Communication Complexity

Eyal Kushilevitz*
Department of Computer Science
Technion
Haifa 32000, Israel

Abstract

In this chapter we survey the theory of two-party communication complexity. This field of theoretical computer science aims at studying the following, seemingly very simple, scenario: There are two players Alice who holds an n-bit string x and Bob who holds an n-bit string y. Their goal is to communicate in order to compute the value of some boolean function $f(x, y)$, while exchanging a number of bits which is as small as possible.

In the first part of this survey we present, mainly by giving examples, some of the results (and techniques) developed as part of this theory. We put an emphasis on proving lower bounds on the amount of communication that must be exchanged in the above scenario for certain functions f. In the second part of this survey we will exemplify the wide applicability of the results proved in the first part to other areas of computer science. While it is obvious that there are many applications of the results to problems in which communication is involved (e.g., in distributed systems), we concentrate on applications in which communication does not appear explicitly in the statement of the problems. In particular, we present results regarding the following models of computation:

- Finite automata
- Turing machines
- Decision trees
- Ordered binary decision diagrams (OBDDs)
- VLSI chips
- Networks of threshold gates

We provide references to many other issues and applications of communication complexity which are not discussed in this survey.

*e-mail: eyalk@cs.technion.ac.il; http://www.cs.technion.ac.il/~eyalk

1 Introduction

Communication complexity aims at studying the amount of communication bits that the participants of a communication system need to exchange in order to perform certain tasks. [Yao 1979] defined a very simple model for studying this type of questions. In this model there are only two parties (Alice and Bob) and their task is to evaluate a function $f(x, y)$, where x is Alice's input and y is Bob's input. It turns out that this very simple model already captures many of the fundamental issues related to the complexity of communication and that results proven in this model can be often extended to more complicated scenarios.

While in a first look it may seem that the field of communication complexity is related only to questions in networks, distributed computing and related areas, we will show that in fact this field has much wider applications. This was first identified by [Thompson 1979] who discovered the applications of communication complexity to VLSI (in particular, to Area-Time trade-offs for VLSI chips). Later, applications of communication complexity to many other fields were found. These in particular include many computation models such as Turing machines, finite automata, ordered binary decision diagrams (OBDDs) and others.
Research in communication complexity goes in several directions:

- exploring the properties of the two-party model,
- extending the model in various ways, and
- finding applications of communication complexity to other fields.

In this chapter we try to give some highlights of the field of communication complexity. We will not present the results in their full generality. Rather, we will concentrate on some basic results which are later used in the applications. Our intention is mainly to give the flavor of the type of results that can be proved using the theory of communication complexity. The interested reader who wants to extend his knowledge communication complexity (and in particular to complete some of the missing proofs and facts) can use the bibliographic list to extend his background. In addition, several surveys that cover various aspects of communication complexity appeared in [Orlitsky and El-Gamal 1988, Lovász 1989, Lengauer 1990]. A comprehensive introduction to the theory of communication complexity (and its application) which covers all the material of the current chapter and much more appears in the forthcoming book of [Kushilevitz and Nisan 1996].

Organization: In Section 2 we deal with the *theory* of communication complexity. We introduce the basic two-party model in Section 2.1 and present some lower bound techniques in Section 2.2. In Section 2.3 we extend the basic two-party model by allowing randomized protocols, while in Section 2.4 we consider another extension of the basic model which allows

variable partition of the input among the two players. We briefly mention some related topics in Section 2.5.
In Section 3 we consider various *applications* of the results of Section 2. These applications deal with finite automata (Section 3.1), Turing machines (Section 3.2), decision trees (Section 3.3), ordered binary decision diagrams (OBDDs) (Section 3.4), VLSI chips (Section 3.5), and threshold circuits (Section 3.6). We briefly mention some other applications in Section 3.7.

2 Two-Party Communication Complexity – Theory

2.1 The Model

In this section we describe the two-party communication complexity model, as defined by [Yao 1979]. One of the most important features of this model is its *simplicity.* It considers a situation where there are only two communicating parties; it concentrates on very simple tasks – computing two argument (boolean) functions where one argument is known to one party and the other argument is known to the other party; it completely ignores the computational resources needed by the parties; and, it focuses solely on the amount of communication exchanged between the parties. This simplicity of the model allows us to get a quite good understanding of the model. On the other hand, it turns out that the model is still meaningful enough to allow using the results proved in this model (mainly lower bounds) in other contexts (this will be done in Section 3) and rich enough to exhibit a beautiful mathematical theory.

