An Algebraic Point of View on the Crane-Beach Conjecture

Clemens Lautemann, Pascal Tesson*[1], and Denis Thérien**[2]

[1] Département d'Informatique et de Génie Logiciel, Université Laval
pascal.tesson@ift.ulaval.ca
[2] School of Computer Science, McGill University
denis@cs.mcgill.ca

Abstract. A letter $e \in \Sigma$ is said to be neutral for a language L if it can be inserted and deleted at will in a word without affecting membership in L. The Crane-Beach Conjecture, which was recently disproved, stated that any language containing a neutral letter and definable in **FO** is in fact **FO**[<] definable and is thus a regular, star-free language. More generally, we say that a logic or a computational model has the Crane Beach property if the only languages with neutral letter that it can define/compute are regular.
We develop an algebraic point of view on the Crane Beach properties using the program over monoid formalism which has proved of importance in circuit complexity. Using recent communication complexity results we establish a number of Crane Beach results for programs over specific classes of monoids. These can be viewed as Crane Beach theorems for classes of bounded-width branching programs. We also apply this to a standard extension of **FO** using modular-counting quantifiers and show that the boolean closure of this logic's $\mathbf{\Sigma}_1$ fragment has the CBP.

1 Introduction

A number of results of the last ten years indicate that weak models of computation are considerably handicapped when they are used to recognize languages with a so-called *neutral letter*. A letter $e \in \Sigma$ is said to be *neutral* for the language $L \subseteq \Sigma^*$ if it can be inserted and deleted at will in a word without affecting membership in L, i.e. if for all $u, v \in \Sigma^*$ it holds that $uv \in L \Leftrightarrow uev \in L$. It is natural to think that languages with neutral letters pose particular problems for models of computation that rely heavily on the precise location of certain input bits or are too weak to preprocess their input by removing the blank symbols.

To the best of our knowledge, the first result along those lines appears in work of Barrington and Straubing on short bounded-width branching programs: any language recognized by a width k branching program of length $o(n \log \log n)$ is in fact regular and belongs to some well defined class $\mathcal{L}_k$ of regular languages [4].

* Research supported in part by NSERC and FQRNT.
** Research supported in part by NSERC and FQRNT.

In light of this result, Lautemann and Thérien conjectured that any language with a neutral letter in the circuit class AC^0 is in fact regular and star-free (i.e. it can be defined by a regular expression built from the letters of the alphabet, the empty set symbol, union and complementation but no Kleene star). This statement, which came to be known as the *Crane Beach conjecture*, has a nice logical formulation. A language is known to belong to AC^0 iff it can be defined in $\mathbf{FO}[Arb]$, i.e. by a first-order sentence using arbitrary numerical predicates [12]. Restrictions on the set of numerical predicates available can be interpreted as *uniformity restrictions* on circuits [3, 13, 22] (see Section 2). When order is the only available numerical predicate ($\mathbf{FO}[<]$), the class of definable languages corresponds exactly to the star-free regular languages (cf. [22, 33]). Thus if $\mathcal{L}_e$ denotes the class of languages with a neutral letter, the Crane Beach conjectured that $\mathbf{FO}[Arb] \cap \mathcal{L}_e = \mathbf{FO}[<] \cap \mathcal{L}_e$. The underlying intuition was the apparent inability to take advantage of complicated numerical predicates in the presence of a neutral letter. But the Crane Beach conjecture was refuted in [2].

On the other hand [2] show that the boolean closure of the $\mathbf{\Sigma}_1$ fragment of $\mathbf{FO}[Arb]$ does have the Crane Beach property in the sense that $\mathbf{B}\,\mathbf{\Sigma}_1[Arb] \cap \mathcal{L}_e = \mathbf{B}\,\mathbf{\Sigma}_1[<] \cap \mathcal{L}_e$. Such Crane Beach results for fragments of **FO** are related to so-called *collapse results* in database theory [8].

We also consider in this paper the logic $\mathbf{FO}+\mathbf{MOD}$ which is a standard extension of first-order logic on words in which *modular counting* quantifiers are introduced. The expressive power of $\mathbf{FO}+\mathbf{MOD}[<]$ is limited to a specific class of regular languages [25] whereas $\mathbf{FO}+\mathbf{MOD}[Arb]$ captures the circuit class ACC^0.

We will say that a logic or a computation model has the Crane Beach property if all languages with a neutral letter that it can define or compute is regular. We present an algebraic point of view on the Crane Beach property by considering finite monoids as language recognizers. Two methods of recognition by finite monoids have been introduced in the literature: recognition via homomorphism and recognition via *programs over monoids*. The first is more classical and is a key ingredient in a number of major results concerning regular languages and in particular those about the expressive power of fragments of $\mathbf{FO}+\mathbf{MOD}[<]$. Programs are more powerful and yield algebraic characterizations of AC^0, ACC^0 and NC^1 [1, 6].

In many ways, the extra expressive power afforded by programs over morphisms is similar to the extra power afforded to **FO** or $\mathbf{FO}+\mathbf{MOD}$ when we go from sentences that use only $<$ as a numerical predicate to sentences using arbitrary numerical predicates. We show that there are classes of monoids for which the presence of a neutral letter nullifies this advantage in the sense that any language with a neutral letter recognized by a program over a monoid in the class can in fact be recognized by a morphism over a monoid in the same class.

These results allow us to show that the boolean closure of the $\mathbf{\Sigma}_1$ fragment of $\mathbf{FO}+\mathbf{MOD}$, which we formally denote as $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}$ has the Crane Beach property, i.e. $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[Arb] \cap \mathcal{L}_e = \mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[<] \cap \mathcal{L}_e$. They also provide some additional

insight into a possible dividing line between classes of monoids or fragments of $\mathbf{FO}^+\mathbf{MOD}$ which exhibit the Crane Beach property from those who do not.

We begin by reviewing in section 2 the necessary background in circuit complexity, descriptive complexity and the study of finite monoids as language recognizers. In section 3 we obtain our results on the Crane Beach property for programs over monoids and study their application to logic in section 4. We conclude with a discussion on how our results fit with other results concerning languages with a neutral letter.

