Monoids and Computations

Pascal Tesson and Denis Thérien

School of Computer Science, McGill University
{ptesso,denis}@cs.mcgill.ca

Abstract. This contribution wishes to argue in favour of increased interaction between experts on finite monoids and specialists of theory of computation. Developing the algebraic approach to formal computations as well as the computational point of view on monoids will prove to be beneficial to both communities. We give examples of this two-way relationship coming from temporal logic, communication complexity and boolean circuits. Although mostly expository in nature, our paper proves some new results along the way.

1 Introduction

This paper is an invitation to members of the semigroup community to increase their interaction with colleagues working in the theory of computation. We are convinced that computational theorists generate ideas that can be extremely profitable in the study of semigroups and that, in turn, semigroup theorists can adapt their techniques to problems in theoretical informatics. Increased collaboration will be beneficial to both groups.

Our first point is that the notion of formal computation is fascinating both from a strictly mathematical point of view and also from an epistemological one. Investigations of computational issues give rise to questions that satisfy the most demanding criteria of mathematical "beauty" and "elegance". The bonus is that it is not so far-fetched to view the human brain as a computational device so that our whole understanding of the universe is mediated through some sort of "computation".

Since Gödel and Turing, formal computations have been recognized as legitimate objects of investigations and their theory serves as the foundations of computer science (which could hardly be called a "science" if this theory did not exist). This last fact is recognized by granting agencies all over the world and computation theorists are certainly among the mathematicians with the easiest access to funding. The idea of computation is now being exported to other scientific disciplines: one can for instance argue that bioinformatics is an attempt to view DNA as a "computer" and exploit this perspective. This will be much more fundamental to the development of bioinformatics than the strict usage of gigantic computing power to handle massive sets of data.

Our second point is that semigroup theory could and should be among the areas of classical mathematics that have the most to say about computations, just like logic, probability and combinatorics. Looking at finite semigroups as computational devices that recognize languages has proved to be extremely fruitful both for the semigroup community and for the formal language community and it is fair to say that no one doing research on finite semigroups can ignore their computational flavour. Moreover, *all* computer science students are introduced to formal computations via the deterministic finite automaton model, which is close enough to a finite semigroup that we can disregard the distinction. Unfortunately, this association is short-lived: theoretical computer science quickly abandons the DFA model to study more general devices and the major conferences in the field hardly have any contributions dealing with questions on finite automata. Similarly, semigroup theory does not seem to have much interest in facing the mathematically difficult challenges arising from computations.

Our last point is that this state of affairs could and should change. Semigroups constitute the most natural computation devices and our community should explicitly include in its mission the development of algebraic tools that can be used in the investigations of computations. Semigroup theorists can both benefit and contribute by increasing their interaction with theoretical computer scientists.

In this paper, we will sketch three areas where such interaction has proved fruitful: temporal logic, communication complexity and boolean circuits. In these (very incomplete) notes we will try to point out both the gains in our understanding of semigroups as well as the new light brought from algebra to computation theorists. As will be seen, key questions are still open and there is great potential for creative members of our community to impact non-trivially on a mathematical area of prime importance. We generally follow standard notation as can be found e.g. in [19, 20] and, unless specified otherwise, we refer the reader to that book for notions that are not explicitly defined here.

2 Temporal Logic and Block Products

Temporal logic (TL) is a formalism commonly used to analyze behaviours of dynamic processes [12]. We will consider this framework in the context of finite strings. The seminal work of Kamp [15] established that a language can be described by a TL formula if and only if it is star-free, i.e. if and only if its syntactic monoid is aperiodic. A small fragment of the full logic was algebraically analyzed in [11] and complete algebraic characterizations of the levels of the natural TL hierarchies were given in [35, 34]. In this section, we outline the approach of [34] and show how it impacts on our understanding of monoids.

The key link between two-sided temporal logic and finite monoids rests on the notion of block-product first introduced in [24]. The *block product* of two monoids S and T, denoted $S \square T$ is a two-sided version of the wreath product: its underlying set is $S^{T \times T} \times T$ with the multiplication given by

$$(f_1, t_1) \cdot (f_2, t_2) = (g, t_1 t_2)$$

where $g : T \times T \to S$ is given by $g(x, y) = f_1(x, t_2 y) f_2(x t_1, y)$.

As we shall see the investigations on temporal logic have shed new light on this operation. We begin by giving a congruence description of block products, based on [32].

Let M be a finite monoid and A a finite alphabet. We define a congruence on A^* by letting $x \equiv_{M,A} y$ if and only if $\phi(x) = \phi(y)$ for any morphism $\phi : A^* \to M$. We will usually abbreviate this as $\equiv_M$ when the alphabet used is clear from context. It is easy to see that $\equiv_M$ is the coarsest congruence on A^* such that $A^*/\equiv_M$ belongs to the variety generated by M. We next describe the congruence $\equiv_{S\Box T}$ on A^*: we set $x \equiv_{S\Box T} y$ if and only if $x \equiv_T y$ and $h(x) \equiv_S h(y)$ where $h(x_1 \dots x_n) = X_1 \dots X_n$ with $X_i = ([x_1 \dots x_{i-1}]_{\equiv_T}, x_i, [x_{i+1} \dots x_n]_{\equiv_T})$ ($h(x)$ and $h(y)$ are therefore words over the alphabet $A^*/\equiv_T \times A \times A^*/\equiv_T$). We leave it as an exercise to verify that indeed $x \equiv_{S\Box T} y$ if and only if $\phi(x) = \phi(y)$ for every morphism $A^* \to S \Box T$.

As usual, we extend the operation $\Box$ to varieties by defining $\mathbf{V} \Box \mathbf{W} = \{M : M \prec S \Box T$ for some $S \in \mathbf{V}$ and $T \in \mathbf{W}\}$. In view of the above, $A^*/\alpha \in \mathbf{V} \Box \mathbf{W}$ if and only if $\alpha \supseteq \equiv_{S\Box T}$ for some S in $\mathbf{V}$ and T in $\mathbf{W}$. A well known result about block product decompositions is that the variety $\mathbf{A}$ of aperiodic monoids is the smallest variety satisfying $\mathbf{V} = \mathbf{J_1} \Box \mathbf{V}$, where $\mathbf{J_1}$ is the variety of semilattices.