Consider a two-argument, boolean function $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$. Let Alice and Bob be the two communicating parties. Alice is given an input $x \in \{0,1\}^n$ and Bob is given an input $y \in \{0,1\}^n$. They wish to compute the value of $f(x,y)$. Some examples of functions in which we will be interested include:

equality $EQ(x,y)$ is defined to be 1 if $x = y$ (and 0 otherwise).

inner-product $IP(x,y)$ is defined as $\sum_{i=1}^{n}(x_i \cdot y_i) \bmod 2$, where $x_1, \ldots, x_n$ are the bits of x and $y_1, \ldots, y_n$ are the bits of y.

greater-than $GT(x,y)$ is defined to be 1 if $x > y$ (and 0 otherwise), where x and y are viewed as the binary representation of numbers in the range $0, 1, \ldots, 2^n - 1$.

disjointness $DISJ(x,y)$ is defined to be 1 if there is no index i such that $x_i = y_i = 1$ (and $DISJ(x,y) = 0$ if such an index exists). (We think of x and y as subsets of $\{1, \ldots, n\}$ represented by their characteristic vectors. In this case $DISJ(x,y) = 1$ if and only if these two subsets are disjoint, i.e., $x \cap y = \emptyset$.)

Again, the functions we concentrate on are simple enough so that we can get a full understanding of what can be done with them. On the other hand, they are very "natural" functions. For example, the equality function represent a computation which is often done in a distributed environment which is checking if files (strings) are consistent (that is, equal).

The computation of the value $f(x,y)$ is done using a *communication protocol.* During the execution of the protocol, the two parties alternate roles in sending messages. Each of these messages is a string of bits. The protocol, based on the communication so far, specifies whether the execution terminated (in which case it also specifies what is the output). If the execution has not terminated, the protocol specifies what message the sender (Alice or Bob) should send next, as a function of its input and of the communication so far. A communication protocol $\mathcal{P}$ computes the function f, if for every input pair $(x,y) \in \{0,1\}^n \times \{0,1\}^n$ the protocol terminates with the value $f(x,y)$ as its output.

Every function f can be computed by the following trivial protocol: Alice sends her input x to Bob (n bits of communication), and then Bob who now knows both x and y computes the bit $f(x,y)$ and sends it to Alice. However, we think of n as a large number and hence sending the whole input is very expensive. Indeed, in many cases (as we will see), there are much more efficient protocols. The study of communication complexity aims at identifying the minimal number of bits that must be sent in order to compute a function f. More precisely, the complexity measure for a protocol $\mathcal{P}$ is the worst case number of bits exchanged by the two parties. Formally, let $s_{\mathcal{P}}(x,y) = (m_1, m_2, \ldots, m_r)$ be the communication exchanged on input (x,y) during the execution of $\mathcal{P}$, where m_i denotes the i-th message sent in the protocol. Also denote by $|m_i|$ the length (number of bits) of m_i and let $|s_{\mathcal{P}}(x,y)| = \sum_{i=1}^{r} |m_i|$. We now define the *(deterministic) communication complexity* of $\mathcal{P}$ as the worst case number of bits exchanged by the protocol. That is,

$$D(\mathcal{P}) \triangleq \max_{(x,y)\in\{0,1\}^n\times\{0,1\}^n} |s_{\mathcal{P}}(x,y)|.$$

The *(deterministic) communication complexity* of a function f is the communication complexity of the best protocol that computes f. That is:

$$D(f) \triangleq \min_{\mathcal{P}:\mathcal{P} \text{ computes } f} D(\mathcal{P}).$$

By using the trivial protocol described above we get,

$$\forall f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\} \qquad D(f) \le n+1.$$

