Multiparty Communication Complexity of Finite Monoids

Jean-François Raymond **Pascal Tesson**

Denis Thérien*

School of Computer Science,
McGill University
3480 rue University, Montréal (PQ)
H3A 2A7, Canada

Abstract

We study the relationship between the complexity of languages, in Yao's 2-party communication game and its extensions, and the algebraic properties of finite monoids that can recognize them.

For a finite monoid M, we define $C^{(k)}(M)$ to be the maximum number of bits of communication that players need to exchange, in the k-party game of Chandra, Furst and Lipton, to decide membership in any language recognized by M. We show that communication complexity classes induce pseudo-varieties of finite monoids in a natural way and characterize some of them.

Our results lead us to conjecture an extension of Szegedy's algebraic characterization of languages having bounded 2-party communication complexity.

We also mention some applications of communication complexity lower bounds to circuit complexity.

1 Introduction

1.1 Communication complexity

Complexity theory traditionally measures the complexity of functions in terms of time and space ressources needed for computation. In [26], however, Yao introduced the notion of communication complexity in which time and space requirements are irrelevant.

In this model, two players, each having access to half of the input, communicate bits to each other in order to compute the value of a function on some given

*This research was supported by FCAR and NSERC grants

input. The complexity of the function is then measured as the minimal amount of communication that the two parties need to exchange on inputs of length n.

Different extensions and variants of communication complexity models have been proposed and studied since then and have helped to answer various open questions. For instance, the Karchmer-Widgerson game [15] established links between monotone circuit-depth and communication complexity and later allowed Raz and McKenzie [21] to show that the monotone-NC hierarchy is infinite and strictly contained in monotone-P.

Chandra, Furst and Lipton [7] introduced an extension of Yao's model to k players. Here, each player has some portion of the input written on his forehead and available to all but himself. This is a stronger model than the original 2-player game and only few lower bounds are known for it. These results however have many applications [1, 7], particularly in the study of circuit complexity [10, 12, 13, 14].

1.2 Finite automata, finite monoids and circuit complexity

The relationship between finite monoids and regular languages has been well uncovered [9, 19], and has been exploited since the 50's to classify these languages and analyze their combinatorial structure. Using the formalism of programs over monoids, introduced by Barrington, these ideas were recently extended to obtain algebraic characterizations of small classes of circuits [3, 4, 17].

A surprising link between algebra and communication complexity was discovered by Szegedy in [23]. He proved that the membership question for a language can be decided using constant communication in the 2-party model of Yao if and only if that language can be recognized by a program over a commutative monoid.

It thus seems worthwhile to explore the relationship between the communication complexity of a language and the algebraic properties of finite monoids that can recognize them, particularly in the light of the many applications of communication complexity to circuit complexity.

We prove first that communication complexity classes give rise naturally to varieties of finite monoids and then try to characterize the varieties thus induced. We prove that evaluating a product in a group requires constant communication in the k-party game iff the groups is nilpotent of class $(k-1)$ and linear communication otherwise. We also show that in the 2-party game, evaluating a product in an aperiodic monoid M requires constant communication if M is commutative, $\Theta(\log n)$ communication if M belongs to the pseudo-variety $\mathbf{DA}$ but is not commutative and $\Theta(n)$ communication otherwise. When M is in $\mathbf{DA}$, we further show that there is a k such that M has constant k-party communication complexity.

In section 2, we provide some background on communication complexity and survey some of its applications to circuit complexity. We then present some basic notions of algebraic automata theory in section 3. In section 4, we prove that communication complexity classes induce monoid varieties and characterize some of them in section 5.

2 Communication complexity models and their applications

We give here the basic definitions of 2-party and multiparty models of communication complexity. We survey some of the proof techniques, fundamental results and applications to small complexity classes.

2.1 The 2-party model

This communication complexity model was first introduced by Yao in [26], motivated by distributed computing issues. It has however found various other applications since then.

Its game characterization is the following: two players P_1 and P_2 want to collaborate to compute a function $f : \{0,1\}^X \to \{0,1\}$. The set of input variables $X = X_1 \dot{\cup} X_2$ is partitionned in two and player P_i only has access to variables in X_i. The computing power of each player is assumed to be unbounded. They are allowed to communicate, exchanging bits in turn according to some fixed protocol until both of them know the value of f on the given input. Similarly, a protocol for a language L stops when both players have determined whether the input belonged to L or not.

We define the cost of a protocol as the maximum number of bits exchanged by the two parties on any given input. The communication complexity of the function f with respect to some partition (X_1, X_2) is the cost of the cheapest valid protocol for f and is denoted $C^{(2)}_{(X_1,X_2)}(f)$.

The symmetric communication complexity of f, denoted $C^{(2)}(f)$ is the communication complexity of f with respect to the worst possible partition:

$$C^{(2)}(f) = \max_{X = X_1 \dot{\cup} X_2} C^{(2)}_{(X_1,X_2)}(f)$$

Symmetric communication complexity will be our main focus in this paper, and we will simply refer to it as the communication complexity.

Note that, in general, we are interested in studying functions from $\{0,1\}^*$ to $\{0,1\}$. Consequently, we will define $C^{(2)}(f)$ as a function from $\mathbf{N}$ to $\mathbf{N}$, such that $C^{(2)}(f)(n)$ is the communication complexity of f restricted to inputs of length n.