2 Logic, Circuits, Programs over Monoids and Automata

2.1 Circuits

An n-input boolean circuit $\mathcal{C}_n$ is a directed acyclic graph with a single node of out-degree 0 called the *output gate*. The *input gates* have in-degree 0 and are labeled either with an input variable x_i, its complement $\overline{x_i}$ or one of the boolean constants $0, 1$. When the inputs are not boolean but take values in some finite alphabet Σ, input nodes are labeled by $x_i = a$ for some $a \in \Sigma$. Finally, any non-input gate g is labeled by some symmetric boolean function f_g taken from some predetermined set. Our focus will be on the case where these functions are either the AND or the OR function, or the function MOD_q which outputs 1 if the sum of its entries is divisible by q. The *depth* and *size* of a circuit $\mathcal{C}_n$ are, respectively, the longest path from an input node to the output node and the number of gates. A circuit naturally computes a function $f_{\mathcal{C}_n} : \Sigma^n \to \{0,1\}$ and we define the *language accepted by* $\mathcal{C}_n$ as $L_{\mathcal{C}_n} = \{x : f_{\mathcal{C}_n} = 1\}$. This language is a subset of Σ^n: in order to recognize subsets of Σ^* we use families of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\geq 0}$ where each $\mathcal{C}_n$ is an n-input circuit which is used to process inputs of that particular length. We can then consider the depth and size of $\mathcal{C}$ as a function of n and study its asymptotics.

The class ACC^0 consists of languages which can be accepted by a family of circuits having polynomial-size and bounded-depth and constructed with AND gates, OR gates and MOD_q gates for some $q \geq 2$. We further define AC^0 as the restriction of ACC^0 where only AND and OR gates are allowed and CC^0 as the restriction of ACC^0 where only MOD_q gates (for some $q \geq 2$) are used. All of these classes lie in the class NC^1 of languages recognized by circuits of depth $O(\log n)$ constructed with AND and OR gates of bounded fan-in.

We have not imposed any sort of restriction on the effective constructibility of the circuit families and the circuit classes are correspondingly dubbed 'non-uniform'. It makes sense to require that the nth circuit of a family $\mathcal{C}$ be constructible efficiently and such requirements are called *uniformity conditions*. For any complexity class $\mathcal{D}$, we say that a family of circuits $\mathcal{C} = \{C_n\}$ is $\mathcal{D}$-*uniform* if there is an algorithm in $\mathcal{D}$ which on input n computes a representation of C_n (see e.g. [3] for a formal discussion). DLOGTIME-*uniformity* is widely accepted as the desired 'correct' notion of uniformity for subclasses of NC^1: roughly speaking it requires that there exists an algorithm which on input $\langle t, a, b, n\rangle$ can check

in time $O(\log n)$ whether the ath gate of C_n is of type t (i.e. which function it computes) and feeds into the bth gate [3].

2.2 Logic over Words

We are interested in considering first-order logical sentences defining sets of finite words over an alphabet Σ. We only briefly overview this logical apparatus and refer the reader to [22, 15] for a more thorough and formal discussion.

Let us start with an example. Over the alphabet $\{a, b\}$ we view the sentence

$$\exists x \exists y \;\; Q_a x \wedge Q_b y \wedge (y = x + x)$$

as defining the sets of words in which there exists a position x holding an a such that the position $2x$ holds a b. The variables in the sentence stand for positions in a finite word and the access to the content of these positions is provided by the unary predicates Q_a and Q_b.

More generally, for any alphabet Σ we construct sentences using two types of atomic formulas. First, for each $a \in \Sigma$, we include a *content predicate* $Q_a x$ which is interpreted as true of a finite word w if the position x in w holds the letter a. The second atomic formulas are *numerical predicates* $P(x_1, \ldots x_k)$. The truth of $P(x_1, \ldots, x_k)$ depends only on the values $x_1, \ldots, x_k$ and on the length of w but not on the actual letters in w. For a set $\mathcal{P}$ of numerical predicates, we denote as $\mathbf{FO}[\mathcal{P}]$ the class of sentences which can be constructed from the atomic formulas $Q_a x$ and $P(x_1, \ldots, x_k)$ with $P \in \mathcal{P}$ using existential and universal quantifiers and boolean connectives. For $\phi \in \mathbf{FO}[\mathcal{P}]$ we further denote as L_ϕ the language in Σ^* defined by ϕ i.e. the set of finite words such that $w \models \phi$.

We also consider the case where first-order is extended by the introduction of modular-counting quantifiers. The formula $\exists^{i \pmod p} x \; \psi(x)$ is true if the number of positions x such that $\psi(x)$ is equal to i modulo p. We denote as $\mathbf{FO}+\mathbf{MOD}[\mathcal{P}]$ the class of sentences constructed from the atomic formulas, boolean connectives and both modular and existential/universal quantifiers.

The case where $\mathcal{P}$ contains only the order relation $<$ has been thoroughly investigated [22, 33, 32]. A corollary of Büchi's theorem about monadic second-order logic over words establishes that $\mathbf{FO}+\mathbf{MOD}[<]$ contains only regular languages. In fact these can be characterized as languages whose syntactic monoid is solvable (see next subsection). The expressive power of various fragments of $\mathbf{FO}+\mathbf{MOD}[<]$ can also be characterized using algebraic automata theory and in particular $\mathbf{FO}[<]$ captures exactly the star-free languages which in turn correspond to languages with aperiodic syntactic monoids.

On the other end of the spectrum, let *Arb* be the set of all numerical predicates. The classes $\mathbf{FO}[Arb]$ and $\mathbf{FO}+\mathbf{MOD}[Arb]$ correspond exactly to non-uniform AC^0 and ACC^0 respectively. Restrictions on the set of allowed numerical predicates translate in many natural cases into uniformity restrictions on the circuits [3, 7]. Most notably, $\mathbf{FO}[+, *]$ and $\mathbf{FO}+\mathbf{MOD}[+, *]$ correspond to the DLOGTIME-uniform versions of AC^0 and ACC^0.

The class $\mathcal{R}eg$ of regular numerical predicates has also been the focus of some attention. A numerical predicate is said to be *regular* if it can be defined by an

$\mathbf{FO}+\mathbf{MOD}[<]$ formula. By definition $\mathbf{FO}+\mathbf{MOD}[\mathcal{R}eg]$ has the same expressive power as $\mathbf{FO}+\mathbf{MOD}[<]$ and thus contains only regular languages. It is also known that a language L is definable in $\mathbf{FO}[\mathcal{R}eg]$ iff it is regular and can be recognized by an AC^0 circuit. In other words $\mathbf{FO}[Arb] \cap \mathrm{REG} = \mathbf{FO}[\mathcal{R}eg]$, where REG denotes the class of regular languages. For a number of other fragments of $\mathbf{FO}+\mathbf{MOD}$ it has been shown that when defining regular languages arbitrary numerical predicates hold no expressive advantage over regular predicates.