2.2 Proving Lower Bounds

Our main concern will be proving lower bounds on the communication complexity of given functions. These lower bounds will later be used for proving lower bounds in other models of computation as well. Before doing so, we emphasize that the task of proving *lower bounds* is usually much more difficult than the task of proving *upper bounds*. The later task usually involves the analysis of some particular solution to a problem, while the former task involves a proof that *any* solution to a given problem (a solution that we can assume nothing about it other than the fact that it solves the problem) has at least a certain complexity. The main motivation for proving lower bounds is to know the limits of our upper bounds. For example, how do we know whether a protocol that we have for a certain communication complexity problem is efficient or if we should look for a better protocol. If the upper bound given by the protocol meets the lower bound then we obviously have the best possible solution. In computational complexity, for example, almost no good lower bounds are known. The most famous case where lower bounds seem to be hard to prove is the question "Is $P \neq NP$?". That is, can we prove a super-polynomial lower bound for the time required to solve the satisfiability problem. As mentioned, the model of communication complexity is relatively simple and this allows, in many cases, proving good lower bounds (which can also be applied in other domains, as shown in Section 3). For proving communication complexity lower bounds, we analyze the combinatorial structure imposed by protocols. The basic combinatorial element in this analysis is called a *rectangle*:

Definition 1: *A* rectangle *is a subset of $\{0,1\}^n \times \{0,1\}^n$ of the form $A \times B$, where each of A and B is a subset of $\{0,1\}^n$. A rectangle $R = A \times B$ is called f-*monochromatic *if for every $x \in A$ and $y \in B$ the value of $f(x,y)$ is the same.*

Pictorially, we can think of the space of inputs $\{0,1\}^n \times \{0,1\}^n$ as a matrix. In this case A is just a subset of the rows, B a subset of the columns and $R = A \times B$ is a rectangle. This is illustrated in Figure 1. Note however that, although it is convenient to draw *geometric* rectangles, the definition allows A and B to be arbitrary subsets of $\{0,1\}^n$ (and not necessarily subsets that correspond to adjacent rows or adjacent columns). For an additional example, consider the equality function EQ. Let A be the set of all strings in $\{0,1\}^n$ whose first bit is 1 and let B be the set of all strings in $\{0,1\}^n$ whose first bit is 0. Then, $A \times B$ is a rectangle which is EQ-monochromatic: for every $x \in A$ and $y \in B$ we have $x \neq y$ and therefore $EQ(x,y) = 0$.

The following lemma claims that the inputs for which the communication is the same form an f-monochromatic rectangle.

Lemma 1: *Let $\mathcal{P}$ be a protocol that computes a function f and $(m_1, \ldots, m_r)$ be a sequence of messages. The set of inputs (x,y) for which $s_{\mathcal{P}}(x,y) = (m_1, \ldots, m_r)$ is an f-monochromatic rectangle.*

Proof: First, we prove by induction on i that the set of inputs on which the communication starts with $(m_1, \ldots, m_i)$ is a rectangle. For $i = 0$ this set is $\{0,1\}^n \times \{0,1\}^n$ which is certainly a rectangle. Let R be the set of inputs for which the communication starts with $(m_1, \ldots, m_i)$. By the induction

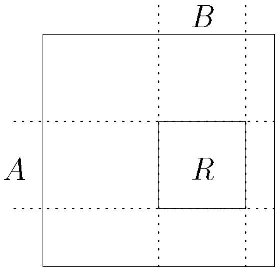

Figure 1: A rectangle

hypothesis R forms a rectangle $A \times B$. Suppose, without loss of generality, that the message m_{i+1} is sent by Alice. Let A' be the set of inputs $x \in A$ on which, given $m_1, \ldots, m_i$, the message sent by Alice is m_{i+1}. Then, the set of inputs consistent with $(m_1, \ldots, m_{i+1})$ is $A' \times B$ which is a rectangle. (The case where Bob sends the message m_{i+1} is handled similarly, where this time the message m_{i+1} determines a set $B' \subseteq B$.) Therefore, the set of inputs (x, y) for which $s_{\mathcal{P}}(x, y) = (m_1, \ldots, m_r)$ is a rectangle. Finally, since the output is determined solely by the communication, then this rectangle is f-monochromatic. $\square$

For a function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we define $C^P(f)$ as the minimum number of f-monochromatic rectangles that partition the space of inputs, $\{0,1\}^n \times \{0,1\}^n$.