One convenient way of analyzing the communication complexity of a function f is through its communication matrix. Given a partition (X_1, X_2) of the input variables, we can build a $2^{|X_1|} \times 2^{|X_2|}$ matrix in which rows are labeled by possible settings of variables in X_1 and columns by settings of variables of X_2. Each entry in the matrix naturally corresponds to some input x and we assign it the value $f(x)$. Now, suppose that the communication string, that is the sequence of bits exchanged by the two parties, is the same on two distinct inputs (x_1, x_2) and (x'_1, x'_2). Then that same communication string will also be induced by inputs (x_1, x'_2) and (x'_1, x_2).

Indeed, the communication sent by each player at any given time can only depend on its input and on the information broadcasted thus far. Using the very same argument, the value of f on these four inputs must be identical. Hence, every communication string of a protocol corresponds to some homogeneous rectangle (i.e. a submatrix with all entries equal) of the communication matrix, and every protocol of cost c must partition the matrix into at most 2^c disjoint homogeneous rectangles.

This fact is used crucially in almost all lower bound arguments for the 2-party model, such as the fooling-set technique. A fooling set for the function $f(x_1, x_2)$ is a set F of input pairs (x_1, x_2) such that

- for all $(x_1, x_2) \in F$, we have $f(x_1, x_2) = v$
- for any two distinct pairs (x_1, x_2), $(x'_1, x'_2) \in F$, either $f(x'_1, x_2) \neq v$ or $f(x_1, x'_2) \neq v$.

Lemma 2.1 *If f has a size $|F|$ fooling set, then* $C^{(2)}(f) \geq \log |F|$

Proof: By definition, two elements of a fooling set can not be elements of the same homogeneous submatrix. Therefore, we need at least $|F|$ such submatrices, hence at least $\log |F|$ bits of communication. $\square$

Classical applications of this lemma yield linear lower bounds on the 2-party communication complexity of the equality and disjointness functions, where $EQ(x, y) = 1$ iff $x = y$ and $DISJ(x, y) = 1$ iff $\sum x_i y_i = 0$.

2.2 The multiparty model

An interesting extension of Yao's model to more than two players was proposed by Chandra, Furst and Lipton in [7]. The input variables are partitionned in k sets $X = X_1 \dot\cup X_2 \dot\cup \ldots \dot\cup X_k$. Each player P_i now has access to every variable *except* the ones in X_i. Every input bit is thus known by $(k-1)$ of the k players.

The k parties communicate by writing bits on a blackboard seen by all. We define the cost of a protocol as in the 2-party model, and denote the (symmetric) k-party complexity of a function f as $C^{(k)}(f)$.

When only two players are involved, this model is equivalent to the original 2-party game. The power of this model grows as the number of players increases since it is easy for $(k+1)$ players to simulate a k-party protocol. Lower bounds for the multiparty communication complexity of specific functions are difficult to prove, but they have rich and diverse applications, particularly in the study of small circuit classes or branching programs.

These applications share a common structure of argument: following [10], we say that a circuit is k-evaluated using c bits of communication if k players sharing the input variables as above can determine the output of the circuit by exchanging at most c bits of information.

The communication complexity of the circuit is clearly an upper bound for the communication complexity of the function it computes but it can often be bounded in terms of the size or depth of the circuit. Lower bounds for the communication complexity of a specific function can thus provide lower bounds on various parameters of circuits that can compute it. Similar arguments can also be applied to the study of branching programs.

Using Ramsey-theoretic arguments, Chandra, Furst and Lipton first established in [7] an $\omega(1)$ lower bound (i.e. non-constant, see e.g. [8] for notation) on the k-party communication complexity of the $EXACTLY_N^k$ function, defined as

$$EXACTLY_N^k(x_1, x_2, \ldots, x_k) = 1 \text{ iff } \sum x_i = N$$

where N is some n-bit integer known to all. They used that result to prove a super-linear lower bound on the length of bounded-width branching programs computing $\mathrm{EXACTLY}_N^k$.

In [1], Babai, Nisan and Szegedy showed an $\Omega(n/4^k)$ lower bound on the k-party communication complexity of the generalized inner product GIP_k, defined over an $n \times k$ matrix A by $GIP_k(A) := 1$ iff the number of all 1 rows of A is congruent to 0 (mod 2). They also proved a similar lower bound on the communication complexity of computing the Legendre symbol $(\frac{x}{p})$, where $x = \sum x_i$ is the sum of the k n-bit inputs and p is a given n-bit prime. These results yield time-space tradeoffs for Turing machines and length-width tradeoffs for restricted types of branching programs.

Grolmusz [10] later generalized this lower bound to the function $GIP_{k,p}$ where the number of all 1 rows is counted mod p, for some fixed prime p. This allowed him to show that polynomial-size depth-2 circuits constructed with MOD_p gates, where p is prime, have less computational power than polynomial-size depth-2 circuits constructed with MOD_m gates, where m has at least two distinct prime divisors. More precisely, he proved that these MOD_p circuits can be k-evaluated using $O(k \log p)$ bits of communication when $k \geq p$. When p and k are constant, this proves that such circuits can only compute functions with constant k-party communication complexity, whereas depth-2 MOD_m circuits can compute, for any k, a function closely related to GIP_k and having linear k-party communication complexity.

Other similar applications, in particular to circuits involving treshold gates, have stemmed from probabilistic and other variants of this model [1, 11, 13, 14, 18].

