An algebraic approach to communication complexity

Pascal Tesson

School of Computer Science
McGill University, Montreal
November, 1998

A thesis submitted to the Faculty of Graduate Studies
and Research in partial fulfillment of the requirements
of the degree of Master of Science.

Abstract

Communication complexity has recently been identified as an interesting tool in the study of small complexity classes. Yao defined a model in which two players, each having access to half the input, try to collaboratively evaluate a function, while minimizing the communication among them. This model, together with its multiparty generalization, has found various applications in complexity.

Close ties between finite algebraic structures and complexity occur in different contexts. The relationship between finite monoids and regular languages, for instance, is well known.

Szegedy proved that languages whose 2-party communication complexity is bounded by a constant are exactly the languages that can be recognized by a program over a finite commutative monoid. This surprising result suggests the possible existence of further links between the communication complexity of a language and the properties of algebraic structures that can recognize it.

In this work, we define the communication complexity of a monoid M as the maximum complexity of any language recognized by M, and show that monoid classes defined by communication complexity classes form varieties. Then we prove that a group G has constant communication complexity for k players if and only if G is a nilpotent group of class at most $(k-1)$, and has linear complexity otherwise. When M is aperiodic, we show that its 2-party communication complexity is constant if M is commutative, logarithmic if M is not commutative but is in the variety **DA**, and is linear otherwise. Moreover, we show that if M is in **DA**, there exists a k such that M has constant k-party communication complexity. We conjecture that this condition is also necessary and prove lower bounds in that direction.

These results lead us to state a conjecture which would generalize Szegedy's theorem to $O(1)$ players. They also suggest several interesting possibilities to further uncover algebraic characterizations of communication complexity classes.

Résumé

La complexité de communication s'est récemment imposée comme un outil intéressant dans l'étude des petites classes de complexité. Yao a défini un modèle, où deux joueurs tentent d'évaluer une fonction, en n'ayant accès chacun qu'à la moitié des données, tout en minimisant la communication entre eux. Ce modèle, et sa généralisation à k joueurs, ont trouvé diverses applications en complexité.

Par ailleurs, les liens entre les structures algébriques finies et la complexité sont nombreux. Par exemple, l'étroite relation entre langages réguliers et monoïdes finis est bien connue.

Szegedy a démontré que les langages dont la complexité de communication est constante pour deux joueurs correspondent exactement aux langages qui peuvent être reconnus par un programme sur un monoïde fini commutatif. Ce résultat surprenant laisse entrevoir l'existence possible de liens entre la complexité de communication d'un langage et les propriétés des structures

algébriques qui peuvent le reconnaître.

Dans ce travail, nous définissons la complexité de communication d'un monoïde M comme étant la complexité maximale d'un langage reconnu par M et montrons que les classes de complexité de communication engendrent ainsi des variétés de monoïdes. Nous démontrons ensuite que la complexité de communication d'un groupe G est constante pour k joueurs si et seulement si G est un groupe nilpotent de classe au plus $(k-1)$ et est linéaire autrement. Dans le cas où M est un monoïde apériodique, nous démontrons que sa complexité de communication à deux joueurs est constante si M est commutatif, logarithmique si M est un élément de la variété $\mathbf{DA}$ mais n'est pas commutatif, et est linéaire autrement. D'autre part, lorsque M est dans $\mathbf{DA}$, nous montrons qu'il existe un entier k tel que M a une complexité constante pour k joueurs. Nous conjecturons que cette condition est également nécessaire et établissons quelques résultats en ce sens.

Ces résultats permettent d'énoncer une conjecture qui généraliserait le théorème de Szegedy pour un nombre de joueurs constant. Ils indiquent également plusieurs possibilités de nouvelles caractérisations algébriques de classes de complexité de communication.

Acknowledgments

First and foremost, I want to express my deep gratitude to my supervisor Denis Thérien. I am grateful to him for getting me interested in complexity theory and giving me the opportunity to work in this field. I have greatly appreciated his trust and enthusiasm, which have guided me in my work over the last year.

Much of the research which lead to this work was done in collaboration with Jean-François Raymond. I am grateful to him for his invaluable help. More importantly, I want to thank him for making my first months of research a fun and rich experience. I also wish to thank Peter Bro Miltersen and Adam Smith for helpful discussions.

I want to thank the administrative and academic staff of the School of Computer Science of McGill University. Through their work and dedication, they provide a great environment for study and research.

I also want to thank NSERC and FCAR for financial support.

Je voudrais également remercier mes parents et ma sœur qui ont toujours su m'offrir leurs encouragements et leurs précieux conseils dans les moments difficiles. Un gros merci aussi à Daniel Dulude et à mes colocs et amis pour le soutien et l'aide qu'ils m'ont offerts lorsque j'en avais besoin.

Contents

Chapter 1

Introduction

1.1 Communication complexity in boolean circuit complexity

Complexity theory is traditionally concerned with measuring the complexity of a function in terms of time and space. In 1979, however, Yao [33] introduced a model of computation in which two players, each having access to half of the input, try to collaboratively evaluate a function on that input while minimizing the communication between them. In this model, time and space required for the players' private computations are irrelevant. This model of communication complexity was originally motivated by applications to the study of VLSI and distributed computing [19].

But communication games have also found very deep applications in various other complexity issues, particularly in small complexity classes such as bounded-width branching programs and restricted circuit classes. Another communication complexity model introduced by Karchmer and Widgerson [17] has allowed the separation of the monotone NC hierarchy [27].

Chandra, Furst and Lipton [6] proposed an interesting extension of Yao's model, in which k-players try to evaluate a k-argument function. Each player has one of the arguments written on his forehead so that it is known to all but himself. Lower bounds in multiparty communication complexity are difficult to prove, but they offer an interesting payoff, as these results can be applied in different contexts such as lower bounds for bounded-width branching programs [6], constructions for pseudorandom generators [3], time-space tradeoffs [3], and lower bounds for hard-to-handle circuit classes of shallow depth [15, 11, 23]. The latter share a common type of argument: circuits of some target class can be evaluated by k players using a small amount of communication so they fail to compute any function of high communication complexity. There is also some hope that this model could, through similar arguments, lead to a separation of ACC^0 from some larger complexity class [15, 26].

1.2 Algebraic methods in complexity

Since the 1950's, the study of algebraic properties of finite monoids has revealed itself as a powerful tool to classify regular languages and understand their structure [10, 24]. In this context, monoids act as language recognizers through morphisms.

In the mid 80's, Barrington [4] introduced the formalism of programs over monoids, which helped establish algebraic characterizations of several natural subclasses of NC^1 [5, 21], by further exploiting ideas from algebraic automata theory.

There were a priori no reasons to suspect the existence of links between monoids and communication complexity, but Szegedy found a surprising characterization of languages requiring only a bounded amount of communication in Yao's 2-party game. He showed that these are exactly the languages that can be recognized by a program over a commutative monoid [30].

It is thus natural to further investigate the relation between communication complexity of languages and algebraic properties of finite monoids that can recognize them via morphisms or programs. In particular, it would be very interesting to know if a result similar to Szegdy's exists for multiparty communication complexity.

1.3 Overview of the work

In this work, we explore the relationship between the algebraic structure of languages and their communication complexity. We define the communication complexity of a monoid M as the maximum complexity of any language it can recognize, and establish connections between this measure and the algebraic properties of M.

Our work is divided into two main parts. First, Chapters 2 and 3 give some background on communication complexity and finite monoids. In Chapter 2, we describe communication complexity models and survey the most important results and proof techniques for these models. In Chapter 3 we present classical tools for studying the algebraic structure of finite monoids and explain how monoids can be considered as language recognizers. We also describe how special monoid classes, called varieties, are used to classify them in terms of their computing power.

In Chapter 4, we establish links between finite monoids and the communication complexity of languages that they recognize via morphisms or programs. We show that communication complexity classes naturally induce monoid varieties. We also discuss Szegedy's theorem, give a brief overview of its proof, and relate it to our results thus far. In Chapter 5, we study the correspondence between varieties of finite monoids and communication classes, first restricting to groups, then to aperiodic monoids. In the case of groups, we prove that all languages recognized by the group G can be decided in constant communication by k players if and only if G is a nilpotent group of class $k-1$. For aperiodic monoids, we show that, in the 2-player game, languages recognized by M will have a communication complexity of $O(1)$ if M is commutative, $\Theta(\log n)$ if M is not commutative but belongs to the variety **DA**, and $\Theta(n)$ otherwise. We also show that for any monoid in **DA**, there is a k such that M has constant k-party communication complexity, and show that not all aperiodic monoids have this property.