For technical reasons it is convenient to have a quantifier-free description of regular numerical predicates. For integers $t \geq 0$ and $q \geq 2$ and any $n < t + q$ we define a binary relation $\delta_{n,t,q}$ by setting $x\delta_{n,t,q}y$ if either $x - y = n$ or $x \equiv y \mod q$. We further define for any $n < t + q$ a unary relation $\kappa_{n,t,q}$ by setting $\kappa_{n,t,q}(x) \Leftrightarrow x\delta_{n,t,q}0$. A numerical predicate is regular iff it can be defined as a boolean combination of $\delta_{n,t,q}$, $\kappa_{n,t,q}$ and $<$ [17].

2.3 Programs over Monoids

We now turn to an algebraic characterization of the circuit classes presented earlier. We refer the reader to [31] for a more thorough discussion of the links between complexity and the algebraic theory of regular languages.

A *monoid* is a set M equipped with a binary associative operation $\cdot_M$ and a distinguished identity element 1_M. A class of finite monoids forms a *variety* (or more precisely a *pseudovariety*) if it is closed under direct product, formation of submonoids and morphic images.

The *free monoid* Σ^* over the alphabet Σ is the set of finite words over Σ with concatenation as the monoid operation. The empty word ϵ acts as the identity element in this case. With the exception Σ^*, all monoids considered in this paper are finite and we view these algebraic objects as language recognizers.

We say that a language $L \subseteq \Sigma^*$ is *recognized via morphism* or simply *recognized* by the finite monoid M if there exists a morphism $\phi : \Sigma^* \to M$ and a set $F \subseteq M$ such that $L = \phi^{-1}(M)$. This definition simply restates algebraically the notion of acceptance by a finite automaton and a simple variant of Kleene's theorem shows that a language is regular if and only if it can be recognized by some finite monoid. For every regular language L, the syntactic monoid $M(L)$ of L is the smallest monoid recognizing L and $M(L)$ is in fact isomorphic to the transition monoid of L's minimal automaton. For a variety $\mathbf{V}$ we denote as $\mathbf{L}(\mathbf{V})$ the class of regular languages with syntactic monoids in $\mathbf{V}$. These classes (which form *language varieties*) are a natural unit of classification for regular languages and are at the heart of the algebraic theory of regular languages [18].

We give a list of varieties that bear importance in this paper but also in other applications of algebraic automata theory [18, 31].

- The variety $\mathbf{A}$ consists of *aperiodic* or *group-free* monoids, i.e. monoids having no non-trivial subgroup.
- The variety $\mathbf{G_{nil}}$ consists of *nilpotent groups*, i.e. groups which are direct products of p-groups. An alternate and in our case more useful definition of nilpotency can be given as follows. For a finite group G and any $g, h \in G$, the

commutator $[g,h]$ of g and h is the element $g^{-1}h^{-1}gh$. For any subgroups $H_1, H_2 \subseteq G$ we denote as $[H_1, H_2]$ the subgroup generated by the commutators $[h_1, h_2]$ with $h_1 \in H_1$ and $h_2 \in H_2$. Now define inductively the chain of subgroups of G by $G_0 = G$ and $G_i = [G, G_i]$. We say that a group is *nilpotent of class k* if G_k is the trivial group and denote as $\mathbf{G_{nil,k}}$ the variety of such groups. A group is *nilpotent* if it is nilpotent of class k for some k. Note that a group is nilpotent of class 1 iff it is Abelian.

- The variety $\mathbf{G_{sol}}$ of solvable groups and the variety $\mathbf{M_{sol}}$ of *solvable monoids*, i.e. monoids whose subgroups are solvable.
- For any variety of groups $\mathbf{H}$, we denote as $\overline{\mathbf{H}}$ the variety of monoids whose subgroups all belong to $\mathbf{H}$.
- The variety $\mathbf{DO}$ consists of monoids which for some $n \geq 1$ satisfy the identity $(xy)^n(yx)^n(xy)^n = (xy)^n$.
- The variety $\mathbf{DA}$ consists of monoids which satisfy $(xy)^n y (xy)^n = (xy)^n$ for some n. In fact $\mathbf{DA}$ is the intersection of $\mathbf{DO}$ and $\mathbf{A}$.
- The variety $\mathbf{J}$ of $\mathcal{J}$-trivial monoids consists of aperiodic monoids which satisfy $(xy)^n = (yx)^n$ for some n.

For any variety $\mathbf{V}$ in the above list, the corresponding class of regular languages $\mathbf{L}(\mathbf{V})$ admits nice descriptions [18, 31] and the varieties $\mathbf{DA}, \mathbf{DO}$ and $\mathbf{G_{nil}}$ are often central in investigations in the complexity of regular languages and their logical descriptions [29, 30, 32].

The *program over monoid* formalism introduced by Barrington and Thérien provides a slight extension of a finite monoid's computing power. An n-input program ϕ_n over M of length ℓ is a sequence of *instructions* $\phi_n : (i_1, f_1) \dots (i_\ell, f_\ell)$ with $1 \leq i_j \leq n$ and where each f_i is a function from the input alphabet Σ to M. Given an input $w \in \Sigma^n$ a program produces a string of ℓ monoid elements $\phi_n(w) = f_1(w_{i_1}) \dots f_\ell(w_{i_\ell})$ which are then multiplied in M. We abuse notation and also denote as $\phi_n(w)$ the product $f_1(w_{i_1}) \cdot_M \cdots \cdot_M f_\ell(w_{i_\ell})$. By specifying a set of accepting elements $F \subseteq M$ we can use such a program to recognize a subset of Σ^n and subsets of Σ^* can be recognized through families of programs. As is the case for circuits, one can consider uniformity restrictions on these families.

A result of Barrington [1] shows that a language L can be recognized by a polynomial-length family of programs over a finite monoid iff L belongs to NC^1. We denote as $\mathbf{P}(\mathbf{V})$ the class of languages which can be recognized by a program of polynomial length over a monoid in $\mathbf{V}$. Further refinements of Barrington's theorem appear in [6]: L belongs to AC^0 iff L lies in $\mathbf{P}(\mathbf{A})$, L belongs to CC^0 iff it lies in $\mathbf{P}(\mathbf{G_{sol}})$ and L belongs to ACC^0 iff it lies in $\mathbf{P}(\mathbf{M_{sol}})$. These results are robust with respect to many standard uniformity restrictions [3].

A variety $\mathbf{V}$ of finite monoids forms a *program-variety* if every regular language with a neutral letter in $\mathbf{P}(\mathbf{V})$ is in $\mathbf{L}(\mathbf{V})$. Alternatively, we can introduce the notion as follows: say that the multiplication of a monoid M can be program-simulated by a monoid N if for every element $m \in M$ the language $L_m \subseteq M^*$ defined as $L_m = \{m_1 m_2 \dots m_n : m_1 \cdot m_2 \cdot \cdots \cdot m_n = m\}$ can be recognized by a polynomial-length program over N. Now $\mathbf{V}$ forms a program-variety if any M which can be simulated by some $N \in \mathbf{V}$ is in fact in $\mathbf{V}$ itself [16, 23].