Lemma 2*: For every function $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$,*

$$D(f) \geq \log_2 C^P(f).$$

Proof: By Lemma 1, every protocol $\mathcal{P}$ partitions the space of inputs, $\{0,1\}^n \times \{0,1\}^n$, into f-monochromatic rectangles. The number of these rectangles (i.e., the number of possible communications) is at most $2^{D(\mathcal{P})}$ and $D(f) \leq D(\mathcal{P})$, hence $C^P(f) \leq 2^{D(f)}$. The lemma follows. $\square$

Lemma 2 implies that for proving lower bounds on the communication complexity of f, we can prove lower bounds on the number of f-monochromatic rectangles that partition the space of inputs. In what follows we present two methods for proving lower bounds on $D(f)$. Both methods go through proving lower bounds on $C^P(f)$. The first of these methods is called the *fooling set* method. It was implicitly used in [Yao 1979] and was made more explicit in [Lipton and Sedgewick 1981]. The fooling set method is based on exhibiting a large set of inputs such that each of them must be in a different f-monochromatic rectangle. Formally,

Definition 2*: A set of input pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_\ell, y_\ell)\}$ is called a* fooling set *(of size ℓ) with respect to f if there exists $b \in \{0,1\}$ such that*

1. For all i, $f(x_i, y_i) = b$.

2. For all $i \neq j$, either $f(x_i, y_j) \neq b$ or $f(x_j, y_i) \neq b$.

Lemma 3: *If there exists a fooling set of size ℓ with respect to f then*

$$D(f) \geq \log_2 \ell.$$

Proof: By Lemma 2, it suffices to prove that $C^P(f) \geq \ell$. For this, we prove that in any partition of $\{0,1\}^n \times \{0,1\}^n$ into f-monochromatic rectangles the number of rectangles is at least ℓ. Suppose that the number of f-monochromatic rectangles is smaller than ℓ. In this case, there exist two pairs in the fooling set (x_i, y_i) and (x_j, y_j) that belong to the same rectangle $A \times B$. This implies that $x_i, x_j \in A$ and $y_i, y_j \in B$ which means that (x_i, y_j) and (x_j, y_i) also belong to the rectangle $A \times B$. However, by the definition of fooling set, $f(x_i, y_i) = f(x_j, y_j) = b$ while at least one of $f(x_i, y_j)$ and $f(x_j, y_i)$ is different than b. This implies that the rectangle $A \times B$ is not f-monochromatic. $\square$

Corollary 4:

- $D(EQ) \geq n$.
- $D(GT) \geq n$.
- $D(DISJ) \geq n$.

Proof: We first prove the claim for the equality function, EQ. The proof for the function GT is almost identical.

By Lemma 3, it suffices to present a fooling set of size 2^n for the function EQ. The fooling set will be:

$$\{(\alpha, \alpha) : \alpha \in \{0,1\}^n\}.$$

This is clearly a set of size 2^n. Also $EQ(\alpha, \alpha) = 1$ for every α, while $EQ(\alpha, \alpha') = EQ(\alpha', \alpha) = 0$ for every $\alpha \neq \alpha'$.

For the function $DISJ$, we observe that the following is a fooling set of size 2^n for this function:

$$\left\{(A, \overline{A}) : A \subseteq \{1, \ldots, n\}\right\}.$$

$\square$

The second method for proving lower bounds on $D(f)$ has an algebraic nature, and hence it allows applying known results from linear algebra. For this, we associate with every function f a $2^n \times 2^n$ zero-one matrix denoted M_f. Each row of the matrix M_f is associated with a string $x \in \{0,1\}^n$ and each column of the matrix M_f is associated with a string $y \in \{0,1\}^n$. The (x, y) entry of the matrix M_f contains the value $f(x, y)$. The following lemma, due to [Mehlhorn and Schmidt 1982], relates the communication complexity of f to the rank of the matrix M_f (over any field):

Lemma 5*: Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ be a function. Then,*

$$D(f) \geq \log rank(M_f).$$

Proof: By Lemma 2, it suffices to prove that $C^P(f) \geq rank(M_f)$. Given an optimal cover of $\{0,1\}^n \times \{0,1\}^n$ with f-monochromatic rectangles, let $R_1, \ldots, R_t$ be those rectangles which are 1-monochromatic. It is enough to prove that $t \geq rank(M_f)$. For each of the rectangles R_i $(1 \leq i \leq t)$, define a $2^n \times 2^n$ matrix M_i whose (x,y) entry is 1 if $(x,y) \in R_i$ and 0 otherwise. It follows that

$$M_f = \sum_{i=1}^{t} M_i.$$

Since the rank is sub-additive, we get

$$rank(M_f) \leq \sum_{i=1}^{t} rank(M_i).$$

Since R_i is a 1-monochromatic rectangle then for every i, $rank(M_i) = 1$. Hence, $rank(M_f) \leq t$, as desired. □

Corollary 6*:*

- *$D(EQ) \geq n$.*
- *$D(IP) \geq n$.*

Proof: The matrix corresponding to the equality function EQ is the $2^n \times 2^n$ identity matrix (a matrix with 1 in each entry of the main diagonal and 0 elsewhere). The rank of this matrix is 2^n.