There is also hope that communication complexity could help answer open questions about ACC^0, the class of functions that can be computed by polynomial-size circuits built with AND, OR, NOT and MOD_m gates. It is known that such circuits can be simulated by depth-2 circuits consisting of a symmetric gate at the output level (one whose output depends only on the sum of its inputs) and a polylogarithmic number of AND gates of polylogarithmic fan-in [27, 6]. As was observed by Håstad and Goldmann [14], this implies that a polylogarithmic number of players can evaluate an ACC^0 circuits by exchanging a polylogarithmic number of bits. Thus, proving a super-polylogarithmic lower bound on the communication complexity of some specific function for a polylogarithmic number of players would separate ACC^0 from whatever complexity class that function belongs to. However, the best multiparty communication complexity lower bound known thus far, is the $\Omega(n/4^k)$ for the GIP_k function, which yields only a trivial bound when $k \geq \log_4 n$.

3 Monoids as machines

We refer the reader to [9] or [19] for a more detailed introduction to the material sketched in this section. All monoids considered in this paper will be finite with the exception of the free monoid A^*.

We say that a monoid M divides N and write $M \prec N$ if M is the morphic image of a submonoid of N. A class $\mathbf{V}$ of finite monoids is a pseudo-variety (or variety for short) if it is closed under finite direct product and division.

For any monoid M, we define the evaluation morphism $eval_M : M^* \rightarrow M$ as $eval_M(m_1, m_2, \ldots m_n) = m_1 \cdot m_2 \cdot \ldots \cdot m_n$.

We say that a monoid is aperiodic if no subset of it forms a non-trivial group.

3.1 Regular languages

We say that a language $L \subseteq A^*$ is recognized (via morphism) by a finite monoid M if there exists a morphism $\phi : A^* \rightarrow M$ and a subset $F \subseteq M$ such that $L = \phi^{-1}(F)$. It is well known that a language is recognized by some finite monoid M iff it is regular.

Given a language L, we can define for any $x \in A^*$ the set $Y_x = \{(u, v) : uxv \in L\}$. The syntactic congruence of L is defined by $x \sim_L y$ iff $Y_x = Y_y$. When L is regular, $\sim_L$ has finite index and the monoid $A^*/\sim_L$ is called the syntactic monoid of L and is denoted $M(L)$.

Fact 3.1 ([9]) *$M(L)$ recognizes L and for any N recognizing L, $M(L) \prec N$.*

We say that a class of languages $\mathbf{L}$ is a language variety if it is closed under boolean operations, left and right quotients (i.e. if L is in $\mathbf{L}$ then for any strings $u, v \in A^*$ we have $u^{-1}Lv^{-1} = \{w : uwv \in L\} \in \mathbf{L}$) and inverse morphisms. (i.e. If $L \subseteq A^*$ is in $\mathbf{L}$, and $\phi : B^* \rightarrow A^*$ is a monoid morphism, $\phi^{-1}(L)$ is in $\mathbf{L}$).

It has been shown [9] that for every language variety $\mathbf{L}$ there is a monoid variety $\mathbf{V}$ such that a language L is in $\mathbf{L}$ iff $\mathrm{M}(L)$ is in $\mathbf{V}$ and that varieties are thus the natural unit for classifying monoids in terms of their computing power.

3.2 Programs over monoids

An n-input program of length over a monoid M is a sequence of instructions

$$(i_1, f_1)(i_2, f_2) \dots (i_l, f_l)$$

where $i_j \in \{1, 2, \dots, n\}$ and $f_j : A \to M$, together with an accepting subset $F \subseteq M$. A word $a_1 a_2 \dots a_n$ is accepted by the program if the product $f_1(a_{i_1}) \cdot f_2(a_{i_2}) \cdot \dots \cdot f_l(a_{i_l})$ evaluates to some element of F.

A language $L \subseteq A^*$ is said to be recognized by a length $l(n)$ program over the monoid M if there exists a sequence of programs $(\phi_1, \phi_2, \dots)$, where ϕ_n is an n-input program of length at most $l(n)$ over M which accepts $L \cap A^n$.

We say that M P-divides N and write $M \prec_{\mathbf{P}} N$ if any language recognized via morphism over M can be recognized by a polynomial length program over N. Note that if $M \prec N$ then $M \prec_{\mathbf{P}} N$ since a morphism is simply a very restricted type of program. A class $\mathbf{V}$ of finite monoids is a P-variety if it is closed under finite direct product and P-division.

Lemma 3.2 *Commutative monoids form a P-variety.*

Proof: It is clearly sufficient to show that this class is closed under P-division. Suppose that $M \prec_{\mathbf{P}} N$, and that N is commutative, but M is not. There exist $a, b \in M$ such that $ab \neq ba$. Let $L \subseteq M^*$ be the regular language defined as $L = \{m_1 m_2 \dots m_n : eval_M(m_1, m_2, \dots, m_n) = ab\}$. This language is clearly recognized via morphism over M. We will show however that it can not be recognized by a program over N.

We can always assume that a program over a commutative monoid has the special form:

$$(1, f_1), (2, f_2), \dots (n, f_n)$$

Because of commutativity, instructions of the program can indeed be permuted as we want. Moreover, adjacent instructions querying the same bit can be merged into a single instruction.

Since M and N are finite, there are only finitely many choices for the f_i's. So for some n large enough, the n-input program for L contains the same function $f_i = f_j$, for two different instructions, $i \neq j$. Consider the two following length n words of M^*: $11 \dots 1a1 \dots 1b1 \dots 1$ and $11 \dots 1b1 \dots 1a1 \dots 1$, with a and b in positions i and j. Only the first one is equal to ab in M, but since N is commutative, the program outputs the same result on both inputs, a contradiction. $\square$

Programs over monoids have provided algebraic characterizations of small circuit classes. In particular, Barrington proved that a language can be recognized by a sequence of NC^1 circuits (boolean circuits of logarithmic depth) iff it can be recognized by a polynomial length program over any non-solvable group [2]. Further algebraic characterizations for natural subclasses of NC^1 were also presented in [4, 22].