The lower bounds for AC^0 circuits computing the MOD_p function [21] can be rephrased as showing that the aperiodic monoids form a program-variety. Many of the important questions in circuit complexity can similarly be rephrased in algebraic terms: for instance ACC^0 is strictly contained in NC^1 iff the solvable monoids form a program-variety.

Programs over finite monoids are closely related to bounded-width branching programs (BWBP). An n-input BWBP of width k and length ℓ over input alphabet Σ is a leveled directed graph with the following structure. Each level $1 \leq i < \ell$ is associated with an input variable x_{j_i} and contains k nodes that each have $|\Sigma|$ outgoing edges (to level $i+1$) labeled by the possible values of the input variable x_{j_i}. Moreover, the first level contains a distinguished start node while the last level contains an accepting and a rejecting node. Any word $w \in \Sigma^n$ naturally traces out a unique path in this graph and the language accepted by the BWBP is the set of w leading to the accepting node.

Note that in a BWBP a letter $a \in \Sigma$ induces a function $f_{i,a}$ from the k nodes of level i to the k nodes of level $(i+1)$. It is not hard to see that the difference between BWBP is essentially cosmetic since a program over M can immediately be rewritten as a BWBP of width $|M|$ while, conversely, a BWBP of width k can be rewritten as a program over the finite monoid generated by the functions $f_{i,a}$. The algebraic point of view provides a finer analysis of the BWBP model by parameterizing its power in terms of the algebraic structure of the $f_{i,a}$.

3 The Crane Beach Property

We say that a class $\mathcal{L}$ of languages has the *Crane Beach property* (or CBP) if every language with a neutral letter in $\mathcal{L}$ is regular. As we mentioned in the introduction, it was conjectured but later disproved that $\mathbf{FO}[Arb]$ has the CBP and one can infer from [4] that BWBP of length $o(n \log\log n)$ have the CBP.

For a class of languages having the Crane Beach property, it is also interesting to understand exactly what regular languages with a neutral letter belong to the class. In the case of a logical fragment of $\mathbf{FO}+\mathbf{MOD}$ using numerical predicates in some class $\mathcal{P}$ we are often most interested in cases where the presence of the neutral letter reduces the expressive power to that obtained with the *same* fragment but using $<$ as the sole numerical predicate. For instance, $\mathbf{B}\,\mathbf{\Sigma}_1[Arb]$ has the CBP [2] and the regular languages with a neutral letter definable in this fragment are exactly those definable in $\mathbf{B}\,\mathbf{\Sigma}_1[<]$. We will usually refer to such theorems as *strong Crane Beach results*.

3.1 A Communication Complexity Crane Beach Theorem

The "input on the forehead" model of communication complexity, first introduced in [9] has found a wide variety of applications in numerous areas of complexity theory [14]. It involves k parties wishing to compute a function f of k variables $x_1, \ldots, x_k$: the ith player receives access to all the inputs except x_i so that one can conveniently picture this player as having x_i written on his forehead. The players

want to minimize the number of bits that need to be exchanged when computing f on the worst-case input. When the function to be computed is not explicitly given as a k variable function, we further assume that input bits are partitioned in a way that is known to the different parties but chosen adversarially. The k-party *communication complexity of a language* L is the function $D_k(L) : \mathbb{N} \to \mathbb{N}$ giving for each n the minimum number of bits that k parties need to exchange to compute membership in L of the worst-case input w of length n under the worst-case partition of the letters in w. The following theorem which combines two results of [10] establishes a Crane Beach property for the k-party model.

Theorem 1.

a) If L is a language with a neutral letter such that $D_k(L) = O(1)$ for some fixed $k \geq 2$ then L is regular.

b) If L is a regular language with a neutral letter then $M(L)$ lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ iff there exists some k such that $D_k(L) = O(1)$.

We define the k-party *communication complexity of a finite monoid* M (denoted $D_k(M)$) as the complexity for k parties to evaluate the product in M of n elements $m_1, \ldots, m_n$ distributed on their foreheads (note that up to a constant factor, the worst partition in this case gives to player i access to all elements except those with an index congruent to i modulo k). Underlying the previous theorem is the result of [26] that a monoid lies in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ iff there exists a k s.t. $D_k(M) = O(1)$.

Suppose that k players want to test if a word w of length n belongs to a language L which is recognized by a program ϕ of length $\ell(n)$ over a finite monoid M. The output of an instruction querying bit x_i can be computed privately by any of the $k-1$ players having access to x_i and the output of the program $\phi(w)$ can then be evaluated using a protocol which evaluates the product of the monoid elements resulting from individual instructions. Hence, the k-party communication complexity of L on inputs of length n is at most the communication complexity of M on strings of length $\ell(n)$. These observations lead to the following lemma [19, 26]:

Lemma 2. *Let $k \geq 2$ be some integer and let $f : \mathbb{N} \to \mathbb{N}$ be such that $f = O(\log^c n)$ for some $c \geq 0$. The class $\mathbf{V}$ of monoids M such that $D_k(M) = O(f)$ is a program-variety.*

We give a proof in the appendix for completeness. This lemma provides a way to use recent results on the communication complexity of finite monoids [19, 29, 10, 26] to identify program-varieties.

Corollary 3. *The following are program-varieties:* $\mathbf{DA}, \mathbf{G_{nil}}, \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ *and* $\mathbf{G_{nil,k}}$ *for each $k \geq 1$.*

Proof (Sketch). It is not hard to see that the intersection of two program-varieties also forms a program-variety. We know that aperiodic monoids form a program-variety. Moreover, by Lemma 2, the class of monoids with 2-party communication

complexity $O(\log n)$ also forms a program-variety. By results of [29], an aperiodic monoid has 2-party communication complexity $O(\log n)$ if it belongs to **DA** and so **DA** is a program-variety.

The statement for $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ follows directly from lemma 2 and theorem 1.