In light of these results, we discuss in the conclusion a conjecture which would extend Szegedy's theorem to the multiparty game. Other possibilities for future research along these lines are also presented.

Chapter 2

Communication complexity

In this chapter, we introduce some basic models, proof techniques and important results in communication complexity. For a more in-depth survey of the subject, we refer the reader to [18].

2.1 The 2-party model

In the 2-party game, two players P_1 and P_2, each having unlimited computational power, try to jointly compute a function $f : \{0,1\}^{2n} \to \{0,1\}$.

The set A of input variables is partitioned in two, $A = A_1 \dot\cup A_2$, so that each player sees only variables assigned to him. They communicate according to a fixed protocol and when the communication stops, both players should know the value of f on the given input.

The cost of a protocol is the maximum number, taken over all possible inputs, of bits exchanged between P_1 and P_2. The communication complexity of f with respect to the partition (A_1, A_2), denoted $C^{(2)}_{(A_1,A_2)}$ is the cost of the cheapest protocol computing f given this partition.

The symmetric communication complexity (or just the communication complexity) of f, denoted $C^{(2)}(f)$ is the maximum, taken over all possible partitions of the input variables, of f's communication complexity. That is:

$$C^{(2)}(f) = \max_{A=A_1 \dot{\cup} A_2} C^{(2)}_{(A_1,A_2)}(f)$$

Note that trivially $C^{(2)}(f) \leq n+1$ for any function f. Indeed, this bound is always achieved by the following protocol: one player sends all its bits, and the second one computes f and sends the result.

A communication string is the sequence of bits exchanged by the two parties on a given input.

Fact 2.1.1 *Suppose that the two communication strings induced by a protocol on input pairs (x_1, x_2) and (y_1, y_2) are equal, then this protocol will also generate that same communication string on input pairs (y_1, x_2) and (x_1, y_2).*

The proof of this fact is quite straightforward: notice that the communication sent by either player at any time depends solely on its input and on the communication exchanged so far. It also follows from the fact above that the function takes on the same value on these four inputs.

Communication matrices are a convenient way to visualize the problem of 2-party games. Given a partition $A = A_1 \dot{\cup} A_2$ of the input variables, and an input alphabet X, we build a $|X|^{|A_1|} \times |X|^{|A_2|}$ matrix. Each row represents a possible assignment to the variables in A_1 and each column an assignment to variables in A_2. Each entry in the matrix naturally corresponds to an input x and receives the value $f(x)$. By Fact 2.1.1 every communication string in a protocol must correspond to a homogeneous submatrix (or rectangle) of the communication matrix, i.e. a submatrix in which all entries are equal. For a c-bit protocol, there are at most 2^c different communication strings and so the matrix can be covered by at most 2^c disjoint homogeneous submatrices. This fact is often used to prove lower bounds in 2-party communication complexity. We present here the so-called 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 2.1.2 *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. $\square$

The 2-party model has been thoroughly studied [1, 19] and is well understood. We state here some classical lower bounds of 2-party communication complexity.

Define the equality function EQ, the disjointness function $DISJ$ and the inner product (mod p) function IP_p, all functions from $\{0,1\}^{2n}$ to $\{0,1\}$, as:

- $EQ(x,y) = 1$ iff $x = y$
- $DISJ(x,y) = 1$ iff $\sum x_i y_i = 0$
- $IP_p(x,y) = 0$ iff $\sum x_i y_i = 0 \mod p$

Fact 2.1.3 *Equality, Disjointness and Inner Product all have linear 2-party communication complexity.*

2.2 The k-party model

One intuitive way to extend the 2-player model to allow k players $P_1, \ldots P_k$ is to partition the input variables in k sets: $A = A_1 \dot{\cup} A_2 \dot{\cup} \ldots \dot{\cup} A_k$, such that player P_i only has access to variables of A_i [8, 9]. This can be convenient to study distributed and parallel computing issues. However, the power of this model decreases as the number of players increases.

In [6], Chandra, Furst and Lipton introduced another model of k-party communication complexity. Player P_i has access, to all the variables but the ones in A_i. There is a considerable overlap of information since every variable is now known to $(k-1)$ players. This model is often referred to as the "number in the forehead" model since it is convenient to imagine each player having some part of the input written on his forehead, available to all but himself.

The players have unlimited computational power and they communicate according to a fixed protocol by writing bits on a blackboard seen by all players. Until the computation is over, the protocol should specify which player writes the next bit. This should be a function of the communication thus far, and not of the input. What a party broadcasts should be a function of the information on the board and of the parts of the input it sees.

The notion of protocol cost, communication complexity with respect to a given partition and symmetric communication complexity are defined in the obvious way. We denote the k-party communication complexity of f

as $C^{(k)}(f)$, and define $CC^{(k)}(g(n))$ as the class of functions f such that $C^{(k)}(f) = O(g(n))$.

One can see easily, that when $k = 2$, this model of computation is equivalent to the 2-party model defined in section 2.1. Also note that since $(k+1)$ players can easily simulate k players, we get:

Fact 2.2.1 *If $k < k'$ then $C^{(k)}(f) \geq C^{(k')}(f)$*

The model is thus getting more powerful as the number of players increases. Note also that the trivial protocol yields the following general upper bound:

Fact 2.2.2 $C^{(k)}(f) \leq \frac{n}{k} + 1$

By analogy to communication matrices, we can represent the k-party game as a k-dimensional hypercube. A protocol partitions this cube into cylinder intersections, as defined by Babai, Nisan and Szegedy [3].

Definition: Given a partition of the input variables, $A = A_1 \dot{\cup} \ldots \dot{\cup} A_k$, we say that a subset S of the input space is a cylinder in the i^{th} dimension if membership in S is independent of the variables in A_i. We say that S is a cylinder intersection if it can be written as the intersection of k cylinders.

One can easily verify that

Fact 2.2.3 *Any c-bit multiparty protocol partitions the input space into at most 2^c homogeneous cylinder intersections.*

Unfortunately, it is very difficult to prove a non-trivial lower bound on the number of cylinder intersections required to partition a specific communication hypercube, because their structure is not as rigid as the communication matrices' rectangles. And although it has been shown [12] that for any k most boolean functions have linear communication complexity, lower bounds for explicit functions are rare for any $k \geq 3$. However, these few lower bounds have many applications in diverse areas of complexity.

Using Ramsey-theoretic arguments, Chandra, Furst and Lipton first established in [6] an $\omega(1)$ lower bound on the k-party communication complexity of the $EXACTLY_N^k$ function, defined as

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

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

In [3], 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 branching programs.

Grolmusz [11] 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 only compute functions with constant k-party communication complexity for any $k \geq p$, 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.

2.3 Variations on the model

One can define the non-deterministic communication complexity of a function f as the minimum number of bits that the players need to exchange in order to be convinced that $f(x_1, \dots, x_k) = 1$. Naturally, co-non-deterministic communication complexity can be defined symmetrically. It is easy to see, for instance, that EQ and $DISJ$ have logarithmic co-non-deterministic 2-party communication complexity. These upper bounds are optimal since the gap between determinism and non-determinism can be shown to be at most exponential.

One can also define different probabilistic models of communication complexity by allowing the players to flip coins. The string of random bits can be taken to be public or private (see [22] for a comparison of the two), and we can allow either one-sided or two-sided error. In contrast to the non-deterministic model, EQ has been shown to have logarithmic one-sided error 2-party communication complexity when private coins are used and only constant complexity when public coins are allowed. $DISJ$ [16, 28] and IP_2 [7] however have linear complexity in both cases.

The ϵ-distributional communication complexity, introduced in [34], is yet another interesting variation on this model. It is defined as the minimum number of bits that players need to exchange in order to correctly compute a function on a $(1+\epsilon)/2$ fraction of the input space. The $GIP_{k,p}$ lower bound was actually proved for this model. Håstad and Goldmann [15] and later Grolmusz [13, 14] used this lower bound and discriminator arguments to show exponential lower bounds on the size of certain restricted depth-3 circuits with a threshold gate as output that compute $GIP_{k,2}$. Babai, Nisan and Szegedy [3] had originally exploited this lower bound to build an explicit pseudorandom generator for Logspace.

While the above mentioned extensions of the basic model increase the power of the players, simultaneous communication complexity [2] reduces it. Here, the communication sent by a player is restricted to be independent of the other players' communication. We can imagine each player privately sending its communication to a referee which sees none of the input. The referee should then be able to compute the value of the function based solely on the messages sent to him by the players. This model has received considerable attention since Håstad and Goldmann noted in [15] that any function in ACC^0 has poly-logarithmic simultaneous communication complexity for some poly-logarithmic number of players.

