mathrm{AC}}^0_1 \cap \mathrm{Reg}$ iff L is a Boolean combination of languages of the form vA^*, A^*v and $(A^q)^*v_1(A^q)^* \dots (A^q)^*v_r(A^q)^*$, where v and all v_i's are in A^+.*

Proof. It is easily seen that every language described in the statement belongs to $\widehat{\mathrm{AC}}^0_1 \cap \mathrm{Reg}$. To prove the converse, we use part c) of Corollary 8. We will denote by 0 the identity of Z_q, by 1 its generator, and by $\underline{c}$ the integer $c \bmod q$. We have that L is recognized by a morphism $\phi : A^+ \to S \circ Z_q$ where $S \in \mathbf{DD}_1$ and the projection onto Z_q satisfies $\pi(a) = 1$ for each $a \in A$. By the wreath product principle, L is thus a Boolean combination of languages of the form $\pi^{-1}(c)$ for some $c \in Z_q$, and $\sigma^{-1}(K)$, where σ is the sequential function defined in Lemma 9. In the first case, we have $\pi^{-1}(c) = (A^q)^*A^c = \cup_{w \in A^c}(A^q)^*w(A^q)^*$. In the second case, since K is recognized by $S \in \mathbf{DD}_1$, we know that K is a Boolean combination of languages of the form uB^*, B^*v and $B^*v_1B^* \dots v_rB^*$. Because σ^{-1} commutes with Boolean operations, it suffices to consider the case when K is of the form just described. Obviously, the condition on the projection implies that $\sigma^{-1}(uB^*)$ is not empty iff $u = (0, a_1)(1, a_2)\ldots(\underline{k-1}, a_k)$ for some $u' = a_1 a_2 \ldots a_k \in A^+$, in which case it is equal to $u'A^*$. A similar argument shows that $\sigma^{-1}(B^*v)$ is empty or of the form A^*v' for some $v' \in A^+$. For the last case, we have that $\sigma^{-1}(B^*v_1B^*\ldots v_rB^*)$ is not empty iff $v_1 = (\underline{c_1}, a_{11})\ldots(\underline{c_1+k_1-1}, a_{1k_1})$, $\ldots$, $v_r = (\underline{c_r}, a_{r1})\ldots(\underline{c_r+k_r-1}, a_{rk_r})$. Let $v_1' = a_{11}\ldots a_{1k_1}$, $\ldots$, $v_r' = a_{r1}\ldots a_{rk_r}$. We then have $\sigma^{-1}(B^*v_1B^*\ldots v_rB^*) = (A^q)^*A^{d_1}v_1'(A^q)^*A^{d_2}\ldots v_r'A^*$, where the d_i's are chosen such that the occurrence of v_i' starts in a position congruent to c_i mod q. Now the final A^* can be rewritten as $\cup_{d\in Z_q} A^d(A^q)^*$ and each A^d as $\cup_{w\in A^d} w$; distributing concatenation over union will yield an appropriate expression for $\sigma^{-1}(K)$. $\square$

5 An application to the logical description of regular languages

Recall the logical framework described in Section 2.4. As mentioned there, formulas using arbitrary numerical predicates define exactly the languages in the class AC^0. This characterization can be sharpened to characterize precisely the various levels of the $\widehat{\mathrm{AC}}^0_k$ hierarchy. Say that a formula is Σ_0 if it is quantifier-free. Then, recursively define Σ_k formulas, for $k \geq 1$, as formulas of the form $\exists(i_1, \ldots, i_r)\phi(i_1, \ldots, i_r)$, where ϕ is a Boolean combination of Σ_{k-1} formulas.

Proposition 11 *A language belongs to $\widehat{\mathrm{AC}}^0_k$ if and only if it can be defined by a Boolean combination of Σ_k formulas using arbitrary numerical predicates.*

Proof. The reverse implication is easy and left to the reader.

For the forward implication, it is sufficient to consider an arbitrary family of depth-$(k+1)$ circuits in which gates at level 1 have constant fan-in. Rearrange the circuit so that (1) the internal gates form a full tree of height k and arity n^c, (2) the leaves of this tree are level 1 gates of fan-in t, and (3) gate type alternates from level to level. For simplicity of exposition, assume for the moment that $c = 1$.

Order the inputs to every gate in some arbitrary way. Gates at level k can now be identified by a number i_k from 1 to n. Gates at level $k-1$ are identified by a pair (i_k, i_{k-1}); this represents input i_{k-1} of gate i_k from level k. Similarly, gates at level d are identified by a $(k+1-d)$-tuple $(i_k, \ldots, i_d)$.

For each level $d \geq 2$ we recursively define a formula $\phi_d(i_k, \ldots, i_d)$ which expresses the fact that gate $(i_k, \ldots, i_d)$ at level d outputs 1. (For $d = k+1$, the formula ϕ_{k+1} expresses the fact that the output gate of the circuit outputs 1.)