The matrix corresponding to the inner-product function IP, is a so-called Hadamard matrix whose rank is known to be $2^n - 1$. □

2.3 Randomized Communication Complexity

In this section we wish to strengthen the two-party model described above by allowing the two parties to "toss coins" (that is, to make randomized choices) in order to decide which messages they send. The focus in this section will be on proving *upper bounds*. The goal here is to show, for specific functions f, randomized protocols whose cost is significantly smaller than the lower bounds proved for deterministic protocols.

Before presenting some randomized protocols, few remarks about definitions are in place. First, we allow Alice and Bob to make random choices. That is, each message sent by Alice or Bob is a *probabilistic* function of the sender's input and of the communication so far. In addition, we allow the protocols to make errors. More specifically, we say that a protocol $\mathcal{P}$ computes the function f if for *every* input (x,y) the probability that the output of $\mathcal{P}$ on (x, y) equals $f(x, y)$ is at least $3/4$. The *(randomized) communication complexity* of $\mathcal{P}$, denoted $R(\mathcal{P})$, is the worst case number of bits exchanged (over all inputs (x, y) and over all random choices). Finally, the *(randomized) communication complexity* of f is defined by

$$R(f) \triangleq \min_{\mathcal{P}:\mathcal{P} \text{ computes } f} R(\mathcal{P}).$$

We emphasize that we require the protocol to succeed for *every* input with high probability and not, for example, to succeed on *most* inputs. In other words, the only probability distribution we have in this model is on the random choices made by Alice and Bob and there is no notion of distribution over the inputs.

The first example shows that the equality function can be computed using $O(\log n)$ bits. This should be contrasted with the lower bound of n proved for the deterministic communication complexity of this function (Corollary 4 and Corollary 6).

Lemma 7*:*

$$R(EQ) = O(\log n).$$

Proof: Denote the input of Alice by $a = a_0 a_1 \dots a_{n-1}$, and the input of Bob by $b = b_0 b_1 \dots b_{n-1}$. To compute $EQ(a, b)$, we think of these inputs as two polynomials over the field $GF[p]$ (the field consisting of the numbers $0, 1, \dots, p-1$ with the operations of addition and multiplication modulo p) where p is a prime such that $4n^3 < p < 8n^3$ (theorems regarding the density of primes guarantee the existence of such p). That is,

$$A(x) = a_0 + a_1 x + \dots + a_{n-1} x^{n-1} \pmod{p}$$

and

$$B(x) = b_0 + b_1 x + \dots + b_{n-1} x^{n-1} \pmod{p}.$$

Alice picks uniformly at random a number t in $GF[p]$ and sends Bob the values t and $A(t)$. Bob outputs 1 if $A(t) = B(t)$ and 0 otherwise. The number of bits exchanged is $2 \cdot \log p = O(\log n)$. For the correctness, note first that if $a = b$ then $A(t) = B(t)$ for all t, so the output is always 1. If $a \neq b$ we have two distinct polynomials A and B of degree $n-1$. Such polynomials can be equal on at most $n-1$ (out of p) elements of the field (since their difference is a non-zero polynomial of degree at most $n-1$ who may have at most $n-1$ roots). Hence the probability of error is at most

$$\frac{n}{p} \leq \frac{n}{4n^3} = \frac{1}{4n^2} \leq \frac{1}{4},$$

as needed. $\square$

An additional example of a function for which we have a randomized protocol whose communication complexity is much better than the deterministic lower bound is the "greater than" function. By Corollary 4 we have $D(GT) \geq n$ while in the randomized case:

Lemma 8:

$$R(GT) = O(\log^2 n).$$

Proof: The protocol can be viewed as a binary search for the first index i in which x and y differ (if such an index exists). For this, Alice and Bob maintain two variables: L, the left border of search (initially 1), and R, the right border of search (initially n). In each step Alice and Bob both compute $M = \lfloor (L+R)/2 \rfloor$ and make the equality test

$$EQ(x_L \dots x_M, y_L \dots y_M)$$

by using the protocol of Lemma 7. If the protocol outputs 0 (that is, the strings are different) Alice and Bob set $R = M$; if the protocol outputs 1 (that is, the strings are equal) they set $L = M + 1$. If $L = R$ they exchange the bits x_L and y_L and decide which is larger (if the two bits are equal they assume that the numbers are equal).