The statement for $\mathbf{G_{nil}}$ is a consequence of the work of [5]. Once we have that $\mathbf{G_{nil}}$ is a program-variety however, we can again use lemma 2 to obtain that each $\mathbf{G_{nil,k}}$ is a program-variety because [19] shows that a group has bounded $k+1$-party communication complexity iff it is nilpotent of class k. □

3.2 The Crane Beach Property for Programs over Monoids

Definition 4. *A variety of monoids* $\mathbf{V}$ *is said to have the* weak Crane-Beach property *if* $\mathbf{P}(\mathbf{V}) \cap \mathcal{L}_e \subseteq$ REG*, the class of regular languages. Furthermore,* $\mathbf{V}$ *has the* strong Crane-Beach property *(CBP) if* $\mathbf{P}(\mathbf{V}) \cap \mathcal{L}_e \subseteq \mathbf{L}(\mathbf{V})$*, that is if polynomial length programs over* $\mathbf{V}$ *are no more powerful than morphisms over* $\mathbf{V}$ *in the presence of a neutral letter.*

Note that if $\mathbf{V}$ has the weak CBP then any subvariety of $\mathbf{V}$ also has this property. However, the same statement does not hold for the strong CBP.

Lemma 5. *If* $\mathbf{V}$ *has the weak CBP and* $\mathbf{W} \subseteq \mathbf{V}$ *is a program-variety then* $\mathbf{W}$ *has the strong CBP.*

Proof. Because $\mathbf{V}$ has the weak CBP and $\mathbf{W} \subseteq \mathbf{V}$ we have that if L is a language with a neutral letter accepted by a program ϕ over $M \in \mathbf{W} \subseteq \mathbf{V}$, then L is regular. By definition of program-varieties we can now guarantee that $L \in \mathbf{L}(\mathbf{W})$ and so $\mathbf{W}$ has the strong CBP. □

We can use the communication complexity results cited earlier to obtain:

Theorem 6. *The variety* $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ *has the strong CBP.*

Proof. The result for $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ stems from Theorem 1. Indeed, we need to show that any language L with a neutral letter which is recognized by a program ϕ over some $M \in \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ is regular and has its syntactic monoid in $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$.

Since L can be recognized by a program over a monoid that has bounded k-party communication complexity for some k, it follows from our comments preceding Lemma 2 that $D_k(L) = O(1)$. Since L has a neutral letter part a) of theorem 1 guarantees that L is regular. Now, using part b), L is a regular language with a neutral letter and has bounded k-party complexity so we must have $M(L) \in \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$. □

In fact, this theorem is not the first indication that programs over $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ are weak. It was shown in [26], building on work of [27] that this variety has the *polynomial-length property* in the sense that any program ϕ over $M \in \mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ is equivalent to a program ψ over M that has polynomial length. In contrast, a monoid M is said to be *universal* if any language can be recognized by some program over M of possibly exponential length. A simple counting

argument shows that any monoid having the polynomial length property cannot be universal and there are indications that the two notions are in fact dual [27].

A direct application of Corollary 3 and Lemma 5, the following subvarieties of $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ also have the strong CBP.

Corollary 7. *The following varieties have the strong CBP:* $\mathbf{G_{nil}}, \mathbf{J}, \mathbf{DA}$ *and* $\mathbf{G_{nil,k}}$ *for any* $k \geq 1$.

It is possible to exhibit varieties that do not have even the weak CBP. In particular, a main result of [2] can be rephrased algebraically as stating that the variety of aperiodic monoids does not have the weak CBP. Furthermore, Barrington showed that polynomial length programs over any non-solvable group are as powerful as NC^1 which in particular contains languages with a neutral letter which are not regular. We thus obtain:

Theorem 8. *If* $\mathbf{V}$ *is a variety containing a non-solvable group or containing the variety* $\mathbf{A}$ *of aperiodic finite monoids, then* $\mathbf{V}$ *does not have the weak CBP.*

We conjecture that in fact any variety containing a universal monoid fails to have the CBP. In particular, we believe that there are non-regular languages with a neutral letter definable in $\mathbf{\Sigma}_2[Arb]$.

4 An Application to Logic

The study of Crane Beach properties was foremost motivated by logical considerations and we use the results of the preceding section to describe fragments of $\mathbf{FO}^+\mathbf{MOD}$ which have the CBP.

For any $s \geq 0$ and $p \geq 2$ we denote as $\mathbf{\Sigma}_1^{(s,p)}$ the fragment of $\mathbf{FO}^+\mathbf{MOD}$ which consists of sentences of the form $\exists^{t,i \pmod p}(x_1, \ldots, x_k)\ \ \phi(x_1, \ldots, x_k)$ where ϕ is quantifier-free and where the quantifier $\exists^{t,i \pmod p}$, which ranges over k-tuples of variables, is true if the number of k-tuples satisfying ϕ is either equal to $t < s$ or congruent to i modulo p. Note that if $s = 0$, this fragment does not have the ability to simulate an existential quantifier. For a sentence $\phi \in \mathbf{B}\,\mathbf{\Sigma}_1\,(s,p)$, we define the *maximum arity* of ϕ to be the maximum arity of any of the quantifiers in ϕ.

The expressive power of the $\mathbf{\Sigma}_1^{(s,p)}$ fragment was studied in depth in [24]. In particular, it is recalled that a language L is definable in $\mathbf{\Sigma}_1^{(s,p)}[<]$ iff it is regular and the syntactic monoid of L lies in $\mathbf{J} \vee \mathbf{G_{nil}}$, the variety generated by $\mathbf{J}$ and $\mathbf{G_{nil}}$. This is not surprising given the existing combinatorial descriptions of languages in $\mathbf{L}(\mathbf{J} \vee \mathbf{G_{nil}})$ which we describe next.

We say that a word $u = u_1 \ldots u_k$ is a *subword* of w if w can be factorized as $w = \Sigma^* u_1 \Sigma^* \ldots \Sigma^* u_k \Sigma^*$ and we denote as $\binom{w}{u}$ the number of such factorizations. A language L is *piecewise-testable* if there exists a k such that membership of $w \in L$ depends only on the set of subwords of length at most k that occur in w. It is not hard to see that L is piecewise testable language iff it is definable in

$\mathbf{B}\,\mathbf{\Sigma}_1[<]$. A theorem of Simon moreover shows that L is piecewise testable iff its syntactic monoid lies in $\mathbf{J}$.

Similarly, we say that a language L *counts subwords of length k modulo p* if membership of a word w in L only depends on the values $\binom{w}{u_1}, \ldots, \binom{w}{u_n}$ modulo p for some words $u_1, \ldots, u_n$ of length at most k. Again, it is not hard to see that L is of that form iff it can be defined in $\mathbf{\Sigma}_1^{(0,p)}[<]$. It can also be shown that this class corresponds to languages with syntactic monoids in $\mathbf{G_{nil,k}}$.