Let $P_d(i_k, \ldots, i_d)$ be the numerical predicate that is true if and only if gate $(i_k, \ldots, i_d)$ at level d is an OR gate. Then, for $d \geq 3$, we define

$$\begin{aligned}\phi_d(i_k, \ldots, i_d) \equiv &(P_d(i_k, \ldots, i_d) \wedge \exists i_{d-1} \phi_{d-1}(i_k, \ldots, i_{d-1})) \\ &\vee(\neg P_d(i_k, \ldots, i_d) \wedge \neg \exists i_{d-1} \neg \phi_{d-1}(i_k, \ldots, i_{d-1})).\end{aligned}$$

There only remains to define $\phi_2(i_k, \ldots, i_2)$.

Let $\mathbf{i}$ denote $(i_k, \ldots, i_1)$. The formula $\phi_2(i_k, \ldots, i_2)$ will be defined by

$$\phi_2(i_k, \ldots, i_2) \equiv (P_2(i_k, \ldots, i_2) \wedge \exists i_1 \psi(\mathbf{i})) \vee (\neg P_2(i_k, \ldots, i_2) \wedge \neg \exists i_1 \neg \theta(\mathbf{i})),$$

where $\psi(\mathbf{i})$ expresses the fact that gate $\mathbf{i}$ outputs 1, assuming it is an AND gate, and $\theta(\mathbf{i})$ does the same, but assuming that gate $\mathbf{i}$ is an OR gate. To define ψ and θ, first let $R_{j,a}(\mathbf{i}, l)$, for $a \in A$, be the numerical predicate that is true if and only if "$X_l = a$" is the jth input of gate $\mathbf{i}$. The fact that this jth input outputs 1 can be expressed by either of the following two formulas:

$$\psi_j(\mathbf{i}) \equiv \exists l((R_{j,a_1}(\mathbf{i}, l) \wedge Q_{a_1}(l)) \vee \ldots \vee (R_{j,a_{|A|}}(\mathbf{i}, l) \wedge Q_{a_{|A|}}(l)))$$

or

$$\theta_j(\mathbf{i}) \equiv \neg \exists l((R_{j,a_1}(\mathbf{i}, l) \wedge \neg Q_{a_1}(l)) \vee \ldots \vee (R_{j,a_{|A|}}(\mathbf{i}, l) \wedge \neg Q_{a_{|A|}}(l))),$$

where $\{a_1, \ldots, a_{|A|}\} = A$. The formulas ψ and θ can then be defined as

$$\psi(\mathbf{i}) \equiv \psi_1(\mathbf{i}) \wedge \ldots \wedge \psi_t(\mathbf{i})$$

and

$$\theta(\mathbf{i}) \equiv \theta_1(\mathbf{i}) \vee \ldots \vee \theta_t(\mathbf{i}).$$

It is easy to verify that ϕ_2 is a Boolean combination of Σ_1 formulas. Then, by induction, we get that for every $d \geq 2$, ϕ_d is a Boolean combination of Σ_{d-1} formulas. Therefore, ϕ_{k+1} is a Boolean combination of Σ_k formulas that correctly expresses the fact that the circuit outputs 1, as required. To remove the assumption that $c = 1$, simply replace every variable i_d, for $d = k-1, \ldots, 0$, by a c-tuple of variables $(i_{d,1}, \ldots, i_{d,c})$. $\square$

As we also already mentioned, formulas using regular numerical predicates characterize precisely the regular languages in AC^0. It has been conjectured (e.g., [13]) that this characterization can also be sharpened to the various levels of the $\widehat{\mathrm{AC}^0_k}$ hierarchy, that is, that the regular languages in $\widehat{\mathrm{AC}^0_k}$ are precisely those defined by Boolean combinations of Σ_k formulas using regular numerical predicates. But until now, the only known result was that the regular languages defined by Σ_1 formulas are precisely those that can be defined by Σ_1 formulas using regular numerical predicates [13,19].

In this section, we use our results on $\widehat{\mathrm{AC}}_1^0$ and $\mathbf{DD}_1$ to generalize this characterization to Boolean combinations of Σ_1 formulas and thus to $\widehat{\mathrm{AC}}_1^0$. Note that this generalization is not trivial. If L is regular and defined by a Boolean combination of Σ_1 formulas, then L is a Boolean combination of languages $L_1, \ldots, L_c$ all of which can be defined by Σ_1 formulas. If all the L_i were regular, then we would be done. However, there is no guarantee that any of the L_i is regular.

Theorem 12 *A regular language belongs to $\widehat{\mathrm{AC}}_1^0$ if and only if it can be defined by a Boolean combination of Σ_1 formulas using regular numerical predicates.*

Proof. The reverse implication follows directly from Fact 2 and the previous proposition.