It was immediately noticed that $\Box$ is not associative and that in general $(\mathbf{U} \Box \mathbf{V}) \Box \mathbf{W} \subsetneq \mathbf{U} \Box (\mathbf{V} \Box \mathbf{W})$ but only the second type of bracketing (the "stronger" one) was ever considered before [29]. In that paper, it was established that the first bracketing is the one to be considered in relation to temporal logic.

In [33], using Ehrenfeucht-Fraïssé games, Thérien and Wilke showed that a language can be decribed in so-called *unary temporal logic* (UTL) iff its syntactic monoid belongs to the variety $\mathbf{DA}$. Later building on the work of [34], Straubing and Thérien [29] proved that $\mathbf{DA}$ is in fact the smallest variety satisfying $\mathbf{V} = \mathbf{V} \Box \mathbf{J_1}$. This induces a parametrization for $\mathbf{DA}$ as

$$\mathbf{DA_k} = \underbrace{(\dots(\mathbf{J_1} \Box \mathbf{J_1}) \Box \dots) \Box \mathbf{J_1}}_{k \text{ times}}.$$

We now proceed to show that this parametrization actually corresponds to the natural parametrization of UTL.

In (two-sided) unary temporal logic, formulae are constructed using subsets of the alphabet A, Boolean connectives and the temporal operators Eventually-in-the-future $\diamondplus$ and Eventually-in-the-past $\diamondminus$. These formulas are interpreted at positions in strings according to the following rules. Let $x = x_1 \dots x_n$ and $i \in [n]$:

$(x,i) \models B$ (where $B \subseteq A$) if and only if $x_i \in B$;
$(x,i) \models \diamondplus\phi$ if and only if there exists $i < j \leq n$ such that $(x,j) \models \phi$;
$(x,i) \models \diamondminus\phi$ if and only if there exists $1 \leq j < i$ such that $(x,j) \models \phi$;
$(x,i) \models \phi \vee \psi$ if and only if $(x,i) \models \phi$ or $(x,i) \models \psi$;
$(x,i) \models \phi \wedge \psi$ if and only if $(x,i) \models \phi$ and $(x,i) \models \psi$;
$(x,i) \models \neg\phi$ if and only if not $(x,i) \models \phi$.

To describe languages, we use formulas which are Boolean combinations of subformulas, each of which has either $\diamondplus$ or $\diamondminus$ as top level operator. Note that consequently, if $B \subseteq A$, the subformula B by itself is not associated to a language. To associate languages to formulas, we proceed by extending the above semantics to include:

$(x, 0) \models \diamondplus\phi$ if and only if there is $0 < j \leq n$ such that $(x, j) \models \phi$;
$(x, n+1) \models \diamondminus\phi$ if and only if there is $1 \leq j < n+1$ such that $(x, j) \models \phi$;

and we define $L(\diamondplus\phi) = \{x : (x, 0) \models \diamondplus\phi\}$ and $L(\diamondminus\phi) = \{x : (x, n+1) \models \diamondminus\phi\}$. Boolean operators have their natural meaning. For example, note that we have:

$$L(\diamondplus B) = L(\diamondminus B) = A^*BA^*$$

$$L(\diamondplus(C \wedge \neg\diamondminus(\neg B))) = B^*CA^*$$

The natural parametrization of UTL is in terms of the number of nested $\diamondplus$ and $\diamondminus$ appearing in a formula. Let us denote by UTL_k the class of languages that can be described by a UTL formula in which at most k $\diamondplus/\diamondminus$ operators appear nested one inside the other.

Theorem 1. *$L \in UTL_k$ iff $M(L) \in \mathbf{DA_k}$.*

Proof. We only sketch the argument as it relies heavily on material already presented in [34]. For $k = 1$, it is well-known that a language has its syntactic monoid in $\mathbf{DA_1} = \mathbf{J_1}$ iff it is a boolean combination of sets of the form A^*BA^* with $B \subseteq A$, and these are easily seen to be exactly what can be expressed with UTL formulas of depth 1. Furthermore, one can assume that the only temporal operator occuring in these formulas is $\diamondplus$ (or symmetrically $\diamondminus$).

For $k > 1$, in one direction it is enough to show that $[x]_{\equiv_{S \Box T}}$ can be expressed by a formula of depth k when $S \in \mathbf{DA_{k-1}}$ and $T \in \mathbf{J_1}$. The non-trivial part is to show that $L = \{y \in A^* : h(y) \equiv_S h(x)\}$ has the property. By the induction hypothesis, there is a formula ϕ of depth $k-1$ on the alphabet $A^*/\equiv_T \times A \times A^*/\equiv_T$ such that $h(y) \models \phi$ iff $h(y) \equiv_S h(x)$. An occurrence of a "letter" $([u]_{\equiv_T}, a, [v]_{\equiv_T})$ in ϕ can be replaced by the subformula $a \wedge \phi_1 \wedge \phi_2$, where ϕ_1 describes $[u]_{\equiv_T}$ using only $\diamondminus$ and ϕ_2 describes $[v]_{\equiv_T}$ using $\diamondplus$ only, to yield a formula of depth k which can be checked to describe L.