Chapter 3

Finite monoids

This chapter introduces concepts related to algebraic automata theory. We refer the reader to [10, 24] for a detailed discussion of these issues.

3.1 Monoids and morphisms

A monoid M is defined as a set together with an associative binary operation which has an identity. All monoids considered in this chapter are finite, with the exception of X^*, the free monoid generated by X.

A morphism ϕ from a monoid M to a monoid N is a function from M to N mapping the identity of M to the identity of N and such that for any $x, y \in M$ we have $\phi(xy) = \phi(x)\phi(y)$.

Given two monoids M and N, we say that M divides N, and write $M \prec N$, if M is a morphic image of a submonoid of N.

The operation on a monoid M defines a canonical surjective morphism $eval_M$: $M^* \to M$

$$eval_M(m_1 m_2 \dots m_n) = m$$

where $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$ is the product in M of this sequence of monoid elements.

For every $m \in M$, we define the word problem of m as the language $\{w \in M^* : eval_M(w) = m\}$.

We say that a language $L \subseteq X^*$ is recognized by the monoid M if there is a morphism $\phi : X^* \to M$ such that $L = \phi^{-1}(F)$ for some subset $F \subseteq M$.

Given a regular language $L \subseteq X^*$, define for any word x the set $Y_x = \{(u, v) : uxv \in L\}$. We get a congruence of finite index on X^* by declaring $x \sim_L y$ iff $Y_x = Y_y$. The monoid $X^* / \sim_L$ is called the syntactic monoid of L, and is denoted $M(L)$.

Fact 3.1.1 ([10]) *The monoid $M(L)$ recognizes L and is minimal with that property, that is for any N also recognizing L, $M(L) \prec N$.*

The theorem of Kleene can be restated as saying that L is regular iff it can be recognized by some finite monoid M.

Green's relations are a convenient way to analyze the algebraic structure of a finite monoid. We define, for any monoid M, four equivalence relations:

1. $x\mathcal{R}y$ iff x and y generate the same right ideal, that is $xM = yM$.

2. $x\mathcal{L}y$ iff x and y generate the same left ideal, that is $Mx = My$.

3. $x\mathcal{J}y$ iff x and y generate the same two-sided ideal, that is $MxM = MyM$.

4. $x\mathcal{H}y$ iff $x\mathcal{R}y$ and $x\mathcal{L}y$.

We can define a natural pre-order on these equivalence classes using set inclusion. We will write $a \leq_{\mathcal{R}} b$ whenever $aM \subseteq bM$, and $a <_{\mathcal{R}} b$ when $aM \subset bM$. Note that, trivially, $ab \leq_{\mathcal{J}} a$, $ab \leq_{\mathcal{J}} b$, $ab \leq_{\mathcal{R}} a$ and $ab \leq_{\mathcal{L}} b$.

We will also denote $\mathcal{R}_a$ the $\mathcal{R}$-class of a.

Fact 3.1.2 *$\mathcal{R}$ is a left congruence, i.e. $a\mathcal{R}b$ implies $ca\mathcal{R}cb$ for any $c \in M$.*

$\mathcal{L}$ is a right congruence, i.e. $a\mathcal{L}b$ implies $ac\mathcal{L}bc$ for any $c \in M$.

Lemma 3.1.3 *$\mathcal{R}$ and $\mathcal{L}$ commute, that is $\mathcal{R} \circ \mathcal{L} = \mathcal{L} \circ \mathcal{R}$.*

Let $\mathcal{D} = \mathcal{R} \circ \mathcal{L}$. If the monoid M is finite then $\mathcal{D} = \mathcal{J}$.

Proof: Suppose $a\mathcal{L}c\mathcal{R}b$. There exist $u, v \in M$ s.t. $a = uc$ and $cv = b$. So $av = ucv = ub$. Call this monoid element d. We show that $a\mathcal{R}d\mathcal{L}b$. Since $\mathcal{L}$ is a right congruence, $a\mathcal{L}c$ implies $d = av\mathcal{L}cv = b$. Similarly, $b\mathcal{R}c$ implies $d = ub\mathcal{R}uc = a$.

For the second part of the lemma, note first that if $a\mathcal{D}b$ then $a\mathcal{R} \circ \mathcal{L}b$, so $a\mathcal{J} \circ \mathcal{J}b$, and by transitivity of $\mathcal{J}$, $a\mathcal{J}b$.

Conversely, suppose $a\mathcal{J}b$. There exist $u, v, x, y \in M$ s.t. $b = uav$ and $a = xby$, so $a = (xu)a(vy) = (xu)^k a(vy)^k$ for all k. Since M is finite, we can choose k s.t. both $(xu)^k$ and $(vy)^k$ are idempotent, i.e. $(xu)^k = e = e^2$ and $(vy)^k = f = f^2$. We have $a = eaf$, so multiplying by e on both sides we get $ea = a$ and similarly $af = a$.

Now, $ua\mathcal{L}a$ since $(xu)^{k-1}xua = ea = a$. Similarly $av\mathcal{R}a$. Therefore $b = uav\mathcal{L}av\mathcal{R}a$, and so $a\mathcal{D}b$. $\square$

Lemma 3.1.4 (Green) *Suppose a and b are two elements of the same $\mathcal{R}$-class, i.e. there exist u, v s.t. $au = b$ and $bv = a$. Denote by $\rho_u : M \to M$ the function defined by $\rho_u(s) = su$. Then ρ_u and ρ_v are bijections from $\mathcal{L}_a$ to $\mathcal{L}_b$ and from $\mathcal{L}_b$ to $\mathcal{L}_a$ respectively.*

Moreover $\rho_u = \rho_v^{-1}$ and they preserve $\mathcal{H}$-classes.

Proof: Suppose $x\mathcal{L}a$. $\rho_u(x) = xu\mathcal{L}au = b$. Now let $x = ta$, then $\rho_v \circ \rho_u(x) = xuv = tauv = tbv = ta = x$.

The $\mathcal{H}$-classes are obviously preserved since $x\mathcal{R}\rho_u(x)$. $\square$

These facts allow the so-called "eggbox" representation of finite monoids. Each $\mathcal{J}$-class of the monoid is represented as a table where rows correspond to $\mathcal{R}$-classes, columns to $\mathcal{L}$-classes and cells to $\mathcal{H}$-classes. Since $\mathcal{J} = \mathcal{L} \circ \mathcal{R} = \mathcal{R} \circ \mathcal{L}$, we know that none of the cells are empty.

Idempotent elements play a crucial role in the algebraic structure of a monoid.

Lemma 3.1.5 *Let $e = e^2$ be an idempotent. $a \leq_{\mathcal{R}} e$ iff $ea = a$. Similarly $a \leq_{\mathcal{L}} e$ iff $ae = a$.*

Proof: Only one direction requires a proof. If $a \leq_{\mathcal{R}} e$, there is an s s.t. $a = es$. We have $ea = ees = es = a$. $\square$

Lemma 3.1.6 *Let $a\mathcal{J}b$. Then $ab \in \mathcal{R}_a \cap \mathcal{L}_b$ iff $\mathcal{L}_a \cap \mathcal{R}_b$ contains an idempotent $e = e^2$.*

Proof: If there is an idempotent e in $\mathcal{L}_a \cap \mathcal{R}_b$, we have $ae = a$, $eb = b$. By Green's lemma $aeb \in \mathcal{R}_a \cap \mathcal{L}_b$ and $aeb = (ae)(eb) = ab$.

Conversely, if $ab \in \mathcal{R}_a \cap \mathcal{L}_b$, then, by Green's lemma, there is an f in $\mathcal{L}_a \cap \mathcal{R}_b$ such that $fb = b$. By lemma 3.1.5, since $b \leq_R f$, f must be idempotent. $\square$

An element a of monoid M is said to be regular if there exists some $x \in M$ such that $axa = a$. A $\mathcal{J}$-class is said to be regular if all its elements are regular.

Lemma 3.1.7 *The following are equivalent for a $\mathcal{J}$-class J of a finite monoid:*

1. *J is regular*
2. *J contains a regular element*
3. *Every $\mathcal{R}$-class and every $\mathcal{L}$-class of J contains an idempotent.*
4. *J contains an idempotent.*

Proof:

$1 \rightarrow 3$: Suppose $asa = a$, then $(as)(as) = as\mathcal{R}a$. Similarly, if a is regular then there is an idempotent f with $a\mathcal{L}f$.