We say that two words v and w have the same number of subwords of length k *up to threshold s and modulo p* and write $v \sim_{k,s,p} w$ if for any u of length at most k we have either $\binom{v}{u} \leq s$ and $\binom{v}{u} = \binom{w}{u}$ or $\binom{v}{u} > t$ and $\binom{v}{u} = \binom{w}{u}$ modulo p. It can be shown that this relation is a congruence on Σ^* and that a regular language L has a syntactic monoid in $\mathbf{J} \vee \mathbf{G_{nil}}$ iff L is a union of $\sim_{k,s,p}$-classes for some k, s, p (see e.g. [24]). Straubing also establishes the following [24]:

Lemma 9 ([24]). *If L is a regular language definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[Arb]$ then L is in fact definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[\mathcal{R}eg]$.*

Note that the above statement does not assume that the language has a neutral letter. A stronger result can be proved under that hypothesis.

Lemma 10. *If L is a regular language with a neutral letter and is definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[Arb]$ then it is in fact definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[<]$.*

Proof (Sketch). By lemma 9 we already know that L is definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[\mathcal{R}eg]$ and we will furthermore show that it lies in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[<]$ by proving that its syntactic monoid $M(L)$ belongs to $\mathbf{J} \vee \mathbf{G_{nil}}$. Let ϕ be the $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[\mathcal{R}eg]$ defining L and let k be the maximum arity of any quantifier in ϕ. Further let r be the maximum of s, p and any t or q occurring in any $\delta_{n,t,q}$ and any $\kappa_{n,t,q}$ (see section 2.2) needed to express the regular numerical predicates occurring in ϕ.

Let $e \in \Sigma$ be the neutral letter for L and let v and w be two words in $(\Sigma - \{e\})^*$ such that $v \sim_{k,s,p} w$. In particular, the length of v and w are equal up to treshold s and modulo p. We claim that v is in L iff w is in L. This suffices to establish the result because L is then a union of $\sim$ classes and thus $M(L)$ lies in $\mathbf{J} \vee \mathbf{G_{nil}}$.

To establish the claim, we construct words $V = e^{r!-1}v_1e^{r!-1}\ldots e^{r!-1}v_{|v|}e^{r!-1}$ and $W = e^{r!-1}w_1e^{r!-1}\ldots e^{r!-1}w_{|w|}e^{r!-1}$. Since e is neutral v (resp. w) is in L iff V (resp. W) is in L. Let $\psi(x_1, \ldots, x_k)$ be any quantifier-free formula constructed from the content predicates, the order predicate $<$ and $\delta_{n,t,q}$ or $\kappa_{n,t,q}$ predicates with $t, q \leq r$. We claim that the number of tuples $(x_1, \ldots, x_k)$ such that $\psi(x_1, \ldots, x_k)$ is true on V is equal up to threshold s and modulo q to the number of tuples such that ψ is true on W. This is sufficient to show that V and W satisfy the same sentences in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}$ sentences and are thus either both in L or both not in L.

We can rewrite $\psi(x_1, \ldots, x_k)$ as a boolean combination of formulas

$$\bigwedge_i (Q_{a_i}x_i \wedge \kappa_{n_i,t_i,q_i}x_i \wedge \bigwedge_{i \neq j}((x_i * x_j) \wedge x_i\delta_{n_{ij},t_{ij},q_{ij}}x_j)$$

with $a_i \in \Sigma$; $t_i, t_{ij}, q_i, q_{ij} \leq r$ and $*$ is one of $\{<, >, =\}$.

Suppose for simplicity that for $i < j$ the above formula requires $x_i < x_j$. To evaluate the number of tuples satisfying ψ over V, we first choose values for the *non-neutral positions* x_i, i.e. the x_i such that $i \in I$, such that position x_i in V holds the desired non-neutral letter a_i. We denote as u the word of length $|I| \leq k$ formed by these letters. Note that each non-neutral x_i is congruent to 0 modulo $r!$ because non-neutral letters only occur at such positions. Hence, we can discard any $\delta_{n,t,q}$ predicates involving two such x_i and any $\kappa_{n,t,q}$ involving one such x_i.

Any tuple of non-neutral positions $\overline{D}$ in V corresponds naturally to a tuple of positions $\overline{d} = (d_1, \ldots, d_{|I|})$ in the original word v which specifies a subword of v of length at most $|I| \leq k$. Note that if $|I| = k$ then the number of tuples satisfying $\psi(x_1, \ldots, x_k)$ is simply the number of occurrences of the subword $u = a_1 \ldots a_k$ in v which is equal up to threshold s and modulo p to $\binom{w}{u}$.

We define the *mod p signature* of $\overline{d}$ as the vector $(d_1, d_2, \ldots, d_{|I|}) \mod p$. The sum of the number of $\overline{d}$ of any possible signature is again $\binom{v}{u}$.

The possibilities for choosing the neutral positions x_j, i.e. those such that $x_j \in \overline{I}$, are constrained in a number of ways. First, these positions must be chosen such that they hold the neutral letter e. Secondly, each x_j must satisfy some predicates $\kappa_{n_j,t_j,q_{ij}}$ and $\delta_{n_{ij},t_{ij},q_{ij}}$. But it is easy to see that for any fixed set D of non-neutral positions, the number of choices (threshold s and modulo p) for the neutral positions depends only on the signature of d because we are only concerned with the number modulo p of blocks of neutral letters occurring between two non-neutral positions and this information is provided by the signature.

Hence, the total number of tuples (treshold s, modulo p) $(x_1, \ldots, x_k)$ satisfying $\psi(x_1, \ldots, x_k)$ over V is some function of the number of occurrences threshold t modulo p of u in v. Since the same holds for W and $v \sim_{k,s,p} w$ we have that V and W and thus v and w either both lie in L or lie both outside L. □

In fact, this lemma resolves an open problem of [24].

Lemma 11. *If L is definable by a boolean combination of sentences of the form $\phi : \exists^{(t,i \pmod p)}(x_1, \ldots, x_k).\ \psi(x_1, \ldots, x_k)$ in which the quantifier has arity at most k then the $k+1$-party communication complexity of L is $O(1)$.*

Proof. Suppose $k+1$ parties want to verify if an input w belongs to L, i.e. check that $w \models \phi$. They simply need to count the number of k-tuples $(x_1, \ldots, x_k)$ that satisfy $\psi(x_1, \ldots, x_k)$ over w up to threshold t and modulo p. For any fixed tuple, the truth value of $\psi(x_1, \ldots, x_k)$ depends only on the letters of w in these k positions. In the $k+1$-party game, each k-tuple of input positions is fully accessible to at least one player so we can construct a simple protocol of cost $(k+1) \cdot (\lceil \log p \rceil + \lceil \log t \rceil)$ in which each player sends up to threshold t and modulo p the number of tuples that he sees satisfying ψ (although they must agree on some scheme that avoids counting twice a tuple seen by more than one of them). □

Combining this result with theorem 1 and lemma 10 we obtain:

Theorem 12. *The boolean closure of* $\mathbf{\Sigma}_1^{(s,p)}$ *has the CBP.*

Proof. Let L be a language with a neutral letter definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[Arb]$. By lemma 11, L has bounded k-party communication complexity for some k and thus, by theorem 1, L is regular. Finally, by lemma 10 any regular language with a neutral letter definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[Arb]$ is in fact definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(s,p)}[<]$. □

As a corollary, we get an alternative proof of the following theorem of [2].