For the converse, let ϕ be a UTL formula of depth k. Let us write $x \equiv_k y$ to denote the fact that x and y are congruent in the free A-generated monoid of $\mathbf{DA_k}$. We need to show that $x \models \phi$ iff $y \models \phi$. Let $\phi_1, \ldots, \phi_s$ be the maximal subformulas of depth 1 that appear in ϕ. We observe the following: suppose $u \equiv_1 u'$ and $v \equiv_1 v'$, then $(uav, |u|+1) \models \phi_j$ iff $(u'av', |u'|+1) \models \phi_j$ for each j. This means that the following process is well-defined; replace in ϕ every occurrence of ϕ_j by the set $\{([u]_{\equiv_1}, a, [v]_{\equiv_1}) : (uav, |u|+1) \models \phi_j\} \subseteq A^*/\equiv_1 \times A \times A^*/\equiv_1$. This yields a formula ψ of depth $k-1$ over the new alphabet and it can be shown that $x \models \phi$ iff $h(x) \models \psi$; by the induction hypothesis, $h(x) \equiv_{k-1} h(y)$ implies $h(x) \models \psi$ iff $h(y) \models \psi$, and this in turns implies $y \models \phi$.

In the full temporal logic one also uses the binary temporal operators Until and Since. In the previous theorem, we proved that the nesting of $\diamondplus$ and $\diamondminus$ operators in temporal logic was tightly related to forming block products with $\mathbf{J_1}$ on the algebraic side; similarly, the nesting of Until and Since operators can be shown to be in correspondence with block products involving the variety $\mathbf{R_1} \vee \mathbf{L_1}$ defined by the equations $x^2 = x$ and $xyxzx = xyzx$. It can further be shown that any TL formula is equivalent to one in which no $\diamondplus/\diamondminus$ operator appears in the scope of an Until or a Since. This has the remarkable algebraic consequence that

$$(\mathbf{DA} \square (\mathbf{R_1} \vee \mathbf{L_1})) \square \mathbf{J_1} = \mathbf{DA} \square (\mathbf{R_1} \vee \mathbf{L_1}).$$

If we define $\mathbf{V_0} = \mathbf{DA}$ and $\mathbf{V_k} = \mathbf{V_{k-1}} \square (\mathbf{R_1} \vee \mathbf{L_1})$ we have the following:

Theorem 2 ([34]). *L can be defined by a TL formula with at most k nested Until/Since operators iff $M(L) \in \mathbf{V_k}$.*

All varieties of the form $\mathbf{DA_k}$ and $\mathbf{V_k}$ are known to be decidable [34] but the decision procedures based on the category approach (see [36, 32]) are terribly inefficient. In view of the connection between these varieties and the natural hierarchies appearing in temporal logic, we suggest it is an important question to design better algorithms to test membership for these varieties.

We close this section by observing the following. $\mathbf{R_1}$ and $\mathbf{L_1}$ both have the property that their closure under weakly bracketed iterated block product is equal to $\mathbf{A}$, the variety of aperiodic monoids. These are the only two minimal varieties with this property; this follows from the easily proved fact that $(\mathbf{DA} \square \mathbf{G}) \square (\mathbf{J} \square \mathbf{G}) = \mathbf{DA} \square \mathbf{G}$, where $\mathbf{J}$ is the variety of J-trivial monoids and $\mathbf{G}$ is the variety of all groups.

3 Communication Complexity

We refer the reader to the book of Kushilevitz and Nisan [16] for a fascinating introduction to communication complexity and its countless applications to computation theory.

In the classical communication complexity model, two players, Alice and Bob, wish to collaborate to compute a function $f(x, y)$. Alice is given $x \in X$ while Bob is given $y \in Y$ and we assume that they each have unlimited computing power. In order to compute f, Alice and Bob exchange information according to some fixed, previously agreed upon protocol and the last bits exchanged should encode $f(x, y)$. The communication complexity of f, denoted $D(f)$ is the worst-case number of bits exchanged on any pair $(x, y) \in X \times Y$ when using an optimal protocol. The model can also be used to study functions $f : A^n \to B$ which are not explicitly given as functions of two variables: for any partition of the n inputs into two sets, we can naturally view f as a function of two inputs and we choose a partition that maximizes the complexity of the ensuing communication problem. In other words, we assume that the input is split in a fixed but adversarial way between Alice and Bob. In general, we will look at the communication complexity of $f : A^* \to B$ and thus consider $D(f) : \mathbb{N} \to \mathbb{N}$ as a function of the input length and study its asymptotic behaviour.

Specifically, we will consider the problem of computing multiplication in some finite monoid M: Alice and Bob receive n elements $m_1, m_2, \ldots, m_n$ and their goal is to compute their product in M. In this case, the most unfavorable input partition is easily seen to be that where Alice is only given the odd-indexed m_i while Bob receives the even-indexed m_i (or vice versa). We denote by $D(M)$ the communication complexity of this problem.

We will also consider two variants of this model. In the simultaneous model, no interaction is allowed: Alice and Bob both send a message to some trusted referee having no access to the input and who must be able to deduce the value of f based solely on the communication that he received. In the probabilistic model, we assume that each player has access to a private source of random bits and require that the result of the protocol be correct with probability at least $3/4$ for any input. We denote by $D^{||}(M)$ and $R(M)$ respectively, the simultaneous and probabilistic communication complexity of computing multiplication in M.

It can easily be verified that for any increasing $f : \mathbb{N} \to \mathbb{N}$, the class $\{M : D(M) = O(f(n))\}$ forms a variety and similarly for $D^{||}(M)$ and $R(M)$. Surprisingly, there are very few (distinct) such varieties and the question was completely resolved in [31]. Let **Com** denote the variety of commutative monoids, **DO** be the variety of monoids whose regular $\mathcal{D}$-classes form orthodox subsemigroups, $\mathbf{V} = \mathbf{DO} \cap \overline{\mathbf{Ab}}$ the variety of monoids in **DO** whose subgroups are all Abelian and **W** the variety of monoids in **DO** satisfying $exwyf = ewxyf = exywf$ for any x, y and any idempotents e, f with $e \leq_{\mathcal{J}} w$ and $f \leq_{\mathcal{J}} w$.

Lemma 1. $\mathbf{W} \subseteq \mathbf{V}$.