$3 \rightarrow 4$ is trivial

$4 \rightarrow 2$ Every idempotent e is regular since $eee = e$

$2 \rightarrow 1$ Suppose $a\mathcal{J}b$ and a is regular. There exists c s.t. $a\mathcal{R}c\mathcal{L}b$. Reusing our first argument, we know that there exists an idempotent e s.t. aRe. Now $c\mathcal{R}e$ so $cx = e$ and $c = ec = eec = cxec$ so c is regular. Hence, the $\mathcal{L}$-class of c contains an idempotent f. Since $b\mathcal{L}f = f^2$, b is regular.

3.2 Monoid varieties

We say that a class $\mathbf{V}$ of finite monoids forms a variety if it is closed under division and finite direct product. 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 from one free monoid to another. (i.e. If $L \subseteq A^*$ is in $\mathbf{L}$, and $\phi : B^* \to A^*$ is a monoid morphism, $\phi^{-1}(L)$ is in $\mathbf{L}$).

Fact 3.2.1 *For every language variety* $\mathbf{L}$ *there is a monoid variety* $\mathbf{V}$ *such that a language* L *is in* $\mathbf{L}$ *iff* $M(L)$ *is in* $\mathbf{V}$.

Varieties are thus the natural unit to classify finite monoids and their power as language recognizers via morphisms. We will give here some examples of varieties that will be of particular interest in this work and relate them to their corresponding language varieties.

A monoid is said to be aperiodic if no subset of it forms a non-trivial group. Equivalently, a monoid is aperiodic if all its $\mathcal{H}$-classes consist of a single element. As a third alternative definition, a monoid is aperiodic iff it satisfies $m^t = m^{t+1}$ for some $t \geq 0$.

We say that a language is star-free if it can be written as a regular expression without using $*$. (Only boolean operations, including complementation, and concatenation are accepted.)

Fact 3.2.2 ([29]) *A language is star-free iff it can be recognized by an aperiodic monoid.*

We define the variety **DA** as the class of aperiodic monoids M such that if a $\mathcal{J}$-class of M contains an idempotent, i.e. if this $\mathcal{J}$-class is regular, then all elements of that class are idempotents. Equivalently, a monoid is in **DA** iff it satisfies $(stu)^n t(stu)^n = (stu)^n$ for some $n \geq 1$.

The concatenation $L = A_0^* a_1 A_1^* a_2 A_3^* \ldots a_k A_k^*$, where $a_i \in A$, $A_i \subseteq A$, is said to be unambiguous if whenever w is in L, the factorization $w = w_0 a_1 w_1 a_2 \ldots a_k w_k$, where $w_i \in A_i^*$, is unique.

Fact 3.2.3 ([25]) *A regular language $L \subseteq A^*$ is recognized by M in* **DA** *iff L is a disjoint union of languages of the form $A_0^* a_1 A_1^* a_2 A_3^* \ldots a_k A_k^*$, where the concatenation is unambiguous.*

We need here to present a technical lemma about aperiodic monoids.

Lemma 3.2.4 *Let M be a non-commutative aperiodic monoid. M is divided by one of the following monoids.*

1. *The syntactic monoid of* $c^*ac^*bc^*$.

2. *The syntactic monoid of the unambiguous concatenation* c^*aA^*.

3. *The syntactic monoid of its reverse* A^*ac^*.

4. *The syntactic monoid, denoted* BA_2, *of* $(c^*ac^*bc^*)^*$.

5. *The syntactic monoid, denoted* U, *of* $((b+c)^*a(b+c)^*b(b+c)^*)^*$.

Moreover, if $M \notin \mathbf{DA}$, *then* M *is divided by one of the last two.*

Proof: Let us look first at the eggbox representation of BA_2 and U. Idempotent elements are marked by $*$.

BA_2:

*1

*ab	a
b	*ba

*0

U:

*1

*ab	a
*b	*ba

*0

The two monoids have six elements, a single non-trivial $\mathcal{J}$-class and differ only in the structure of that $\mathcal{J}$-class.

Suppose first that M is not in the variety **DA**. Then M has a non-trivial regular $\mathcal{J}$-class containing at least one non-idempotent element. Since each of its $\mathcal{R}$ and $\mathcal{L}$ classes contains at least one idempotent, this $\mathcal{J}$-class can be put in one of the following forms:

*c	a	...
b	*d	...
⋮	⋮	

Case 1

*c	a	...
*b	*d	...
⋮	⋮	

Case 2

In the first case, there is exactly one idempotent per $\mathcal{R}$-class. In the second case, at least one $\mathcal{R}$-class contains at least two idempotents.

We consider the submonoid N generated by $\{1, a, b, c, d\}$. In the first case, using lemma 3.1.6, we know that $ab = c$, $ba = d$ and cd, dc fall outside this $\mathcal{J}$-class. We can thus define a morphism from this submonoid N onto BA_2, by mapping a to a, b to b, c to ab, d to ba, and every other element to 0. This morphism is well defined and clearly surjective.

In the second case, using fact 3.1.6, we know that $ab = c$, $ba = d$, $cd = a$ and dc falls outside this $\mathcal{J}$-class. Similarly, we can define a surjective morphism from N onto U, by mapping a to a, b to b, c to ab, d to ab, and every other element to 0.

For the case when M is in **DA**, the same type of argument can yield the result, and we omit the proof. Margolis and Pin also provide an alternative proof of this fact in [20]. $\square$

Remark: The regular languages used to define BA_2 and U are similar. The language $(c^*ac^*bc^*)^*$ requires that a's and b's alternate (the c's being considered as blank) while $((b+c)^*a(b+c)^*b(b+c)^*)^*$ only asks that there be a b between any two a's. Of course a's and b's alternate iff there is at least one a between any two b's and at least one b between any two a's, and this observation can be used to prove $BA_2 \prec U \times U$.

We say that a monoid M is a group if there is a $q \geq 1$ such that every $m \in M$ satisfies $m^q = 1$. For any group G, we can form the sequence $G_0 = G$, $G_1 = [G, G]$, 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 $g_1, g_2 \ldots g_{k+1} \in G$,

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

For any $k \geq 0$, nilpotents of class k form a variety, and nilpotents of class 1 are precisely the commutative groups.

The combinatorial description of languages recognized by nilpotent groups of class k was proved in [31].

We say that $u = u_1 u_2 \ldots u_m$ is a subword of x if x can be factorized as $x = w_0 u_1 w_1 u_2 \ldots u_m w_m$, where $w_i \in A^*$. Let $u \in A^*$, $q \geq 1$. Define $|x|_u$ as the number of occurrences of u as a subword of x. Also, define $[u, i, q] = \{x;\ |x|_u \equiv i \pmod{q}\}$.

Denote by $\mathbf{G_{nil,m}}$ the variety of finite nilpotent groups of class $\leq m$.

Fact 3.2.5 *$L \subseteq A^*$ is recognized by $G \in \mathbf{G_{nil,m}}$ iff L is in the boolean algebra generated by $\{[u, i, q];\ |u| \leq m,\ q \geq 1\}$. Equivalently, $\mathbf{G_{nil,m}}$ is generated by the syntactic monoids of the languages $[u, i, q]$ where $|u| \leq m$.*

3.3 Programs over monoids

The concept of recognition of a language by a morphism can be extended using programs over monoids. Formally, given an input alphabet A, an n-input program of length l over a monoid M is a sequence of instructions $(i_1, f_1), (i_2, f_2) \ldots, (i_l, f_l)$ with $i_j \in \{1, \ldots, n\}$ and $f_j : A \to M$, together with an accepting subset $F \subseteq M$. The program accepts the input $x_1 x_2 \ldots x_n$

iff the sequence $f_1(x_{i_1})\dots f_l(x_{i_l})$ evaluates to an element of F.

We say that $L \subseteq A^*$ is P-recognized by M if there exists a sequence $(\varphi_1, \varphi_2, \dots)$ where φ_n is an n-input program over M, of length $l(n)$ with $l(n)$ polynomial in n, such that the program φ_n accepts $L \cap A^n$.

We say that the monoid M P-divides monoid N and write $M \prec_P N$ if any language recognized by a morphism over M can be recognized by a polynomial length program over N. If $M \prec N$ then $M \prec_P N$ since morphisms are just special cases of programs. Equivalently, $M \prec_P N$ if any word problem of M can be recognized by a polynomial length program over N. We say that a class $\mathbf{V}$ of monoids is a P-variety if it is closed under finite direct product and P-division.

It was shown by Barrington [4] that a language is in NC^1 iff it can be recognized by a polynomial length program over any non-solvable finite group. Similar ideas were then used [5, 21] to relate other natural varieties to natural subclasses of NC^1.