4 Monoids in communication complexity

4.1 Communication complexity of a finite monoid

We define in this section the communication complexity of a monoid and show that communication complexity classes induce monoid varieties in a natural way.

Definition: The k-party communication complexity of a monoid M, denoted $C^{(k)}(M)$ is the maximum complexity in the k-party game of any language recognized by M via morphism.

The results of this section do not depend on the number of players involved. To empasize this and to simplify notation we will fix a k and simply write c_L instead of $C^{(k)}(L)$. One should keep in mind that c_L is a function from $\mathbf{N}$ to $\mathbf{N}$. When this needs to be stressed, we will use the notation $c_L(n)$.

Theorem 4.1 *Let $L \subseteq A^*$ be a regular language and let $M = M(L)$ be its syntactic monoid. Let c_L and c_{eval_M} be the k-party communication complexity of L and $eval_M$ respectively. Then $c_L = O(c_{eval_M})$.*

Moreover, if L has an identity letter then $c_{eval_M} = O(c_L)$ and so $c_L = \Theta(c_{eval_M})$.

Proof: First, there is a morphism $\phi : A^* \to M$ with $L = \phi^{-1}(F)$ for some $F \in M$. Given an input w in A^*, the players can thus privately apply ϕ to each input letter that they see. They obtain a word in M^* and can evaluate that product using c_{eval_M} bits of communication. Finally, they can privately check whether the result belongs to F or not.

For the second part of the lemma, we know that each monoid element is an equivalence class of the syntactic congruence of L. Each such class corresponds to some unique Y_x, and two different Y's must disagree on some pair (u, v). Since M is finite, there are only $|M|$ equivalence classes of $\sim_L$, so we can choose minimal length pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_t, v_t)$, $t \leq |M|^2$, such that $x \sim_L y$ iff Y_x and Y_y contain exactly the same pairs (u_i, v_i).

Using this fact, we can now build a $O(c_L)$ protocol for $eval_M$. Because L has an identity letter, the players can choose, for every $m \in M$, a word $w \in A^*$ of length q such that $\phi(w) = m$. Given a length n input in M^*, say $m_1, \ldots, m_n$, the players privately translate it to a word x of length qn in A^*. Note that

$$eval(m_1, \ldots, m_n) = m \equiv [x]_{\sim_L}$$

Now for each of the t pairs (u_i, v_i) the players can use the c_L-bit protocol to determine if (u_i, v_i) is in Y_x. Finally, each player knows Y_x and there is a unique $m \in M$ corresponding to it. $\square$

The identity letter hypothesis in this theorem can be can be weakened somewhat. We need only require that there is a q such that for every $m \in M$, $\phi^{-1}(m)$ contains a word of length q. However, the theorem does not hold if we completely remove this hypothesis.

We have just shown that $C^{(k)}(M) = \Theta(C^{(k)}(eval_M))$. This will be very convenient to prove bounds on $C^{(k)}(M)$ as $eval_M$, and any language L with $M(L) = M$ having an identity letter, provide explicit representations for it.

Theorem 4.2 *The class of finite monoids $\mathcal{M} = \{M : C^{(k)}(M) = O(f(n))\}$ forms a variety for any function f.*

Proof: It is clear that given protocols of respective costs c_M and c_N to evaluate products in M and N, we can build a protocol of cost $c_M + c_N$ to evaluate a product in $M \times N$. So $\mathcal{M}$ is closed under finite direct product.

Suppose now $N \in \mathcal{M}$ and $M \prec N$. There is a surjective morphism ϕ from a submonoid $T \subseteq N$ onto M. The players want to evaluate a product of n elements $m_1 m_2 \ldots m_n$ of M. Since ϕ is surjective, the players can privately translate this input to a word in T^* using a previously agreed upon scheme. (i.e. for every $m \in M$ they have chosen some $t \in T$ such that $\phi(t) = m$)

Now, using the c_N-bit protocol, they can evaluate in T the product

$$t_1 t_2 \ldots t_n = t'$$

and privately compute $\phi(t') = m_1 m_2 \ldots m_n$. So we get $C^{(k)}(M) = O(f(n))$. $\Box$

We would now like to understand what are the finite monoid varieties thus induced by communication complexity classes. The following provides a useful tool.

Lemma 4.3 *Let $L \subseteq A^*$ be such that L can be recognized by a $O(l(n))$-length program over a finite monoid M. Let c_L and c_M be the communication complexities of L and M respectively, then $c_L(n) = O(c_M(l(n)))$. In particular, if $l(n) = n$ then $c_L = O(c_M)$.*

Proof: Given a length n input in A^*, the players can use that program and then evaluate a string of length $O(l(n))$ in M^* to check if it belongs to the accepting subset $F \subseteq M$. Using the c_M-bit protocol for M, they can do this using $c_M(O(l(n)) = O(c_M(l(n)))$ bits of communication. $\Box$

This lemma can be seen, in fact, as a reduction argument. The construction of the M^* input requires no communication, so it is "easy" in terms of communication requirements and an input w is in L iff the input obtained through the reduction belongs to some well-defined language over M^*. As a corollary, we also get a result used implicitly in many applications of communication complexity lower bounds to boolean circuit complexity. Following the notation of [17] we denote by $P(\mathbf{V})$ the class of languages that can be recognized by a polynomial length program over a monoid in the variety $\mathbf{V}$. The following corollary shows that if we can separate the communication complexities of monoids of two varieties by a large enough factor, then we can separate the corresponding language classes.