Proof. We know that **W** is contained in **DO** and if u, v are elements of a subgroup of M with identity $e = u^\omega = v^\omega$, then by definition of **W**, we have

$$uv = euve = evue = vu.$$

These four varieties suffice to characterize the deterministic, probabilistic and simultaneous communication complexity of any finite monoid:

Theorem 3 **([31]).** *For any finite monoid M:*

$$D(M) = \begin{cases} O(1) & \text{if and only if } M \in \mathbf{Com}; \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{V} - \mathbf{Com}; \\ \Theta(n) & \text{otherwise.} \end{cases}$$

$$D^{||}(M) = \begin{cases} O(1) & \text{if and only if } M \in \mathbf{Com}; \\ \Theta(\log n) & \text{if and only if } M \text{ is in } \mathbf{W} - \mathbf{Com}; \\ \Theta(n) & \text{otherwise.} \end{cases}$$

$$R(M) = \begin{cases} O(1) & \textit{if and only if } M \in \mathbf{Com}; \\ \Theta(\log\log n) & \textit{if and only if } M \textit{ is in } \mathbf{W} - \mathbf{Com}; \\ \Theta(\log n) & \textit{if and only if } M \textit{ is in } \mathbf{V} - \mathbf{W}; \\ \Theta(n) & \textit{otherwise.} \end{cases}$$

In turn, this theorem yields tight bounds on the communication complexity of any regular language in each of the three models and these results help explain why certain regular languages have historically been key examples in communication complexity. Such insight on these models would not have been possible without the use of semigroup theoretic ideas.

The variety $\mathbf{W}$ is not so well-known but Theorem 3 establishes its computational significance. We provide next some characterizations of it.

For a word $u \in A^*$ and a letter $a \in A$, we denote by $|u|_a$ the number of occurrences of a in u and by $\alpha_t(u)$ the $|A|$-dimensional vector giving for each $a \in A$ the value $|u|_a$ up to threshold t. For any $t \geq 0$, $p \geq 1$, we define the congruence $\approx_{t,p}$ on A^* by setting $x \approx_{t,p} y$ if and only if

- $|x|_a \equiv |y|_a \pmod{p}$ for each $a \in A$;
- $\alpha_t(x) = \alpha_t(y)$ and for any factorization $x = x_0 a x_1$ with $|x_0|_a < t$ (resp. $|x_1|_a < t$) there exists a factorization $y = y_0 a y_1$ with $\alpha_t(x_0) = \alpha_t(y_0)$ and $\alpha_t(x_1) = \alpha_t(y_1)$).

Let T_t denote the aperiodic monoid with a single generator and satisfying $m^{t+1} = m^t$. We now define $x \equiv_{\ell_1 \Box \mathbf{A_{com,t}}} y$ if $x \equiv_{T_t} y$ and for every $x = x_0 a x_1$ with either $[x_0 a] \not\equiv_{T_t} [x_0]$ or $[a x_1] \not\equiv_{T_t} [x_1]$ there is a factorization $y = y_0 a y_1$ with $y_0 \equiv_{T_t} x_0$ and $y_1 \equiv_{T_t} x_1$.

We denote $\ell\mathbf{1} \Box \mathbf{A_{com}}$ the variety of monoids that divide some $A^*/\equiv_{\ell_1 \Box \mathbf{A_{com,t}}}$. Our presentation of this congruence and our choice of notation stresses that these are exactly the monoids dividing the block product of a locally trivial category and a commutative aperiodic monoid (see e.g. [21]).

Lemma 2. *The following are equivalent for a monoid M:*

1. $M \in \mathbf{W}$;
2. *M satisfies* $(xy)^\omega (yx)^\omega (xy)^\omega = (xy)^\omega$ *and* $(ws)^\omega xwy (tw)^\omega = (ws)^\omega wxy (tw)^\omega = (ws)^\omega xyw (tw)^\omega$;
3. *If M is A^*/γ then $\gamma \supseteq \approx_{t,p}$ for some t, p;*
4. $M \in (\ell\mathbf{1} \Box \mathbf{A_{com}}) \vee \mathbf{Ab}$;

Proof. $1 \Leftrightarrow 2$ is immediate from our definition of $\mathbf{W}$ and $2 \Leftrightarrow 3$ is a straightforward exercise. Note that 3 is most useful to obtain the upper bounds on $D^{\|}(M)$ and $R(M)$ for $M \in \mathbf{W}$ while the identities of 2 are central to the lower bound arguments for $M \notin \mathbf{W}$.

It is quite obvious from the definition of $\approx_{t,p}$ that $\mathbf{W}$ is the join of $\mathbf{Ab}$ with an aperiodic variety parametrized by $\approx_{t,1}$. The link to $\ell\mathbf{1} \Box \mathbf{A_{com}}$ stems from the simple observation that x and xa (or ax) are T_t-equivalent if and only if $|x|_a \geq t$.

A tricky multiparty extension of the above communication complexity model was proposed by [9]: k parties now wish to compute $f(x_1, x_2, \ldots, x_k)$ with player P_i having access to all the inputs *except* x_i. There is a large overlap in the information held by each player as any input is available to all but one party. Here also, the players exchange information according to some fixed protocol, the last bits of which are $f(x_1, x_2, \ldots, x_k)$. For $k = 2$, this model is equivalent to the classical two-party model described above and more generally we denote by $D_k(f)$ the k-party communication complexity of f. For a monoid M, we denote $D_k(M)$ the k-party complexity of computing the product $m_1 m_2 \ldots m_n$ when m_j is unknown to the i^{th} party for $j \equiv i \pmod{k}$. It can easily be verified that $D_{k+1}(M) \leq D_k(M)$ and this difference can be dramatic as the next example shows:

Example 1. Consider the $\mathcal{R}$-trivial, idempotent monoid $R_1 = \{1, a, b\}$ with multiplication given by $xy = x$ for any $x \neq 1$. Evaluating the product $m_1 m_2 \ldots m_n$ boils down to deciding whether the first m_i which is not 1 (if any) is equal to a or b and one can use this observation to get: $D_2(R_1) = \Theta(\log n)$. On the other hand, if m_t is that first non-identity element in the sequence, three players can figure out its value as follows: each player in turn tells his partners whether the first non-identity m_i that he has access to (if any) is a or b. Two of the three players do have access to m_t and will thus send the correct value while the other player might err. Thus five bits of communication will suffice (an extra two bits is needed to verify that some m_i is not 1) to evaluate the product, regardless of input length.