Lemma 3.3.1 *Commutative monoids form a P-variety.*

Proof: It is clear that this class is closed under finite direct product so it is sufficient to show that it is closed under P-division.

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

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

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.

Suppose now that $M \prec_P N$, and that N is commutative, but M is not. There exist $a, b \in M$ such that $ab \neq ba$. Let φ be a program over N recognizing the word problem $\{w \in M^* : eval_M(w) = ab\}$.

Since the f_i's map M to N and because 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 φ_n contains the same function $f_i = f_j$, for two different instructions, $i \neq j$. Consider the two following length n words of M^*: $11\ldots1a1\ldots1b1\ldots1$ and $11\ldots1b1\ldots1a1\ldots1$, with a and b in positions i and j. Only the first one is equal to ab in M, but since N is commutative, φ_n outputs the same result on both inputs, a contradiction. The class of commutative monoids is thus closed under P-division. $\square$

Chapter 4

Finite monoids varieties in communication complexity

This chapter formally introduces the definition of communication complexity of a monoid and relates communication complexity classes to monoid varieties. We also present Szegedy's algebraic characterization of functions of bounded 2-party communication complexity.

4.1 Communication complexity of a finite monoid

In this section, the number of players k is fixed to some arbitrary finite value, and will be omitted when convenient.

We define the communication complexity of a monoid M, which we denote $C^{(k)}(M)$, as the maximum complexity of any language recognized by M via morphism.

Theorem 4.1.1 *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, we give a $O(c_{eval_M})$-bit k-party protocol for L.

Let $\phi : A^* \to M$ be the recognizing morphism. Given a length n input in A^*, each player privately applies ϕ to each letter of the input. They now have a word of length n in M^*. Using the c_{eval_M}-bit protocol they can evaluate this product. Finally, each player privately checks whether or not the result is in F, where $\phi^{-1}(F) = L$.

For the other direction, we know that each monoid element is an equivalence class of the syntactic congruence of L. Since M is finite, there are only $|M|$ equivalence classes of $\sim_L$. Moreover, each such class corresponds to some unique Y_x, and two different Y's must disagree on some pair (u, v). We can thus 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 $eval_M$.

The players have agreed beforehand on a choice of pairs (u_i, v_i) and, for each $m \in M$ on a word $w \in A^*$ of length q such that $\phi(w) = m$. (Note that this can be done since the existence of an identity letter allows "padding".)

The players are given a length n input in M^*, say $m_1, \ldots, m_n$. Privately, the players translate this input 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 pair (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. Also, since $c_L = O(n)$, we have $c_L(qn) = O(c_L(n))$. Hence $c_{eval_M} = O(c_L)$. $\square$

This theorem shows that $C^{(k)}(M) = \Theta(C^{(k)}(eval_M))$. This is very convenient as $eval_M$ provides an explicit representation for $C^{(k)}(M)$. In fact, any language L having an identity letter and such that $M(L) = M$ will have the same property. This fact will be used more or less implicitly in most of our proofs.

Remark: The identity letter hypothesis can be weakened. It is clear from the proof of the theorem that it is sufficient to require that there exist a $q \geq 1$ such that for every $m \in M$, $\phi^{-1}(m)$ contains a word w of length q, that is every m is the image of some word of length q.

The result, however, does not hold if we remove this condition. Some examples are provided in Chapter 5.

Lemma 4.1.2 *Let M, N be finite monoids. Let c_M, c_N and $c_{M\times N}$ be the communication complexity of M, N and $M \times N$ respectively. Then*

$$\max\{c_M, c_N\} \leq c_{M\times N} \leq c_M + c_N$$

Proof: First, we can obviously build a protocol of cost $c_{M\times N}$ for $eval_M$, using the identity in N for the N-coordinate. This shows the first inequality. To show $c_{M\times N} \leq c_M + c_N$, we build a cost $(c_M + c_N)$ protocol for $eval_{M\times N}$ by evaluating the product coordinate-wise using the c_M and c_N-bit protocols. $\square$

Lemma 4.1.3 *Let M, N be finite monoids such that $M \prec N$. Then $c_M \leq c_N$.*

Proof: Since $M \prec N$, there is a submonoid $T \subseteq N$ and a surjective morphism $\phi : T \to M$. Given a word in M^*, the k players want to evaluate the product $m_1 m_2 \dots m_l$ in M. Since ϕ is surjective, the players can privately translate the 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

$$t_1 t_2 \dots t_l = t'$$

and privately compute $\phi(t')$. Finally, since ϕ is a morphism, we have $\phi(t') = m_1 m_2 \dots m_l$. Note that t' is in T, since T is a submonoid and is therefore closed under the operation of N. $\Box$

Lemma 4.1.4 *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^*.

Corollary 4.1.5 *Suppose $M \prec_P N$ and $c_N = O(\log^r(n))$, for some r; then $c_M = O(c_N)$.*

Proof: Recall that $M \prec_P N$ if any word problem of M can be recognized by a polynomial length program over N. Note also that $\log^r(n^d) = O(\log^r n)$, so this is a straightforward application of the lemma. $\Box$

Lemma 4.1.4 also yields a result used implicitly in many applications of communication complexity results to boolean circuit complexity. Following the notation of [21] 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 4.1.6 *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 4.1.4 gives a contradiction. $\square$

Lemmas 4.1.2 and 4.1.3 and corollary 4.1.5 also reveal the closure properties of monoid classes induced by communication complexity classes.

Theorem 4.1.7 *The monoids in $CC^{(k)}(f(n))$, that is monoids whose k-party communication complexity are $O(f(n))$, form a monoid variety. Moreover they form a P-variety if $f(n)$ is polylogarithmic.*

We are interested here in the varieties defined by the following communication complexity classes:

$$CCC^{(k)} = CC^{(k)}(O(1)) \text{ and } CCC = \bigcup_{k \geq 2} CCC^{(k)}$$

Our interest for classes of bounded communication is motivated by the result of Szegedy.

4.2 Szegedy's Theorem

In [30], Szegedy proved an algebraic characterization for the class $CCC^{(2)}$. His main theorem is:

Theorem 4.2.1 (Szegedy) *A language L has bounded 2-party communication complexity iff L can be recognized by an M-program over a commutative monoid M.*

Szegedy's proof is extremely subtle, and we will not try to describe it in full detail. We give a brief overview of the structure and ideas involved.

In light of our discussion so far, the *if* direction should seem fairly clear. The 2-party communication complexity of a commutative monoid is easily seen to be no more than $2 \cdot \log(|M|)$, since commutativity allows us to assume that one player has a prefix of the string of monoid elements. He can thus evaluate this product and send the result to the second player who, in turn, evaluates the rest of the product and sends his result. Lemma 4.1.4 completes the proof.

The *only if* part of the proof is not as simple. Szegedy uses the fact that any function f of bounded 2-party communication complexity has a bounded number of distinct columns in its communication matrix, thus a bounded number of subfunctions. When f is symmetric, this allows him to show that f is almost periodic, that is there exists integers a, b, p such that f is periodic of period p between the limits a and $n - b$. In that case, it is easy to show that f can be computed by a program over the direct product of three commutative monoids $M_1 \times M_2 \times M_3$, each having one generator and a defining relation: $M_1 = \langle g^p = 1 \rangle$, $M_2 = \langle g^{a+1} = g^a \rangle$, $M_3 = \langle g^{b+1} = g^b \rangle$.

The reduction from the general case to the symmetric case is very intricate. One can define an equivalence relation on the set of input variables by letting $x_i \sim x_j$ if exchanging the values of x_i and x_j does not change the value of f, for any possible setting of the input variables. The number of symmetricity classes of f can be bounded in terms of the number of subfunctions of f and is thus bounded by some constant q. This allows us to write f as $f = g(f_1, f_2, \ldots f_q)$, where the f_i's are symmetric. Since f_i can be computed by a program over a commutative monoid N_i, f can be computed by a program over the commutative monoid $N_1 \times N_2 \times \ldots \times N_q$.

Since commutative monoids form a P-variety, we get as a consequence of this theorem:

Corollary 4.2.2 *A finite monoid N has constant communication complexity iff N is commutative.*

In light of Szegedy's theorem, it seems worthwhile to further investigate the algebraic properties of monoid varieties induced by communication complexity classes.

Chapter 5

Bounds on the communication complexity of specific monoids

This chapter gives upper bounds and lower bounds on the communication complexity of specific monoids. We study separately the cases of groups and aperiodic monoids and describe what monoid varieties are induced by different communication complexity classes.

5.1 Groups

We prove in this section that groups having constant k-party communication complexity are exactly the nilpotent groups of class $(k-1)$. We also show that all other groups have linear k-party communication complexity for a fixed k.