Corollary 4.4 *Let $\mathbf{V}$ and $\mathbf{W}$ be monoid varieties such that:*

- *For all $M \in \mathbf{V}$, we have $C(M) = O(f(n))$.*
- *There is $N \in \mathbf{W}$, such that $C(N) = \omega(f(n^d))$ for any $d > 0$.*

Then $P(\mathbf{V}) \neq P(\mathbf{W})$.

Proof: Suppose $P(\mathbf{V}) = P(\mathbf{W})$. There exists a polynomial length program over some $M \in \mathbf{V}$ recognizing the word problem of N. Lemma 4.3 gives a contradiction. $\square$

4.2 Szegedy's Theorem

Szegedy found in [23] a surprising algebraic characterization of the class of languages of constant communication complexity in the 2-party game:

Theorem 4.5 (Szegedy) *A language L has bounded 2-party communication complexity iff L can be recognized by an M-program over a commutative monoid M.*

Since commutative monoids form a P-variety, we get:

Corollary 4.6 *A finite monoid M has constant 2-party communication complexity iff M is commutative.*

We now want to describe the monoid varieties induced by other 2-party and k-party communication complexity classes.

5 Bounds on the communication complexity of specific monoids

We present here some bounds on the communication complexity of specific classes of monoids. We study separately the case of groups and aperiodic monoids.

5.1 Groups

For any group G, we form the lower central series $G_0 = G$, $G_1 = [G, G]$, and $G_i = [G_{i-1}, G]$, where $[H, G]$ is defined as the subgroup generated by all commutators $[h, g] = h^{-1}g^{-1}hg$, where $h \in H$, $g \in G$. We say that a group G is nilpotent of class k if $G_k = \{1\}$. Equivalently, G is nilpotent of class k iff all its commutators of weight $(k+1)$ are equal to the identity, that is, for any $g_1, g_2, \dots, g_{k+1} \in G$,

$$[[\dots[g_1, g_2], g_3] \dots g_{k+1}] = 1$$

We show in this section that a group G has constant communication complexity in the k-party game iff G is nilpotent of class $(k-1)$ and that $C^{(k)}(G) = \Omega(n)$ otherwise. To prove the upper bound, we use the combinatorial description of regular languages recognized by nilpotent groups of class k proved in [24]. We say that $u = u_1u_2\dots u_m$ is a subword of x if x can be factorized as $x = w_0u_1w_1\dots u_mw_m$, with $w_i \in A^*$. For any $u \in A^*$, $q \geq 1$, we define $|x|_u$ as the number of occurrences of u as a subword of x and $[u, i, q] = \{x : |x|_u \equiv i \pmod q\}$.

Fact 5.1 ([24]) *A regular language L is recognized by a class m nilpotent group iff it is in the boolean algebra generated by the $\{[u, i, q]\}$ where $q \geq 1$ and $|u| \leq m$.*

To determine membership in such a language, it is thus sufficient to count mod some integer the number of occurrences of specific subwords of length at most m in the input. In the $(m+1)$-party game, every subword of length m is seen entirely by at least one player. There is thus a protocol of bounded cost for $(m+1)$ players to count mod q, for any q , the number of occurrences of a subword u of length m: the players agree beforehand on a partition of the $\binom{n}{m}$ m-tuples of input bits such each player is assigned only some m-tuples which he sees entirely. They can each subsequently broadcast the number mod q of occurrences of u in their respective m-tuples. Hence if G is nilpotent of class m, $C^{(m+1)}(G) = O(1)$.

Moreover, this condition is necessary for groups.

Lemma 5.2 *Let G be a group which is not nilpotent of class $(k-1)$. There is a k-input program over G which outputs an element of prime order p when all input bits are on, and outputs the identity e otherwise.*

Proof: Recall that a group is nilpotent of class at most $(k-1)$ if all of its commutators of weight k are the identity.

So G has a commutator $[[\dots[a_1, a_2], a_3], \dots, a_k]$ of order $q > 1$.

Let $C(i)$ be the single instruction outputting a_i if the i^{th} bit is on and e otherwise. We define inductively
$B(1) = C(1)$
$B(i+1) = B(i)C(i+1)B(i)^{-1}C(i+1)^{-1}$

Since we are working here over a group, we can define the inverse of an instruction $(i, f)^{-1}$ as the unique instruction (i, g) such that $f(x_i) = g(x_i)^{-1}$. Similarly, we can define the inverse of a sequence of instructions $((i_1, f_1)(i_2, f_2) \dots (i_l, f_l))^{-1}$ as $(i_l, f_l)^{-1} \dots (i_1, f_1)^{-1}$. Obviously, the concatenation of some sequence of instructions and its inverse will always output the identity, regardless of the input.

Consider the output of $B(k)$. When some bit in the input is off, the product cancels out and the program outputs e. When all input bits are on, we are left with the commutator $[[\dots[a_1, a_2], a_3], \dots, a_k]$. Now let $q = pm$, with p prime, the program consisting of the concatenation of m copies of $B(k)$ satisfies the requirements of the lemma. □

Theorem 5.3 *Let G be a group which is not nilpotent of class $(k-1)$. We have $C^{(k)}(G) = \Omega(n)$.*

Proof: We will show that for some prime p, $GIP_{k,p}$ can be recognized by a linear length program over G.