As in the two-party case, the classes $\{M : D_k(M) = O(f)\}$ form varieties for any k and any increasing f and one can show:

Theorem 4 ([23, 30]). *There exists k such that $D_k(M) = O(1)$ if and only if M lies in the variety $\mathbf{DO} \cap \overline{\mathbf{G_{nil}}}$ of monoids in $\mathbf{DO}$ whose subgroups are nilpotent.*

For a group G, we have $D_k(G) = O(1)$ if G is nilpotent of class $(k-1)$ and $D_k(G) = \Theta(n)$ otherwise.

There is only little known, however, about the varieties $\mathbf{U_k} = \{M : D_k(M) = O(1)\}$ for specific k. Since the case of groups is completely resolved, we focus on the aperiodic case: from the previous theorem, $\mathbf{U_k} \cap \mathbf{A} \subseteq \mathbf{DA}$ and we have seen that the variety of aperiodic monoids with bounded 2-party communication complexity is $\mathbf{A_{Com}}$. These investigations have led us to propose a new parametrization of $\mathbf{DA}$ via congruences as follows: for any $s \geq 1$ and $t \geq 1$ we define $\sim_{s,t}$ on A^*. First, for any t, we let $x \sim_{1,t} y$ for all x, y. Then recursively, we define $x \sim_{s,t} y$ for $s \geq 2$ if and only if

1. $x \sim_{s-1,t} y$;
2. $\alpha_t(x) = \alpha_t(y)$;
3. For all $x = x_0 a x_1$ and $y = y_0 a y_1$ with $|x_0|_a = |y_0|_a \leq t$, we have $x_0 \sim_{s-1,t} y_0$ and $x_1 \sim_{s-1,t} y_1$;

4. For all $x = x_0 a x_1$ and $y = y_0 a y_1$ with $|x_1|_a = |y_1|_a \leq t$, we have $x_0 \sim_{s-1,t} y_0$ and $x_1 \sim_{s-1,t} y_1$.

The following basic facts can be obtained

Lemma 3. *1. A^*/γ lies in $\mathbf{A_{com}}$ if and only if $\gamma \supseteq \sim_{2,t}$ for some t;*
2. A^*/γ is in $\ell\mathbf{1} \Box \mathbf{A_{com}}$ if and only if $\gamma \supseteq \sim_{3,t}$ for some t;
3. A^*/γ lies in $\mathbf{DA}$ if and only if $\gamma \supseteq \sim_{s,t}$ for some s, t;
4. The k-party communication complexity of $A^*/\sim_{k,t}$ is $O(1)$;

The first statement is trivial, while the second follows by observing that $\sim_{3,t} = \approx_{t,1}$. We will not explicitly prove the other two but mention that (3) can be proved by adapting the arguments of [33]. Finally, the upper bound of (4) is based on an iterative use of the trick described in Example 1 and its optimality is unresolved although we conjecture:

Conjecture 1. An aperiodic monoid M has bounded k-party communication complexity if and only if $M = A^*/\gamma$ with $\gamma \supseteq \sim_{k,t}$ for some t.

This is easily shown to hold for $k = 2$ (see Theorem 3) and some very nice lower bounds of [22] can be exploited to show that any aperiodic monoid outside $\ell\mathbf{1} \Box \mathbf{A_{com}}$ has 3-party communication complexity at least $\Omega(\log\log\log n)$. The conjecture is still open for any $k \geq 4$ and we believe that any progress on it first necessitates a better description of the varieties

$$\mathbf{F_k} = \{M = A^*/\gamma \text{ with } \gamma \supseteq \sim_{k,t} \text{ for some } t\}.$$

For instance, we do not know how this parametrization of $\mathbf{DA}$ relates to the one given in Section 2 or whether a simple algebraic operation relates $\mathbf{F_{k+1}}$ and $\mathbf{F_k}$. Answering these questions and possibly obtaining identities that characterize $\mathbf{F_k}$ are crucial to identify specific monoids for which we will later need to prove communication complexity lower bounds (note that the latter is probably quite difficult however).

The multiparty communication model has very nice properties and its applications to other areas of theoretical computer science are numerous [16], yet our understanding of its power is quite limited. The above conjecture would constitute a very nice first step in changing this state of affairs and we believe that it provides equally challenging problems to semigroup theory and to computation theory. Furthermore, as we shall see in the next section, bounds on the communication complexity of monoids can help us understand their limitations as computational devices.

4 Boolean Circuits

Because of their simplicity, circuits provide an intuitively appealing model to analyze computations [27, 37]. For our purposes, we will formalize the model of n-input circuits over the alphabet A as follows: the circuit consists of a directed acyclic graph where nodes of fan-in 0 (*input gates*) are labeled with "$x_i = a$?" for some $1 \leq i \leq n$ and $a \in A$, nodes (or gates) of fan-in $k \geq 1$ are labeled with some symmetric Boolean function $f : \{0,1\}^k \to \{0,1\}$ and there is a unique node of fan-out 0 (*output gate*). Such a machine recognizes a subset of $L \subseteq A^n$ in a natural way. We recognize $L \subseteq A^*$ by using a sequence $(C_n)_{n\geq 0}$ such that C_n is an n-input circuit recognizing $L \cap A^n$. If the sequence is *uniform*, that is if a description of C_n can be effectively computed given n then any language recognized in this way is decidable. There are virtually no lower bound results about this model exploiting uniformity assumptions (with the notable exception of $[1, 2]$) and we will simply consider non-uniform circuits.