Corollary 13. *The boolean closure of* $\mathbf{\Sigma}_1$ *has the CBP.*

Proof. Let L be a language with a neutral letter definable in $\mathbf{B}\,\mathbf{\Sigma}_1[Arb]$. By Theorem 12, L is regular and its syntactic monoid $M(L)$ lies in $\mathbf{J} \vee \mathbf{G_{nil}}$. Moreover, the only regular languages with a neutral letter definable in $\mathbf{FO}[Arb]$ are those whose syntactic monoid is aperiodic and so $M(L) \in \mathbf{A}$. A simple semigroup-theory argument shows that the intersection of $\mathbf{J} \vee \mathbf{G_{nil}}$ and $\mathbf{A}$ is $\mathbf{J}$. □

Readers familiar with the Ehrenfeucht-Fraïssé approach of [2] might find it surprising that our alternative proof seems to avoid the use of Ramsey-theoretic arguments. In fact, the communication complexity result of [10] which is so crucial in our method relies on the Ramsey-like theorem of Hales-Jewett.

Of course our main theorem also specializes to the other extreme case of $\mathbf{\Sigma}^{(s,p)}$ sentences in which existential and universal quantifiers do not appear: $\mathbf{B}\,\mathbf{\Sigma}_1^{(0,p)}$. Moreover, the maximum arity of the quantifiers can be preserved.

Corollary 14. *For each* $k \geq 1$, $\mathbf{B}(\mathbf{\Sigma}_1^{(0,p)})$ *of maximum arity* k *has the CBP.*

Proof. One can show that programs over $\mathbf{G_{nil,k}}$ have exactly the same expressive power as $\mathbf{B}\,\mathbf{\Sigma}^{(0,p)}[Arb]$ of maximum arity k while morphisms over $\mathbf{G_{nil,k}}$ have exactly the same expressive power as $\mathbf{B}\,\mathbf{\Sigma}^{(0,p)}[<]$ of maximum arity k (see lemma 15 in the appendix). The statement of the corollary then follows simply by the fact that each $\mathbf{G_{nil,k}}$ has the strong CBP. □

5 Conclusion

The algebraic perspective on the Crane Beach Conjecture allows for a more detailed study of the fine line separating computational models or logical fragments which possess a Crane Beach-like property. Moreover it enables a systematic use of the powerful communication complexity results of [10]. Theorem 12 about the Crane Beach property of $\mathbf{\Sigma}_1^{(s,p)}$ can be obtained by an ad hoc argument using Ramsey theory and Ehrenfeucht-Fraïssé games reminiscent of the techniques for the $\mathbf{\Sigma}_1$ case [2]. The proof given here is considerably simpler if less transparent.

Our results about varieties of monoids exhibiting the Crane Beach property also provide a nice complement to previously existing work comparing the power

of programs and morphisms over finite monoids in the presence of a neutral letter. Programs over monoids are generally much more epressive than morphism. This extra power of programs stems from (or is limited by) three sources: the algebraic structure of their underlying monoid, their length and their degree of non-uniformity. However, in the presence of a neutral letter the advantages of programs over monoids can disappear:

- results of [4] show that length $\Omega(n \log\log n)$ is necessary to recognize non-regular languages with a neutral letter, regardless of the underlying monoid or the degree of non-uniformity;
- results of [2, 20] show that polynomial length programs which are too uniform cannot break the regular barrier, as long as the monoid is solvable.
- Our results complete the picture: programs over monoids whose structure is unsophisticated cannot recognize non-regular languages with a neutral letter, regardless of their length or their degree of non-uniformity.

We have used the algebraic point of view on the CBP to extend the result of [2] on the CBP for $\mathbf{\Sigma}_1$ to $\mathbf{\Sigma}_1^{(s,p)}$. Some of the more general results of Section 3, however, do not seem to have such simple logical applications. In particular, we have shown that a number of important subvarieties of $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ have the CBP but these do not capture any significant logical fragment of $\mathbf{FO}^+\mathbf{MOD}$. The case of the variety **DA** is particularly intriguing, given its importance in applications of semigroup theory to complexity [28]. Programs over **DA** have the same expressive power as decision trees of bounded-rank [11] and so this model also has the CBP. The languages recognized via morphism by monoids in **DA** are exactly those definable by both a $\mathbf{\Sigma}_2[<]$ and a $\mathbf{\Pi}_2[<]$ sentence and also those definable in the restriction of $\mathbf{FO}[<]$ to sentences using only two variables ($\mathbf{FO}_2[<]$) but programs over **DA** are not known to correspond to the intersection of $\mathbf{\Sigma}_2[Arb]$ and $\mathbf{\Pi}_2[Arb]$ or to $\mathbf{FO}_2[Arb]$. The latter two classes are important candidates for logical fragments of **FO** that may possess the CBP.

References

1. D. A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.
2. D. A. M. Barrington, N. Immerman, C. Lautemann, N. Schweikardt, and D. Thérien. First-order expressibility of languages with neutral letters or: The Crane Beach conjecture. *J. Comput. Syst. Sci.*, 70(2):101–127, 2005.
3. D. A. M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *J. Comput. Syst. Sci.*, 41(3):274–306, 1990.
4. D. A. M. Barrington and H. Straubing. Superlinear lower bounds for bounded-width branching programs. *J. Comput. Syst. Sci.*, 50(3):374–381, 1995.
5. D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990.
6. D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, 1988.
7. C. Behle and K.-J. Lange. **FO**-uniformity. In *Proc. 21st Conf. on Computational Complexity (CCC'06)*, 2006.