We want to count mod p the results of n k-wise AND's. Since we assume that G is not nilpotent of class $(k-1)$, we can use the program defined in the previous lemma. We concatenate n copies of that k-input program. Each separate copy outputs an element of order p in G iff all input bits are on and outputs the identity otherwise, so their concatenation will output the identity iff the number of all-1 rows of the $GIP_{k,p}$ input is congruent to 0 mod p.

This program has linear size, so by lemma 4.3 we get $C^{(k)}(G) = \Omega(n)$. □

We have thus proved:

Theorem 5.4 *Let G be a finite group. G has constant k-party communication complexity if and only if G is nilpotent of class $< k$. Otherwise $C^{(k)}(G) = \Omega(n)$.*

5.2 Aperiodic monoids

For aperiodic monoids, we consider separately the 2-party and multiparty cases.

2-party case

Using corollary 4.6, we have that an aperiodic monoid has bounded 2-party communication complexity iff it is commutative. We show here that any non-commutative aperiodic monoid has communication complexity either $\Theta(\log n)$ or $\Theta(n)$, in the 2-player setting, depending on its membership to the variety **DA** .

A monoid M belongs to the variety **DA** if for all $m, n \in M$, if $m = m^2$ is idempotent and $MmM = MnM$ then n is also idempotent. It can be shown from this definition that monoids in **DA** are aperiodic.

There is also a convenient combinatorial description of regular languages that are recognized by such monoids [20]. We say that $L = A_0^* a_1 A_1^* a_2 \dots a_k A_k^*$, where $a_i \in A$ and $A_i \subseteq A$, is an unambiguous concatenation if for every $w \in L$, the factorization $w = w_0 a_1 w_1 a_2 \dots a_k w_k$, where $w_i \in A_i^*$, is unique. A regular language is recognized by a monoid in **DA** if and only if it is the disjoint union of unambiguous concatenations $A_0^*a_1A_1^*a_2\ldots a_kA_k^*$.

We can now prove a logarithmic upper bound on the communication complexity of monoids in **DA** by exhibiting a logarithmic cost protocol for any unambiguous concatenation $A_0^*a_1A_1^*a_2\ldots a_kA_k^*$. We will refer to the a_i's as the bookmarks.

Lemma 5.5 *Let $L = A_0^*a_1A_1^*a_2\ldots a_kA_k^*$ be an unambiguous concatenation.* $C^{(2)}(L) = O(\log n)$

Proof: We will show by induction on k, the number of bookmarks, that a $O(\log n)$ protocol exists for any such language.

For $k = 1$, notice that all non-trivial unambiguous concatenations of the form $A_0^*a_1A_1^*$ can be reduced to c^*aA^* or its reverse A^*ac^*, where $A = \{a, b, c\}$. The $O(\log n)$ bound is simple in this case: each player gives the position of the first a it sees. Now, that they know where the bookmark a must be, they check if every letter preceding it is a c.

For the induction step, notice that if $A_0^*a_1A_1^*a_2\ldots a_kA_k^*$ is an unambiguous concatenation, then so is $A_{i-1}^*a_iA_i^*a_{i+1}\ldots a_jA_j^*$, for any $1 \leq i < j \leq k$. We will show that two players can always, using $O(\log n)$ bits of communication, reduce the problem for k bookmarks to two problems of less than k bookmarks.

Since $A_0^*a_1A_1^*a_2\ldots a_kA_k^*$ is an unambiguous concatenation, there must be some bookmark letter a_p which is not in both A_0 and A_k. If this was not the case, then the word $a_1a_2\ldots a_ka_1a_2\ldots a_k$ could be factorized either as $a_1a_2\ldots a_k(a_1a_2\ldots a_k)$ or as $(a_1a_2\ldots a_k)a_1a_2\ldots a_k$.

So suppose W.L.O.G. that $a_p \notin A_k$. Using $O(\log n)$ bits of communication, the players can find the position of a_p's last occurrence in the input. First assume, for simplicity, that $a_p \in A_l$, for some $0 < l < k$, but that $a_p \notin A_j$ for all $j \neq l$. Then the input w can be factorized as $w = w_0a_1w_1\ldots a_kw_k$ only if the last occurrence of a_p in w is either a bookmark or is in w_l.

In the first case, that last occurrence of a_p splits w into a word from the language $A_0^*a_1A_1^*a_2\ldots a_{i-2}A_{i-1}^*$ and a word from $A_i^*a_{i+1}\ldots a_kA_k^*$. By induction hypothesis, membership questions for these languages can be solved in $O(\log n)$ communication.

Similarly, if the last occurrence of a_p is in w_l, it must split w into a first segment in $A_0^*a_1\ldots a_lA_l^*$ and a second one in $A_l^*a_{l+1}\ldots a_kA_k^*$.

If a_p occurs in more than one of the A_j's, the players must simply check each case, i.e. first assume that the last occurrence of a_p is in w_{k-1} and see if they can factorize these two segments of w appropriately, if not assume it is in w_{k-2}, and so on. Also, if $l = 0$, then the last occurrence of a_p must necessarily be a bookmark. Finally, if a_p does not occur in the input, then the players know immediately that w is not in L. $\Box$

To establish the matching lower bound for non-commutative monoids of **DA** , we first need to state a technical lemma about non-commutative aperiodics.