Circuits are usually parametrized by size (number of nodes), depth (longest path from input node to output node) and instruction set (Boolean functions allowed as labels on non-input nodes). For a family of circuits $(C_n)_{n\geq 0}$, size and depth are of course functions giving for each n the size/depth of C_n. It is standard practice to abuse terminology by dropping the cumbersome reference to the sequence $(C_n)_{n\geq 0}$ and simply saying that $L \subseteq A^*$ is recognized by a circuit of size $s(n)$ and depth $d(n)$. Among the most commonly studied circuit classes, we find:

- AC^0: bounded-depth, polynomial size circuits with AND/OR gates of arbitrary fan-in;
- $\mathrm{CC}^0[q]$: bounded-depth, polynomial size circuits with MOD_q gates[1] of arbitrary fan-in;
- CC^0: $\bigcup_q \mathrm{CC}^0[q]$;
- $\mathrm{ACC}^0[q]$: bounded-depth, polynomial size circuits with AND/OR and MOD_q gates of arbitrary fan-in;
- ACC^0: $\bigcup_q \mathrm{ACC}^0[q]$;
- NC^1: $O(\log n)$ depth, polynomial size circuits with AND/OR gates of fan-in 2.

We will from now on identify a class of circuits with the class of languages it recognizes. Separating the computing power of the above classes turns out to be extremely difficult and the current knowledge can be summarized by the following. For any integer q and any prime p:

$$AC^0 \subsetneq ACC^0[q] \subseteq NC^1 \qquad \text{and} \qquad ACC^0[p] \subsetneq NC^1.$$

The class NC^1 admits an interesting algebraic characterization: an n-input program of length s over M is a sequence $(i_1, f_1) \dots (i_s, f_s)$ with $1 \leq i_j \leq n$ and $f_j : A \to M$. This program transforms an input $x_1 \dots x_n \in A^n$ into the monoid element $f_1(x_{i_1}) \dots f_s(x_{i_s})$ in M. Fixing a set F of accepting elements we can thus recognize subsets of A^n in a new way. As in the circuit case, we will say that L is recognized by a program of length $s(n)$ over M if there exists a sequence $(\phi_n)_{n\geq 0}$, where ϕ_n has size $s(n)$ and recognizes $L \cap A^n$. The following result is well known in Boolean circuit theory:

[1] The function $\mathrm{MOD}_q : \{0,1\}^k \to \{0,1\}$ is 1 if and only if the sum of its inputs is a multiple of q.

Theorem 5 ([5]). *For a language $L \subseteq A^*$, the following are equivalent:*

1. *$L \in NC^1$;*
2. *L can be recognized by a polynomial length program over a finite monoid M;*
3. *L can be recognized by a polynomial length program over a non-Abelian simple group.*

This theorem is actually an easy consequence of an earlier result of [17] but was rediscovered independently. It is most interesting that other natural subclasses of NC^1 have very nice algebraic descriptions (see e.g. [18, 28]). In particular if we denote by $\mathcal{P}(\mathbf{V})$ the class of languages recognized by polynomial length programs over some monoid in $\mathbf{V}$, then it can be shown ([7]) that:

$\mathrm{AC}^0 = \mathcal{P}(\mathbf{A})$ (aperiodic monoids);
$\mathrm{CC}^0 = \mathcal{P}(\mathbf{G_{sol}})$ (solvable groups);
$\mathrm{ACC}^0 = \mathcal{P}(\overline{\mathbf{G_{sol}}})$ (solvable monoids: monoids whose subgroups are solvable);
$\mathrm{ACC}^0[p] = \mathcal{P}(\overline{\mathbf{G_p}})$ (p-monoids: monoids whose subgroups are p-groups);

A major open problem in this field is to separate ACC^0 from NC^1. This is equivalent to proving that the word problem of the group A_5 can not be recognized by programs of polynomial length over a solvable monoid. As of yet, the best known results only state that programs over a monoid in $\mathbf{G_p} * \mathbf{Ab}$ that compute multiplication in A_5 require exponential length [6] and these arguments can in fact be extended to $\mathbf{LG_p}$ Ⓜ $\mathbf{Com}$ [30]. The next challenge is to prove a similar lower bound for a group outside the varieties $\mathbf{G_p} * \mathbf{Ab}$ and the most reasonnable targets seem to be in $\mathbf{G_p} * \mathbf{G_{nil,2}}$ or more generally $\mathbf{G_p} * \mathbf{G_{nil}}$. This will most certainly require that the current combinatorial methods be combined with more and more sophisticated group theoretical tools (see for example the use of representation theory and Fourier analysis in [14]).

A number of interesting questions about AC^0 and the computing power of programs over aperiodic monoids are also left open. Let us denote by AC^0_i the class of AC^0 circuits of depth at most i.

Every element $m \in M$ defines a function $f_m : M \to M$ given by $f_m(x) = xm$ and we denote by $\overline{M}$ the monoid generated (under composition of functions) by the f_m's and the functions c_m given by $c_m(x) = m$. We will further use the following notation: $\mathbf{DD_1} = \mathbf{J_1}$ is the variety of semilattices, and $\mathbf{DD_{k+1}}$ is the variety generated by the syntactic monoids of languages of the form $L_0 a_1 L_1 \cdots a_s L_s$, where the a_i's are letters and each L_i has its syntactic monoid in $\mathbf{DD_k}$ (i.e. the so-called *dot-depth k monoids*).

Theorem 6. *L lies in $\mathcal{P}(\mathbf{DD_1})$ if and only if L is a Boolean combination of languages in AC^0_1.*

L lies in $\mathcal{P}(\mathbf{DD_k})$ if and only if L is a Boolean combination of languages in AC^0_k, if and only if L lies in $\mathcal{P}(\langle C_k \rangle)$, where $C_1 = U_1$, the 2-element semilattice, and $C_k = U_1 \circ \overline{C_{k-1}}$.

Because, $\mathbf{LI}$ Ⓜ $\mathbf{DD_k}$ is contained in $\mathbf{DD_{k+1}}$, the programs over that variety recognize only languages which are Boolean combinations of AC^0_{k+1} and we further conjecture:

Conjecture 2. $\mathcal{P}(\mathbf{LI} \circledM \mathbf{DD_k}) \subsetneq$ Boolean combinations of AC^0_{k+1}.