First recall from section 3.2 that a language L is recognized by a nilpotent group G of class less or equal to k iff it is in the boolean algebra generated by $\{[u, i, q]; |u| \leq k, q \geq 1\}$. So in order to determine membership in L, it is sufficient to count, mod some integer, the number of occurrences of given subwords of length at most k.

But in the $(k+1)$-player game, every subword of length k can be seen entirely by at least one of the players. There is thus a protocol of bounded cost for $(k+1)$ players to count mod q, for any q , the number of occurrences of a subword u of length k: the players agree beforehand on a partition of the $\binom{n}{k}$ k-tuples of input bits such each player is assigned only some k-tuples which he sees entirely. They can each subsequently broadcast the number mod q of occurrences of u in their respective k-tuples. Hence if $G \in \mathbf{G_{nil,k}}$, then $C^{(k+1)}(G) = O(1)$.

Nilpotents of class k are actually the only groups in $CCC^{(k+1)}$. We will prove this using the linear lower bound on the k-party communication complexity of $GIP_{k,p}$.

Lemma 5.1.1 *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 $[[\ldots[a_1, a_2], a_3], \ldots, a_k]$ 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 $((i_1, f_1)(i_2, f_2)\ldots(i_l, f_l))^{-1}$ as $(i_l, f_l)^{-1}\ldots(i_1, f_1)^{-1}$. 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 $[[\ldots[a_1, a_2], a_3], \ldots, a_k]$. 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 5.1.2 *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 sum 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 4.1.4 we get $C^{(k)}(G) = \Omega(n)$.

We have thus proved:

Theorem 5.1.3 *Let G be a finite group. G has constant k-party communication complexity if and only if G is nilpotent of class $< k$. Otherwise $C^{(k)}(G) = \Omega(n)$.*

Using Theorem 4.1.1, we can restate this result in terms of regular languages as:

Corollary 5.1.4 *Let L be a regular language such that $M(L)$ is a group. If L has an identity letter, then L has constant k-party communication complexity* *iff $M(L)$ is nilpotent of class $< k$, and has linear complexity otherwise.*

We want to stress that the identity letter hypothesis here is crucial.

Example: Let $L = \{x \in \{a,b\}^*; |x|_{ab} \equiv 0 \pmod 2\}$. The syntactic monoid of L is D_4, a nilpotent group of class 2, corresponding to rigid motions of the square. $\varphi(a)$ and $\varphi(b)$ are reflections along axes forming a 45^o angle.

However L has constant 2-party communication complexity. Indeed, the parity of $|x|_{ab}$ is a function of the number $\pmod 4$ of a's and b's in odd and even positions in the input. First, notice that since in D_4

$$aa = bb = abba = baab = 1 \text{ and } abab = baba$$

we can permute pairs of input bits without affecting membership in L and thus assume that inputs have the canonical form

$$aa \ldots aa \; ab \ldots ab \; ba \ldots ba \; ab \ldots ab \; bb \ldots bb$$

Also, interchanging an a and a b in odd positions will not affect membership in L since we have:

$$aaab = ab = abaa$$

$$aabb = 1 = abba$$

$$baab = 1 = bbaa$$

$$babb = ba = bbba$$

Similarly, letters at even positions can be arbitrarily shuffled. Finally, we can replace 4 a's at even positions by 4 b's since:

$$aaaaaaaa = 1 = abababab$$

$$aaaaaaba = ba = abababbb$$

$$aaaababa = baba = abab = ababbbbb$$

$$aabababa = ab = abbbbbbb$$

$$babababa = 1 = bbbbbbbb$$

So a word $w \in \{a,b\}^*$ of even length is in L iff any word of even length, having the same number (mod 4) of a's and b's in odd and even positions as w, is also in L. There is thus an easy constant cost 2-party protocol for this language.

5.2 Aperiodic monoids

We now investigate the communication complexity of aperiodic monoids. The situation here is not as clear as in the group case. In the 2-party game, we provide a complete characterization for all aperiodic monoids. We then present some results for the situation where $O(1)$ players are involved and present some examples of regular languages in which the absence of an identity letter can be exploited to obtain low-cost protocols.

The case with 2 players

By corollary 4.2.2, we know that any monoid, hence any aperiodic monoid, has constant communication complexity iff it is commutative. We now proceed to show that every non-commutative aperiodic monoid has communication complexity of either $\Theta(\log n)$ or $\Theta(n)$, depending if the monoid belongs to the variety **DA** or not. Recall that a language L is recognized by a monoid M in **DA** iff it is the disjoint union of unambiguous concatenations $A_0^* a_1 A_1^* \ldots a_t A_t^*$. We refer to the a_i's as bookmarks.

Lemma 5.2.1 *Let $L = A_0^* a_1 A_1^* a_2 \ldots a_t A_t^*$ be an unambiguous concatenation. Then $C^{(2)}(L) = O(\log n)$.*

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

For $t = 1$, notice that the only non-trivial unambiguous concatenations $A_0^* a_1 A_1^*$ are $c^* a A^*$ and its reverse $A^* a c^*$, 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 $A_0^*a_1A_1^*a_2\ldots a_tA_t^*$ is an unambiguous concatenation, then so is $A_{i-1}^*a_iA_i^*a_{i+1}\ldots a_jA_j^*$, for any $1 \leq i < j \leq t$. We will show that two players can always, using $O(\log n)$ bits of communication, reduce the problem for t bookmarks to two problems of less than t bookmarks.

Since $A_0^*a_1A_1^*a_2\ldots a_tA_t^*$ is an unambiguous concatenation, there must be some a_p which is not in both A_0 and A_t. If this was not the case, then the word $a_1a_2\ldots a_ta_1a_2\ldots a_t$ could be factorized either as $a_1a_2\ldots a_t(a_1a_2\ldots a_t)$ or as $(a_1a_2\ldots a_t)a_1a_2\ldots a_t$.

So suppose W.L.O.G. that $a_p \notin A_t$. 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 $0 < l < t$, but that $a_p \notin A_j$ for all $j \neq l$. Then the input w can be factorized as $w = w_0a_1w_1\ldots a_tw_t$ 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 $A_i^*a_{i+1}\ldots a_tA_t^*$. By the 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 $A_0^* a_1 \dots a_l A_l^*$ and a second one in $A_l^* a_{l+1} \dots a_t A_t^*$.

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_{t-1} and see if they can factorize these two segments of w appropriately, if not assume it is in w_{t-2}, and so on. Also, if $l = 0$ or if a_p belongs to none of the A_i's then the last occurrence of a_p must necessarily be a bookmark. □

Hence, monoids in **DA** have logarithmic 2-party communication complexity. To show the matching lower bound for non-commutative monoids of **DA**, we only need to establish a logarithmic lower bound on the 2-party communication complexity of the languages $c^*ac^*bc^*$ and c^*aA^*, because of lemma 3.2.4.

Lemma 5.2.2 *Let $L = c^*ac^*bc^*$, we have $C^{(2)}(L) = \Omega(\log n)$.*

Proof: Consider the partition where player 1 gets odd-indexed bits and player 2 gets even-indexed bits. The following is a size $\frac{n}{2}$ fooling set for L: $F = \{c^{2i}bac^{n-2i}\}$ None of these words are in L, but for any two words of this set, one of the words in the rectangle they define is of the form $c^j ac^l bc^{(n-j-l-2)}$, which is in L. □

Lemma 5.2.3 *The unambiguous concatenation $K = c^*aA^*$ on the alphabet $A = \{a, b, c\}$ requires $\Omega(\log n)$ communication for two players.*

Proof: For the partition where player 1 gets only odd-indexed bits, one can provide a size n fooling set of similar nature: $F = \{c^i bac^{n-2-i}\}$. None of these are in K, but one corner of the rectangle will be in $c^*ac^*bc^*$ or in $c^*ac^*ac^*$ which are both subsets of K. $\Box$

To prove a lower bound for monoids outside of **DA**, it is sufficient to look at BA_2 and U (again by Lemma 3.2.4). Also, since BA_2 divides $U \times U$, any lower bound for BA_2, will also hold for U. Both monoids have linear 2-party communication complexity. We establish the lower bound for BA_2.

Lemma 5.2.4 $C^{(2)}(BA_2) = \Omega(n)$.

Proof: Recall that BA_2 is the syntactic monoid of the language $(c^*ac^*bc^*)^*$. Here also, we use Lemma 4.1.4; Disjointness can be recognized by the following linear length program over BA_2.