Now suppose that L is a regular language in $\widehat{\mathrm{AC}}_1^0$. Then, by Corollary 10, L is a Boolean combination of languages of the form v_1A^*, A^*v_2 and $(A^q)^*w_1(A^q)^* \ldots (A^q)^*w_r(A^q)^*$, where v_1, v_2 and w_i are in A^+. All of these languages can be described by Σ_1 formulas using the numerical predicates "$i_1 = i_2 + 1$" and "$i_1 \equiv c \pmod q$". For example, $(A^q)^*w_1(A^q)^* \ldots (A^q)^*w_r(A^q)^*$ can be defined by

$$\begin{aligned}
&\exists(i_{1,1}, \ldots, i_{1,|w_1|}, \ldots, i_{r,1}, \ldots, i_{r,|w_r|}) \\
&\quad [Q_{w_{1,1}}(i_{1,1}) \wedge \ldots \wedge Q_{w_{r,|w_r|}}(i_{r,|w_r|}) \\
&\wedge (i_{1,2} = i_{1,1} + 1) \wedge \ldots \wedge (i_{r,|w_r|} = i_{r,|w_r|-1} + 1) \\
&\wedge (i_{1,1} \equiv 1 \pmod q) \wedge \ldots \wedge (i_{r,1} \equiv |w_1| + \cdots + |w_{r-1}| + 1 \pmod q) \\
&\wedge \text{length} \equiv |w_1| + \cdots + |w_r| + 1 \pmod q].
\end{aligned}$$

The predicate "$i_1 = i_2 + 1$" was shown to be regular in Section 2.4. An easy induction shows that the other predicates are also regular since "$i_1 \equiv c \pmod q$" can be defined by $\exists i_0[(i_1 = i_0 + 1) \wedge (i_0 \equiv c - 1 \pmod q)]$. $\square$

Straubing has recently generalized the characterization even further to Boolean combinations of Σ_1 formulas in which *modular quantifiers* may appear in addition to the usual existential quantifiers [20]. His work is independent from ours and uses a totally different approach.

References

[1] M. Ajtai, Σ_1^1-formulae on finite structures, *Ann. Pure Appl. Logic* **24** (1983) 1–48.

[2] D. Mix Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *J. Comput. System Sci.* **38** (1989) 150–164.

[3] D. Mix Barrington, K. Compton, H. Straubing and D. Thérien, Regular Languages in NC^1, *J. Comput. System Sci.* **44** (1992) 478–499.

[4] D. Mix Barrington, H. Straubing and D. Thérien, Non-uniform Automata over Groups, *Inform. and Comput.* **89** (1990) 109–132.

[5] D.A. Barrington and D. Thérien, Finite Monoids and the Fine Structure of NC^1, *J. ACM* **35** (1988) 941–952.

[6] J.A. Brzozowski and I. Simon, Characterization of locally testable events, *Discrete Math.* **4** (1973) 243–271.

[7] L. Cai, J. Chen and J. Håstad, Circuit Bottom Fan-in and Computational Power, *SIAM J. on Computing* **27** (1998) 341–355.

[8] M. Furst, B. Saxe and M. Sipser, Parity, Circuits and the Polynomial Time Hierarchy, *Math. Systems Theory* **17** (1984) 13–27.

[9] J. Håstad, Almost optimal lower bounds for small depth circuits, in: *Advances in Computing Research*, Vol. 5, (JAI Pres, 1989) 143–170.

[10] N. Immerman, Languages That Capture Complexity Classes, *SIAM J. on Computing* **16** (1987) 760–778.

[11] R. Knast, A semigroup characterization of semigroups of dot-depth one languages, *RAIRO Inform. Théor. Appl.* **17** (1983) 321–330.

[12] P. McKenzie, P. Péladeau and D. Thérien, NC^1: the automata-theoretic viewpoint, *Comput. Complexity* **1** (1991) 330–359.

[13] P. Péladeau, Classes de circuits booléens et variétés de monoïdes, Ph.D. Thesis, Université de Paris VI, 1990.

[14] P. Péladeau, H. Straubing and D. Thérien, Finite semigroup varieties defined by programs, *Theoret. Comput. Sci.* **180** (1997) 325–339.

[15] M.P. Schützenberger, On finite monoids having only trivial subgroups, *Inform. and Control* **8** (1965) 190–194.

[16] M. Sipser, Borel sets and circuit complexity, in: *Proceedings of the 15th ACM Symposium on Theory of Computing* (1983) 61–69.

[17] H. Straubing, Varieties of recognizable sets whose syntactic monoids contain solvable groups, Ph.D. Thesis, University of California, Berkeley, 1978.

[18] H. Straubing, Constant-depth periodic circuits, *Internat. J. Algebra Comput.* **1** (1991) 49–88.

[19] H. Straubing, *Finite Automata, Formal Logic and Circuit Complexity* (Birkhauser, 1994).

[20] H. Straubing, Languages defined with modular counting quantifiers (extended abstract), in: *15th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, vol. 1373 (Springer, 1998) 332–343.