Note that a necessary condition for this conjecture to hold would be to establish that the monoid C_k defined above is not in $\mathbf{LI} \circledM \mathbf{DD_{k-1}}$.

Lemma 4. *Conjecture 2 holds for $k = 1$, i.e. $\mathcal{P}(\mathbf{LI} \circledM \mathbf{DD_1}) \subsetneq$ Boolean combinations of AC^0_1.*

Proof. It is known that $\mathbf{LI} \circledM \mathbf{DD_1} = \mathbf{DA}$ [25] and we will bound $\mathcal{P}(\mathbf{DA})$ by exploiting the communication complexity bounds presented in Section 3. Alternatively, we could establish the lemma using either the methods of [18] or a recent characterization of $\mathcal{P}(\mathbf{DA})$ in terms of decision trees of bounded rank [13].

Let $f : A^* \to \{0, 1\}$ be computed by a family $(\phi_n)_{n\geq 0}$ of programs over a monoid $M \in \mathbf{DA}$ with length bounded by the polynomial $p(n)$. Suppose now that Alice and Bob want to compute the output of the program $\phi_n(x)$ with $x \in A^n$. The output of each instruction of ϕ_n depends on a single letter of x and can therefore be computed by one of Alice or Bob, regardless of how the knowledge of x is partitioned between the two players. Their task is thus to evaluate in M a product $m_1 m_2 \dots m_{p(n)}$ where each m_i is known to one of them. By Theorem 3, this can be done with communication $O(\log(p(n)))$ and so we have $D(f) = O(\log n)$. On the other hand, the function Equality given by $\mathrm{Equality}(x, y) = 1$ if and only if $x = y$ is easily seen to lie in AC^0_2 but to have communication complexity $\Theta(n)$.

Conjecture 2 is open for any $k \geq 2$ and is quite closely related to open communication complexity conjectures about the cc-polynomial-hierarchy introduced by [4].

We finally mention an intriguing question about monoids whose mulitplication can be computed by circuits of linear size. Because Boolean circuits compute Boolean-valued functions, we define an instance of the multiplication problem for M as a pair $(m_1 m_2 \dots m_n, t)$ with $m_i, t \in M$ which is to be accepted if and only if $m_1 \cdot m_2 \cdot \ldots \cdot m_n = t$.

Lemma 5. *If multiplication in S can be computed by linear-size AC^0 (resp. CC^0, ACC^0) circuits, then for any $T \in \mathbf{J_1}$ (resp. $T \in \mathbf{Ab}$, $T \in (\mathbf{J_1} \vee \mathbf{Ab})$) the multiplication in $S \,\square\, T$ can also be computed by linear-size AC^0 (resp. CC^0, ACC^0) circuits.*

Proof. It suffices to show that for any alphabet A and any word x in A^* the language $\{y \in A^n : y \equiv_{S\square T} x\}$ can be recognized in linear size. Consider the case $T \in \mathbf{J_1}$ and suppose $|x| = n$. We need to determine $[x]_{\equiv_T}$ and $[h(x)]_{\equiv_S}$, where $h(x)$ is the word of length n over the alphabet $A^*/\equiv_T \times A \times A^*/\equiv_T$ given by $h(x) = X_1 \cdots X_n$, with $X_i = ([a_1 \cdots a_{i-1}]_{\equiv_T}, a_i, [a_{i+1} \cdots a_n]_{\equiv_T})$. As mentioned before, when $T \in \mathbf{J_1}$, $[x]_{\equiv_T}$ is determined by the set of letters that appear in x; for each letter a, we can use a single OR gate to check if a appears or not in x, and then a (fixed-size) boolean combination of such gates can determine

$[x]_{\equiv_T}$. To get the second condition, the hypothesis on S tells us that a linear-size circuit can determine $[h(x)]_{\equiv_S}$, given $X_1, \cdots, X_n$. It thus suffices to show that each X_i can be set up from $a_1, \cdots, a_n$ in constant size. This is clear that for each of the n prefixes of x and each of its n suffixes, the $\equiv_T$-class can be determined as above with fixed-size circuitry, and that consequently each X_i can be determined with fixed-size circuitry. This of course involves reusing the input gates "$x_i = a?''$ as often as necessary. The case $T \in \mathbf{Ab}$ is dealt with in exactly the same way, using MOD_q-gates instear of OR, where q is the exponent of T. The last case is a trivial combination of the first two.

Corollary 1. *If M lies in $\mathbf{DA}$ (resp. $\mathbf{G_{sol}}$, $\mathbf{DA} * \mathbf{G_{sol}}$) then multiplication in M can be computed by linear-size AC^0 (resp. CC^0 and ACC^0) circuits.*

Proof. For a variety $\mathbf{V}$, we define the weak block product closure of $\mathbf{V}$ (denoted *wbpc*$(\mathbf{V})$) as the union of all

$$\mathbf{B_k} = \underbrace{(\ldots(\mathbf{V} \,\Box\, \mathbf{V}) \,\Box\, \ldots) \,\Box\, \mathbf{V}}_{k \text{ times}}.$$

We have already mentioned in Section 2 that *wbpc*$(\mathbf{J_1}) = \mathbf{DA}$ and it can further be proved that *wbpc*$(\mathbf{Ab}) = \mathbf{G_{sol}}$ and *wbpc*$(\mathbf{J_1} \vee \mathbf{Ab}) = \mathbf{DA} * \mathbf{G_{sol}}$ [29] and this yields the corollary.

Converse statements are an important challenge. Currently, although there are functions in AC^0 known to require super-linear size circuits [10], these are very unnatural. It is very tempting to conjecture that there are regular languages in AC^0 that cannot be done in linear size, e.g. the language $A^*ac^*aA^*$ over the three-letter alphabet. We should stress also that the variety $\mathbf{DA} * \mathbf{G_{sol}}$ is not known to be decidable and Corollary 1 further motivates the importance of that question.