Lemma 5.6 *Let M be a non-commutative finite aperiodic monoid. If $M \in \mathbf{DA}$ then M is divided by one of the following:*

- *the syntactic monoid of $c^*ac^*bc^*$*
- *the syntactic monoid of c^*aA^**
- *the syntactic monoid of A^*ac^**

If M does not belong to $\mathbf{DA}$, then it is divided by one of the following:

- *the syntactic monoid of $(c^*ac^*bc^*)^*$ denoted BA_2*
- *the syntactic monoid of $((b+c)^*a(b+c)^*b(b+c)^*)^*$ denoted U*

Presenting a proof of this lemma would require the introduction of additional results on finite monoids and we omit it. A proof of the first part of this lemma can, however, be found in [16] and Lemma 3.3 of [5] essentially yields the second half.

Also, notice that the regular languages that we used to define BA_2 and U are similar. Ignoring the c's, we require alternation of a's and b's in the first case whereas for U, we only ask that there be a b between any two a's. Of course, a's and b's alternate iff there is a b between any two a's and an a between any two b's. This observation can be used to show that BA_2 divides $U \times U$.

We can now prove the logarithmic lower bound for monoids in $\mathbf{DA}$. It is sufficient to prove that the 2-party communication complexity of the regular languages $c^*ac^*bc^*$ and c^*aA^* is $\Omega(\log n)$.

Lemma 5.7 *1. $C^{(2)}(c^*ac^*bc^*) = \Omega(\log(n))$*

2. $C^{(2)}(c^*aA^*) = \Omega(\log(n))$, when $A = \{a, b, c\}$

Proof: We can provide a linear size fooling set for each language. Consider the partition where player 1 sees all the odd-indexed letters and player 2 sees all the even-indexed letters of the input.

- For $c^*ac^*bc^*$, let $F = \{c^{2i}bac^{n-2i-2} : i = 0, 1 \ldots n/2 - 1\}$. None of these $\frac{n}{2}$ words are in the language, but any two distinct words of F will define a rectangle containing a word of the form $c^j ac^k bc^{n-j-k-2}$.
- Similarly, for c^*aA^*, $S = \{c^i bac^{n-2-i} : i = 0, 1 \ldots n-2\}$ is a size $n-1$ fooling set since all words of S are outside the language while one of the corners of the rectangle will be either in $c^*ac^*bc^*$ or in $c^*ac^*ac^*$, both of which are subsets of c^*aA^*.

□

We can also use the same technique to prove a linear lower bound for the communication complexity of monoids outside of **DA** . Still using lemma 5.6, we need only show the lower bound for BA_2 and U. Moreover, since BA_2 divides $U \times U$, any lower bound for BA_2 also holds for U.

Lemma 5.8 $C^{(2)}(BA_2) = \Omega(n)$.

Proof: Equivalently, we show the lower bound for the regular language $L = (c^*ac^*bc^*)^*$. As in lemma 5.7, we consider the partition where player 1 only has access to odd positions in the input. The set $F = \{(cc, ab)^{\frac{n}{2}}\}$ provides an exponential size fooling set. It is clear that $F \subset L$. Now, let $w_1 \neq w_2$ be distinct elements of F. W.L.O.G, suppose that they differ on their first block, i.e. w_1 starts with cc and w_2 with ab. One corner of the rectangle they define starts with cb and is thus outside L. $\Box$

Remark: Alternatively, one can show that $DISJ$ can be recognized by a program of linear length over BA_2. Since $C^{(2)}(DISJ) = \Omega(n)$, lemma 4.3 yields the result.

These bounds yield the following characterization for the 2-party communication complexity of aperiodic monoids:

Theorem 5.9 *Let M be a finite aperiodic monoid*

- $C^{(2)}(M) = \Theta(1)$ *iff M is commutative.*
- $C^{(2)}(M) = \Theta(\log n)$ *iff M is in* **DA** *but is non-commutative.*
- $C^{(2)}(M) = \Theta(n)$ *iff M is not in* **DA** *.*

The multiparty case

The logarithmic upper bound on the 2-party communication complexity of monoids in **DA** holds in the multiparty case. We can however achieve a better bound, and for any monoid M in **DA** , there exists a k such that $C^{(k)}(M) = O(1)$.

Theorem 5.10 *Let $L = A_0^* a_1 A_1^* \ldots A_{k-1}^* a_k A_k^*$ be an unambiguous concatenation, then L has bounded communication complexity in the $(k+2)$-player game.*

Proof: We present a $(k+2)$-player constant cost protocol for this language. Each player first considers the word formed by the letters he has access to, and verifies if it is in L. If so, he outputs a 1 followed by the identities of the players that have access to each letter that he identified as a bookmark, and broadcasts a 0 otherwise. The protocol accepts iff two players named each other.

First, if the input, say w is in the language, we have $w = w_0 a_1 w_1 \ldots a_k w_k$, with $w_i \in A_i^*$. At least two of the $(k+2)$ players see all the bookmarks. They only ignore letters belonging to the w_i's so they will receive inputs in L. Moreover, since the concatenation is unambiguous, they will identify these same bookmarks in their input and will thus name each other, so the protocol will accept w.

Conversely, suppose the protocol accepts some w. Two players, say P_1 and P_2 received inputs in L and have subsequently identified each other. Let w' be the word obtained from w by erasing any letter unknown to P_1 or P_2. We know that w' contains every letter that P_1 identified as a bookmark since P_2 has access to all of them. So w' is in L and it can be properly factorized using the bookmarks of P_1. Similarly, it can be factorized using the bookmarks of P_2. Since the concatenation is unambiguous, the 2 players had the same bookmarks.