For every pair x_i, y_i, we build a four letter string w. Let $w_1 = a$ if $x_i = 1$ $w_2 = a$ if $y_i = 1$ $w_3 = b$ if $x_i = 1$ $w_4 = b$ if $y_i = 1$. Any letter of w is c otherwise. This 2-input program outputs $aabb \equiv 0$ if both input bits are on (in which case the two sets are not disjoint) and outputs either 1 or the partial identity ab otherwise. One can now easily see that the concatenation of the w's corresponding to each pair x_i, y_i is a program over BA_2 recognizing $DISJ$, because it will output 0 iff $x_i = y_i = 1$ for some i. $\square$

Remark: As an alternative proof of this lower bound, it is easily seen that, when player 1 sees only the odd-positioned letters in the input, the set $F = \{(cc + ab)^{\frac{n}{2}}\}$ is an exponential size fooling set for the language $(c^*ac^*bc^*)^*$.

Summing up the bounds proved in Lemmas 5.2.1, 5.2.2, 5.2.3 and 5.2.4, we get the following characterization:

Theorem 5.2.5 *Let M be a finite aperiodic monoid,*

1. *$C^{(2)}(M) = \Theta(1)$ iff M is commutative.*

2. *$C^{(2)}(M) = \Theta(\log n)$ iff M is in* **DA** *and is non-commutative.*

3. *$C^{(2)}(M) = \Theta(n)$ iff M is not in* **DA**.

The case with $O(1)$ players

By the upper bound of Lemma 5.2.1, we know that any monoid in **DA** has $O(\log n)$ complexity for any number of players. However, we can do better and for any monoid $M \in$ **DA**, there is a k such that $C^{(k)}(M) = O(1)$.

Theorem 5.2.6 *Let $L = A_0^* a_1 A_1^* \dots A_{k-1}^* a_k A_k^*$ be an unambiguous concatenation, then $L \in CCC^{(k+2)}$.*

Proof: We exhibit a constant cost protocol for L.

Each player in turn verifies if the input he sees is in L. If so he outputs a 1 followed by the identities of the players who see each letter he identified as one of the bookmarks in his factorization. Otherwise, he outputs 0.

The protocol accepts the input if two players outputted a 1 and subsequently identified each other. We claim that this protocol is valid.

First, if the input w is in L, then w can be uniquely factorized as $w = w_0 a_1 w_1 \dots w_{k-1} a_k w_k$. Two players, at least, see the bookmarks $a_1, \dots, a_k$. The letters they ignore are taken from the $w_i' s$ so they both receive an input which is in L. Moreover, since the concatenation is unambiguous, they must choose these same bookmarks when they factorize their input. Therefore, the protocol will accept.

Conversely, if the protocol accepts, then two players, say P1 and P2, received inputs w^1, w^2 such that $w^1, w^2 \in L$.

Let w^0 be the word obtained from w by erasing letters unknown to either P1 or P2. Since the protocol accepted, w^0 contains the bookmarks of w^1. It can therefore be factorized using these bookmarks. Similarly, w^0 can be factorized using w^2's bookmarks. But by the uniqueness of the factorization, it follows that w^1 and w^2 have the same bookmarks.

Finally, since w^1 and w^2 are in L, and since P1 and P2 collectively know the whole input, w is in L.

The protocol is simultaneous and has cost at most $(k+2)\cdot(1+\log(k+2))$

$\square$

Although it is possible that all monoids of **DA** are in $CCC^{(3)}$, we believe that **DA** is not contained in any of the varieties induced by the communication complexity classes $CCC^{(k)}$. In contrast, the following example shows that there are cases where we need much less than $\#$ *of bookmarks* $+\ 2$ players to drop the communication complexity down to constant.

Example: For any integer t, the unambiguous concatenation $(c^*ac^*bc^*)^t$ can be recognized by four players using only constant communication using the following protocol.

Protocol: The players will go through the input sequentially, sidelining at each step the players that ignored the previous a or b. In the first step, the parties identify the player ignoring the first a: if each player communicates the identity of the player who has the first a on his forehead, then they will all be right, except the player who does ignore that first a. Say it is player

4. Then players 1, 2 and 3 can decide, in the same way, who ignores the first b. Say it is player 3.

Now players 1 and 2 can check if the a is before the b and if there are no a's or b's in between them, because each input letter is known to at least one of them. The crucial step in the protocol is that player 4, although he was sidelined temporarily, knows that the last b identified is the first b that player 3 ignores. So he can return to the game and help identify the next a after this b.

Every time we need to identify the next a or b, we will have three players available, which is precisely what we need. Similarly, whenever we are checking if some segment of the input if free of unwanted a's or b's, we will be able to rely on two players who know exactly where this segment begins and ends, and this is sufficient. □

Not all aperiodic monoids are in the variety defined by CCC:

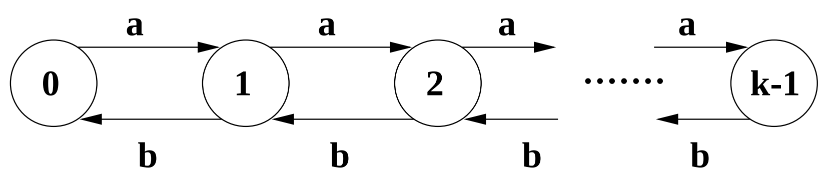
Theorem 5.2.7 *For any fixed k, $C^{(k)}(U) = \Omega(\log n)$*

Proof: It is shown in [32] that U is universal, that is any boolean function can be recognized by a program of length $O(c^n)$ over U, corresponding to the function's disjunctive normal form.

For any k, we know there exists a language L such that $C^{(k)}(L) = \Omega(n)$. So Lemma 4.1.4 completes the proof. $\square$

We introduce now a family of monoids which we will call BA_k. Recall that BA_2 is the syntactic monoid of the regular language $(c^*ac^*bc^*)^*$. The automaton recognizing this language can be seen as maintaining a stack of depth 1. For every a it scans, it pushes something on the stack, and for every b it pops the stack. The automaton accepts if and only if the stack is empty at the end of the computation. Also, any underflow or overflow of the stack causes the automaton to reject the input.

If we increase the depth of the stack, we get a family of regular languages, recognized by the following finite automaton:



Note that there are two undefined transitions in this figure, corresponding to underflow and overflow of the stack. Both of these crash the computation so they should lead to a sink state.

We denote by BA_k the syntactic monoid associated with the stack of depth $(k-1)$.

Theorem 5.2.8 *The k-party communication complexity of BA_k is* $\Omega(\log(n))$

Proof: We use a reduction from the universal U, but not by means of a program, as in most of our previous results. Given a word $v \in U^*$, the k players will build an input $w \in BA_k^*$ such that the value of any element of w unknown to player P_i depends only on elements of v unknown to P_i, and such that $eval_{BA_k}(w) = 0$ if and only if $eval_U(v) = 0$.

Note that $eval_U(v) = 0$ can only happen if some segment of v is of the form ac^*a. Suppose $|v| = n$, then w will be the concatenation of $\binom{n}{k}$ segments of length $2k$, which we denote by t_i, one for every subsegment of v. Every t_i has a similar form: for $1 \leq j \leq k$, the j^{th} letter of t_i will be a, and the $(j+k)^{th}$ letter will be b if the letters of the corresponding subsegment which are unknown to player P_j are compatible with the assumption that this segment is of the form ac^*a. Otherwise, both letters are c's. Note that the value of these two positions will be known to every party but P_i.

Disregarding the c's, each t_i will be of the form $a^m b^m$ for some $0 \leq m \leq k$. It is easily seen that this is a partial identity in BA_k whenever $m < k$ and 0 if $m = k$. So clearly, we could use a k-party protocol of cost $o(\log(n))$ for BA_k to build a $o(\log(n))$ protocol for U, a contradiction. □

This proof demonstrates how we can go beyond the simple reductions using programs to obtain lower bounds. Note also that the very same argument can show that the function $DISJ_k$ has logarithmic k-party communication complexity.

The identity letter issue

Example 5.1 exhibits a group language for which the absence of an identity letter reduces the communication complexity of a language from linear to constant. Similar examples exist also in the case of languages recognized by aperiodic monoids.

One trivial example is the language $(ab)^*$. Its syntactic monoid is BA_2, but it obviously has constant 2-party communication complexity, since for any n, $(ab)^* \cap \{a,b\}^n$ is a singleton.

Perhaps more interesting is the 3-party communication complexity of the language $L = A^*aA^*bA^*cA^*$. We suspect that when $A = \{a,b,c,d\}$, i.e. when we have an identity letter d, $C^{(3)}(L) = \Theta(\log n)$. In contrast, L has constant 3-party complexity if the input alphabet $A = \{a,b,c\}$. The protocol heavily relies on the absence of an identity letter.