8. M. Benedikt and L. Libkin. Expressive power: The finite case. In *Constraint Databases*, pages 55–87, 2000.
9. A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *Proc. 15th ACM Symp. on Theory of Computing (STOC'83)*, pages 94–99, 1983.
10. A. Chattopadhyay, A. Krebs, M. Koucký, M. Szegedy, P. Tesson, and D. Thérien. Functions with bounded multiparty communication complexity. Submitted, 2006.
11. R. Gavaldà and D. Thérien. Algebraic characterizations of small classes of boolean functions. In *Proc. of Symp. on Theoretical Aspects of Comp. Sci. (STACS'03)*, 2003.
12. Y. Gurevich and H. Lewis. A logic for constant-depth circuits. *Information and Control*, 61(1):65–74, 1984.
13. N. Immerman. Languages that capture complexity classes. *SIAM J. Comput.*, 16(4):760–778, 1987.
14. E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.
15. L. Libkin. *Elements of Finite Model Theory*. Springer Verlag, 2004.
16. P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.
17. P. Péladeau. Formulas, regular languages and boolean circuits. *Theor. Comput. Sci.*, 101(1):133–141, 1992.
18. J.-E. Pin. Syntactic semigroups. In *Handbook of language theory*, volume 1, chapter 10, pages 679–746. Springer Verlag, 1997.
19. J.-F. Raymond, P. Tesson, and D. Thérien. An algebraic approach to communication complexity. *Lecture Notes in Computer Science (ICALP'98)*, 1443:29–40, 1998.
20. A. Roy and H. Straubing. Definability of languages by generalized first-order formulas over $(\mathbf{N}, +)$. In *23rd Symp. on Theoretical Aspects of Comp. Sci. (STACS'06)*, pages 489–499, 2006.
21. R. Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proc. 19^{th} ACM STOC*, pages 77–82, 1986.
22. H. Straubing. *Finite Automata, Formal Logic and Circuit Complexity*. Boston: Birkhauser, 1994.
23. H. Straubing. When can one monoid simulate another? In *Algorithmic Problems in Groups and Semigroups*, pages 267–288. Birkhäuser, 2000.
24. H. Straubing. Languages defined by modular quantifiers. *Information and Computation*, 166:112–132, 2001.
25. H. Straubing, D. Thérien, and W. Thomas. Regular languages defined by generalized quantifiers. *Information and Computation*, 118:289–301, 1995.
26. P. Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, McGill University, 2003.
27. P. Tesson and D. Thérien. The computing power of programs over finite monoids. *Journal of Automata, Languages and Combinatorics*, 7(2):247–258, 2002.
28. P. Tesson and D. Thérien. Diamonds are forever: the variety **DA**. In *Semigroups, Algorithms, Automata and Languages*. WSP, 2002.
29. P. Tesson and D. Thérien. Complete classifications for the communication complexity of regular languages. *Theory of Computing Systems*, 38(2):135–159, 2005.
30. P. Tesson and D. Thérien. Restricted two-variable sentences, circuits and communication complexity. In *Proc. 32nd Int. Conf. on Automata, Languages and Programming (ICALP'05)*, pages 526–538, 2005.

31. P. Tesson and D. Thérien. Bridges between algebraic automata theory and complexity theory. *The Computational Complexity Column, Bull. EATCS*, 88:37–64, 2006.
32. P. Tesson and D. Thérien. Logic meets algebra: the case of regular languages. 2006. Submitted.
33. W. Thomas. *Languages, Automata and Logic*, volume III, chapter 7, pages 389–455. Springer, 1997.

Appendix

Lemma 2. *Let $k \geq 2$ be some integer and let $f : \mathbb{N} \to \mathbb{N}$ be such that $f = O(\log^c n)$ for some $c \geq 0$. The class $\mathbf{V}$ of monoids M such that $D_k(M) = O(f)$ is a program-variety.*

Proof. First, note that $\mathbf{V}$ forms a variety because the communication complexity of the direct product $M \times N$ is at most the sum of the complexities of M and N and so $D_k(M) = O(f)$ and $D_k(N) = O(f)$ implies $D_k(M \times N) = O(f)$. Similarly, the communication complexity of a submonoid N of M or a morphic image N of M is at most the comunication complexity of M.

We further need to show that any monoid M whose multiplication can be simulated by a program of length $O(n^d)$ over a monoid $N \in \mathbf{V}$ is in fact in $\mathbf{V}$ and so it suffices to show that the communication complexity of M is $O(f)$. As we noted previously, the communication complexity of evaluating a program of length ℓ over N is at most the complexity of evaluating the product of ℓ elements of N and thus $D_k(M) = O(f(O(n^d))$ which is simply $O(f)$ because f is polylogarithmic.

Lemma 15. *A language L is definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(0,p)}[Arb]$ of maximum arity k iff L can be recognized by a program over a nilpotent group of class k.*

A language L is definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(0,p)}[<]$ of maximum arity k iff L can be recognized by a morphism over a nilpotent group of class k.

Proof (Sketch). We first want to argue that a language L is definable in $\mathbf{B}(\mathbf{\Sigma}_1^{(0,p)})$ with maximum arity k iff it can be recognized by a program over a nilpotent group of class k. Let us first show that the language accepted by a program ϕ over a nilpotent group G of class k can be defined in $\mathbf{B}\,\mathbf{\Sigma}^{(0,p)}[Arb]$ of maximum arity k.

By the combinatorial description of languages recognized by nilpotent groups of class k, the output of ϕ over G on an input w can be determined by counting modulo some p the number of occurrences of each subword of length at most k in the string of G-elements $\phi(w)$. Note that each element of this string is in fact the output of some instruction which depends on the value of exactly one input position. Thus, this counting can be realized with a $\mathbf{\Sigma}^{(0,p)}$-sentence of arity at most k.

For the converse, we use the following result from [27].

Lemma 16. *Let G be a finite group and inductively define $G_1 = G$ and $G_{i+1} = [G, G_i]$. Any function from $\Sigma^c \to G_c$ can be computed by a program over G.*

Moreover, we will need the simple fact that for any $p \geq 2$ there exists a nilpotent group G of class k such that G_k is non-trivial and contains an element g of order p.

We want to show that any language definable in $\mathbf{B}\,\mathbf{\Sigma}_1^{(0,p)}[Arb]$ can also be recognized by a program over this G. It is sufficient to show that there is a

program which on input w counts modulo p the number of tuples $(x_1, \ldots, x_k)$ that satisfy a quantifier free formula $\psi(x_1, \ldots, x_k)$ over w. For each such k-tuple, the truth value of $\psi(x_1, \ldots, x_k)$ is a function of the indices $x_1, \ldots, x_k$ and the content of these positions in w. The latter defines a function from Σ^k into $\{0, 1\}$ and so by the above lemma, there exists a program which outputs the element g of order p when ψ is true and the identity element 1_G otherwise. Concatenating these programs for the n^k possible values of the tuple results in a program whose output is the identity element iff the number of tuples satisfying $\psi(x_1, \ldots, x_k)$ is divisible by p.