Now, every letter in w preceding a_1 must be in A_0 since at least one of P_1 or P_2 see it and both of them can factorize their input using a_1 as the first bookmark. w can thus also be factorized using these k bookmarks so it belongs to L.

The cost of the protocol is clearly independent of the input's length. □

This theorem provides an upper bound on the number of players needed to drop the communication complexity of some monoid in $\mathbf{DA}$ down to $O(1)$. It is consistent with our results that all monoids in $\mathbf{DA}$ require only constant communication for 3 players. However, we believe that for any fixed k, there is a monoid in $\mathbf{DA}$ having non-constant k-party communication complexity.

Some aperiodic monoids have non-constant communication complexity for any fixed k:

Theorem 5.11 *For any fixed k, $C^{(k)}(U) = \Omega(\log n)$.*

Proof: As was shown in [25], arbitrary subsets of A^* can be recognized by a program over U of length $O(n \cdot 2^n)$, essentially corresponding to the function's disjunctive normal form.

For any fixed k, however, there is a function (e.g. GIP_k) of linear k-party communication complexity. Therefore lemma 4.3 implies the logarithmic lower bound. □

6 Conclusion

This paper establishes links between the algebraic structure of finite monoids and the communication complexity of languages that they can recognize. These results lead us to conjecture an analog of Szegedy's Theorem for multiparty communication complexity.

Conjecture 6.1 *A language has constant k-party communication complexity for some constant k iff it can be recognized by a polynomial length program over a monoid in the variety $\mathbf{DA} \vee \mathbf{G_{nil}}$, the smallest variety containing both $\mathbf{DA}$ and all nilpotent groups.*

As a first step, we still need to show that the communication complexity of the monoid BA_2 or, equivalently, of the regular language $(c^*ac^*bc^*)^*$ is non-constant for any constant number of players. We should point out that the variety $\mathbf{DA} \vee \mathbf{G_{nil}}$ is the largest one for which corresponding languages have known characterizations in terms of fixed-length subwords and these are easily accessible in the multiparty model.

It also seems worthwhile to consider the randomized models of communication complexity. The basic results of section 4 hold also in this model, as do our lower bounds in the case of groups since the GIP lower bound was proved for the randomized model.

We only considered in this paper the multiparty case for a constant number of players. As we mentionned earlier, functions in ACC^0 have polylogarithmic communication complexity for a polylogarithmic number of players. From the algebraic point of view, this upper bound holds for solvable monoids and it would be interesting, although certainly difficult, to investigate the communication complexity of non-solvable groups in this case.

References

[1] L. Babai, N. Nisan, and M. Szegedy. Multiparty protocols, pseudorandom generators for logspace, and time-space trade-offs. *J. Comput. Syst. Sci.*, 45(2):204–232, Oct. 1992.

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

[3] D. A. M. Barrington, H. Straubing, and D. Thérien. Non-uniform automata over groups. *Information and Computation*, 89(2):109–132, Dec. 1990.

[4] 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.

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

[6] R. Beigel and J. Tarui. On ACC. *Computational Complexity*, 4(4):350–366, 1994.

[7] A. K. Chandra, M. L. Furst, and R. J. Lipton. Multi-party protocols. In *Proc. 15^{th} ACM STOC*, pages 94–99, 1983.

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

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

[10] V. Grolmusz. Separating the communication complexities of MOD m and MOD p circuits. In *Proc. 33rd IEEE FOCS*, pages 278–287, 1992.

[11] V. Grolmusz. A weight-size trade-off for circuits and MOD m gates. In *Proc. 26th ACM STOC*, pages 68–74, 1994.

[12] V. Grolmusz. On the power of circuits with gates of low L_1 norms. *Theoretical Comput. Sci.*, 188(1–2):117–128, 30 Nov. 1997.

[13] V. Grolmusz. Circuits and multi-party protocols. to appear, 1998.

[14] J. Håstad and M. Goldmann. On the power of small-depth threshold circuits. In *Proc. 31st IEEE FOCS*, volume II, pages 610–618, 1990.

[15] M. Karchmer and A. Wigderson. Monotone circuits for connectivity require super-logarithmic depth. In *Proc. 20th ACM STOC*, pages 539–550, 1988.

[16] S. W. Margolis and J. É. Pin. Minimal non-commutative varieties and power varieties. Technical Report TR-81.49, Université Paris VII, 1981.

[17] P. McKenzie, P. Péladeau, and D. Thérien. NC^1: The automata theoretic viewpoint. *Computational Complexity*, 1:330–359, 1991.

[18] N. Nisan. The communication complexity of treshold gates. (Available from http://www.cs.huji.ac.il/ñoam/papers.html), 1994.

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

[20] J.-É. Pin, H. Straubing, and D. Thérien. Locally trivial categories and unambiguous concatenation. *J. Pure Applied Algebra*, 52:297–311, 1988.

[21] R. Raz and P. McKenzie. Separation of the monotone NC hierarchy. In *Proc. 38thth IEEE FOCS*, 1997.

[22] H. Straubing. *Finite Automata, Formal Logic and circuit complexity.* Boston: Birkhauser, 1994.

[23] M. Szegedy. Functions with bounded symmetric communication complexity, programs over commutative monoids, and ACC. *J. Comput. Syst. Sci.*, 47(3):405–423, 1993.

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

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

[26] A. C. Yao. Some complexity questions related to distributive computing. In *Proc. 11th ACM STOC*, pages 209–213, 1979.

[27] A. C. Yao. On ACC and threshold circuits. In IEEE, editor, *Proc. 31st IEEE FOCS*, pages 619–627, 1990.