Protocol We build the protocol in 2 steps. Let w be the input.

As a first step, the three players try to rule out certain easy cases. If one of them sees a subword abc, we are done and $w \in K$. This is checked using 3 bits.

If this is not the case, then they can decide, as in example 5.2, which two players see the first a and which two players see the last c: each player in turn communicates the identity of the player he thinks ignores the first a. Two of them will be right, and the player actually ignoring it will necessarily be wrong. If there are no a's or c's, we know that $w \notin K$. This requires 12 bits in all.

Without loss of generality, we can suppose that P_1 sees the first a but ignores the last c, P_2 sees the first a and the last c but sees no b in between them, and P_3 sees the last c but ignores the first a. If P_1 sees no b after the first a, or if P_3 sees no b before the last c, then they know that $w \notin K$ and they can communicate this.

After these few bits of communication, the protocol has stopped unless we have the following:

P_1 sees an input of the form $(b+c)^*a(a+c)^*b_1(a+b)^*$

P_2 sees an input of the form $(b+c)^*a(a+c)^*c(a+b)^*$

P_3 sees an input of the form $(b+c)^*b_2(a+c)^*c(a+b)^*$

where b_1 and b_2 are two possibly distinct occurrences of the letter b. Note that $w \in K$ iff b_1 "comes before" b_2. This is what the second step of our protocol determines. We assume that the protocol did not terminate in step 1. Hence, if w is in K, it can be factored as

$$w = (b+c)^* a A^* b_1 A^* b_2 A^* c (a+b)^*$$

Between a and b_1, every bit ignored by P_1 has to be either an a or a c because P_2 sees it. But it can not be an a since P_3 sees it. We can put several such restrictions on the input seen by each player which we can summarize in the following table.

w		a		b_1		b_2		c	
P_1	$(b+c)^*$	a	P_2: c P_3: a or c	b_1	P_2: b P_3: b	b_2	P_2: a P_3: a		$(a+c)^*$
P_2	$(b+c)^*$	a	P_1: c P_3: a or c		P_1: c P_3: b		P_1: a or c P_3: a	c	$(a+c)^*$
P_3	$(b+c)^*$		P_1: c P_2: c	b_1	P_1: c P_2: b	b_2	P_1: a or c P_2: a	c	$(a+c)^*$

By P_3: a or c, we mean that the letters ignored by P_3 in this segment of the input can only be a's or c's. Each strip in the table describes the type of input that each player will receive. So if w is in K, each player sees an input of the above format. Now suppose that w is not in K, but that all players still see inputs of this form. Because the protocol did not end in step 1, w

can only be of the form $A^*b_2A^*aA^*cA^*b_1A^*$ The table above shows that every letter ignored by P_2 between b_2 and the last c must be an a. On the other hand, every bit ignored by P_2 between a and b_1 must be a c, a contradiction.

So P_1, P_2 and P_3 can factorize the input they see in the way described in the table iff w is in K. They can check this using 3 bits.

The proof breaks down if one of the intervals $a \dots b_1$ or $b_2 \dots c$ is free of bits unknown to P_2. However this implies that w is in K, and so we can avoid the problem by using 2 more bits in step 1, and the protocol is complete.

Note that the rigid structure of words in $A^*aA^*bA^*cA^*$ that we exploit here is due to the absence of an identity letter.

Chapter 6

Conclusion

This work has established links between the algebraic structure of finite monoids and the communication complexity of languages they recognize. We conclude by presenting a conjecture that would make this relation stronger. We also discuss possibilities for further research in this line of work.

6.1 Conjecture

One very interesting challenge would be to prove a result similar to Szegedy's theorem for multiparty communication complexity. An algebraic characterization of languages of bounded k-party communication complexity might be too much to ask, but this might be easier, if we consider the communication complexity class CCC, the class of languages having constant communication complexity for some constant number of players.

Conjecture 6.1.1 *A language L has constant communication complexity for some constant number of players iff L can be recognized by a polynomial length program over a finite monoid M of constant communication complexity for some constant number of players.*

We know that this class of monoids forms a P-variety, and from the results of the previous chapter, we know that it contains the variety $\mathbf{DA} \vee \mathbf{G_{nil}}$, the smallest variety containing both $\mathbf{DA}$ and the nilpotent groups.

The communication complexity of BA_2 with $O(1)$ players remains an open question. We conjecture it to be $\Theta(n)$ for any constant number of players. This would show that any aperiodic monoid outside of $\mathbf{DA}$ has non-constant communication complexity for any constant number of players, which in turn would show that the monoid variety corresponding to the communication complexity class CCC is exactly $\mathbf{DA} \vee \mathbf{G_{nil}}$. We thus further conjecture:

Conjecture 6.1.2 *A language L has constant communication complexity for some constant number of players iff L can be recognized by a polynomial length program over a finite monoid $M \in \mathbf{DA} \vee \mathbf{G_{nil}}$.*

There is combinatorial intuition supporting this conjecture. This variety is the largest one for which every language can be characterized by the order of appearance and number (when counting is done with respect to a congruence of finite index) of subwords of some bounded length. Of course, cheap access to subwords of length $(k-1)$ is one of the main features of the k-player game.

6.2 Possibilities for future work

Aside from this conjecture, there are many interesting questions raised by the algebraic point of view on communication complexity and its applications to other complexity issues.

First, notice that most of the basic results of chapter 4, which are important tools in this work, hold for other models of communication complexity. In particular, Theorems 4.1.1 and 4.1.7, as well as Lemma 4.1.4 hold for probabilistic, simultaneous and rounds-limited communication complexity. (They do not hold however in the non-deterministic or distributed models.) It could be interesting to investigate the algebraic properties of varieties defined by communication complexity classes in these models. Our results for groups hold in the probabilistic setting since the GIP lower bound holds in that model. This is also the case for the linear lower bound on the 2-party communication complexity of BA_2. There are, however, monoids in **DA** which require only $\log\log n$ 2-party randomized communication complexity. For instance, the membership question for the regular languages c^*aA^* and $c^*ac^*bc^*$ can be reduced to the comparison of two $\log n$ bit numbers, and a protocol of Nisan and Safra presented in [18] shows that this requires only $\log\log n$ bits of communication in the randomized model.

It also seems worthwhile to look at the case of a non-constant number of players. This is probably a difficult task, as a non-constant lower bound on the communication complexity of a specific function for $O(\log^2(n))$ players has yet to be proved. As we mentioned earlier, a result of Yao implies that any ACC^0 function has polylogarithmic simultaneous communication complexity for a polylogarithmic number of players. From the algebraic point of view, this implies that any solvable monoid has polylogarithmic simultaneous communication complexity for a polylogarithmic number of players.

Bibliography

[1] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory. In *Proc. 27th IEEE FOCS*, pages 337–347, 1986.

[2] L. Babai, P. G. Kimmel, and S. V. Lokam. Simultaneous messages vs. communication. In *12th Annual Symposium on Theoretical Aspects of Computer Science*, pages 361–372, 1995.

[3] 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.

[4] 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.

[5] 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.

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

[7] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. In *Proc. 26th IEEE FOCS*, pages 429–442, 1985.

[8] D. Dolev and T. Feder. Multiparty communication complexity. In *Proc. 30th IEEE FOCS*, pages 428–433, 30 Oct.–1 Nov. 1989.

[9] P. Duris and J. D. P. Rolim. Optimal lower bounds on the multiparty communication complexity. In *12th Annual Symposium on Theoretical Aspects of Computer Science*, pages 350–360, 1995.

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

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

[12] V. Grolmusz. On multi-party communication complexity of random functions. Technical Report MPII-1993-162, Max Planck Institut für Informatik, Dec. 1993.

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

[14] V. Grolmusz. Circuits and multi-party protocols. *Computational Complexity*, 1998.

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

[16] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

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

[18] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[19] L. Lovász. Communication complexity: a survey. Technical Report CS-TR-204-89, Princeton University, 1989.

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

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

[22] I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 31 July 1991.

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

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

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

[26] P. Pudlák, V. Rödl, and J. Sgall. Boolean circuits, tensor ranks, and communication complexity. *SIAM J. Comput.*, 26(3):605–633, June 1997.

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

[28] A. A. Razborov. On the distributed complexity of disjointness. *Theoretical Comput. Sci.*, 106(2):385–390, 14 Dec. 1992. Note.

[29] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, Apr. 1965.

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

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

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

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

[34] A. C. Yao. Lower bounds by probabilistic arguments. In *Proc. 24th IEEE FOCS*, pages 420–428, 1983.