5 Conclusion

In this paper, we have used temporal logic, communication complexity and boolean circuits to argue in favour of greater interaction between semigroup theory and theory of computation. Investigations in temporal logic have brought to light the usefulness of parenthesizing iterated block products in the non-standard way and several important algorithmic questions arise from this work. Communication complexity has received tremendous attention in many areas of computer science and we have shown that key open problems seem to be naturally addressable with algebraic methods. Finally, boolean circuits of small depth also provide a long list of algebraic questions whose answers would have major impact on our understanding of computations.

There are of course other areas where the collaboration algebra-computation could be profitable: constraint satisfaction problems [8, 26], quantum automata [3], probabilistic models of computation, ... We hope that more and more members of our community will take up the challenge and explicitly include theory of computation as an important application domain of their algebraic expertise.

References

1. E. Allender. The permanent requires large uniform threshold circuits. *Chicago Journal of Theoretical Computer Science*, 1999.
2. E. Allender and V. Gore. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing*, 23(5):1026–1049, 1994.
3. A. Ambainis, M. Beaudry, M. Golovkins, A. Ķikusts, M. Mercer, and D. Thérien. Algebraic results on quantum automata. Submitted for publication, 2003.
4. L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proc. 27^{th} IEEE Symp. on Foundations of Comp. Sci. (FOCS'86)*, pages 337–347, 1986.
5. 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. Preliminary version appeared in STOC'86.
6. D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, 1990.
7. D. A. M. Barrington and D. Thérien. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 35(4):941–952, Oct. 1988.
8. A. Bulatov, P. Jeavons, and M. Volkov. Finite semigroups imposing tractable constraints. In G. Gomez, P. Silva, and J-E.Pin, editors, *Semigroups, Algorithms, Automata and Languages*. WSP, 2002.
9. A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *Proc. 15^{th} Symp. on Theory of Computing(STOC'83)*, pages 94–99, 1983.
10. S. Chaudhuri and J. Radhakrishnan. Deterministic restrictions in circuit complexity. In *Proc. 28th ACM Symp. on Theory of Computing (STOC'86)*, pages 30–36, 1996.
11. J. Cohen, D. Perrin, and J.-E. Pin. On the expressive power of temporal logic. *Journal of Computer and System Sciences*, 46(3):271–294, 1993.
12. A. Emerson. Temporal and model logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science. Vol. B: Formal Models and Semantics*, pages 995–1072. Elsevier Science Publishers B.V., 1990.
13. 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.
14. M. Goldmann, A. Russell, and D. Thérien. An ergodic theorem for read-once non-uniform deterministic finite automata. *Information Processing Letters*, 73(1–2):23–28, 2000.
15. J. A. W. Kamp. *Tense Logic and the Theory of Linear Order.* PhD thesis, University of California, Berkeley, 1968.
16. E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.
17. W. Maurer and J. Rhodes. A property of finite simple non-Abelian groups. *Proc. Amer. Math. Soc*, 16:552–554, 1965.
18. P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.
19. J.-E. Pin. *Varieties of formal languages.* North Oxford Academic Publishers Ltd, London, 1986.
20. J.-E. Pin. Syntactic semigroups. In G. R. et A. Salomaa, editor, *Handbook of language theory*, volume 1, chapter 10, pages 679–746. Springer Verlag, 1997.

21. J.-É. Pin, H. Straubing, and D. Thérien. Locally trivial categories and unambiguous concatenation. *J. Pure Applied Algebra*, 52:297–311, 1988.
22. P. Pudlák. An application of Hindman's theorem to a problem on communication complexity. To appear in *Combinatorics, Probability and Computing*, 2003.
23. 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.
24. J. Rhodes and B. Tilson. The kernel of monoid morphisms. *J. Pure and Applied Algebra*, 62:227–268, 1989.
25. M. P. Schützenberger. Sur le produit de concaténation non ambigu. *Semigroup Forum*, 13:47–75, 1976.
26. S. Seif and C. Szabó. Algebra complexity problems involving graph homomorphism, semigroups and the constraint satisfaction problem. *Journal of Complexity*, 19(2):153–160, 2003.
27. H. Straubing. *Finite Automata, Formal Logic and Circuit Complexity.* Boston: Birkhauser, 1994.
28. H. Straubing. When can one monoid simulate another? In *Algorithmic Problems in Groups and Semigroups*, pages 267–288. Birkhäuser, 2000.
29. H. Straubing and D. Thérien. Weakly iterated block products of finite monoids. In *Proc. of the 5th Latin American Theoretical Informatics Conference (LATIN '02)*, 2002.
30. P. Tesson. *Computational Complexity Questions Related to Finite Monoids and Semigroups*. PhD thesis, McGill University, 2003.
31. P. Tesson and D. Thérien. Complete classifications for the communication complexity of regular languages. In *Proc. Symp. on Theoretical Aspects of Comp. Sci. (STACS'03)*, 2003.
32. D. Thérien. Two-sided wreath product of categories. *J. Pure and Applied Algebra*, 74:307–315, 1991.
33. D. Thérien and T. Wilke. Over words, two variables are as powerful as one quantifier alternation. In *Proc. 30th ACM Symposium on the Theory of Computing*, pages 256–263, 1998.
34. D. Thérien and T. Wilke. Nesting until and since in linear temporal logic. In *Proc. Symp. on Theoretical Aspects of Comp. Sci. (STACS'02)*, pages 455–465, 2002. Extended version available from authors' web site.
35. D. Thérien and T. Wilke. Temporal logic and semidirect products: An effective characterization of the until hierarchy. *SIAM Journal on Computing*, 31, 2002.
36. B. Tilson. Categories as algebras. *J. Pure and Applied Algebra*, 48:83–198, 1987.
37. H. Vollmer. *Introduction to Circuit Complexity: A Uniform Approach*. Texts in theoretical computer science. Springer, 1999.