If the values computed by the protocol for EQ are all correct then the correctness of the protocol is obvious. The probability of error is therefore bounded by the number of times that the protocol for EQ is used (that is, $\log n$) times the probability of error in each of them, which is $1/4n^2$. All together the error probability is at most $(\log n)/(4n^2) \leq 1/4n$. $\square$

In fact, it is known that the upper bound for the greater-than function can be improved to $R(GT) = O(\log n)$. For many other functions, lower bounds can be proven to show that randomized protocols are not more efficient than deterministic protocols. Methods for proving lower bounds for randomized protocols are typically much more complicated than their deterministic counterparts. Therefore, we only quote the following two lower bounds:

Lemma 9: ***[Chor and Goldreich 1985]***

$$R(IP) = \Omega(n).$$

Lemma 10: ***[Kalyanasundaram and Schnitger 1987, Razborov 1990]***

$$R(DISJ) = \Omega(n).$$

2.4 Variable Partition Model

Consider a (single argument) function $f : \{0,1\}^m \to \{0,1\}$, where $m = 2n$. The two party model defined above can be viewed as computing the value $f(z)$ in the case where Alice's input is the first n bits of z and Bob's input is the last n bits of z.

In the variable partition model, the two parties are allowed to choose the partition of the m input bits among them (n bits for each party). This partition is chosen based on f (independently of the specific input). For example, by Corollary 4, the function $EQ(x_1, \dots, x_n, y_1, \dots, y_n)$ requires n

bits of communication when the partition is fixed (that is, $x_1, \ldots, x_n$ to Alice and $y_1, \ldots, y_n$ to Bob). However, if the parties are allowed to choose the partition, we can give

$$x_1, \ldots, x_{\frac{n}{2}}, y_1, \ldots, y_{\frac{n}{2}}$$

to Alice and

$$x_{\frac{n}{2}+1}, \ldots, x_n, y_{\frac{n}{2}+1}, \ldots, y_n$$

to Bob. With this partition the function EQ can be computed using only 2 bits. We define $D^{best}(f)$, as the (deterministic) communication complexity of the best protocol for computing the function f with respect to the best partition of the m input bits into two sets. Obviously,

$$D^{best}(f) \leq D(f),$$

and as seen by the example of the function EQ the gap between $D^{best}(f)$ and $D(f)$ can be very large. Proving lower bounds on $D^{best}(f)$ is therefore more difficult.

For $x, y \in \{0,1\}^k$ and $0 \leq i \leq k-1$, define the "shifted equality" function $SEQ(x, y, i)$ to be 1 iff the string $x = x_0 x_1 \ldots x_{k-1}$ equals to the string y shifted circularly by i-bits to the right, that is to $y_i y_{i+1} \ldots y_{k-1} y_0 \ldots y_{i-1}$. In other words, $SEQ(x, y, i) = 1$ iff for all $0 \leq j < k$, $x_j = y_{i+j \bmod k}$. In what follows we will show a lower bound for $D^{best}(SEQ)$. The proof is by a reduction to the lower bound for EQ in the standard model.

Lemma 11*:*

$$D^{best}(SEQ) = \Omega(m).$$

Proof: As is the case with the function EQ above, for certain partitions checking equality may be easy. The idea will be to show that for some values of i the bits are partitioned between the players in a way that makes the equality-test "hard". First, observe that each of the two players holds a "significant" number of bits from a different string. To see this, note that each player gets $\frac{m}{2} = k + \frac{1}{2}\log k$ bits, out of them at least $k - \frac{1}{2}\log k$ are bits of x or y. Without loss of generality, Alice holds $\frac{k}{2}$ bits of x, and hence Bob holds at least $d = \frac{k}{2} - \frac{1}{2}\log k$ bits of y. Let $A \subseteq \{0, \ldots, k-1\}$ be d indices of bits of $x_0 x_1 \ldots x_{k-1}$ held by Alice, and $B \subseteq \{0, \ldots, k-1\}$ be d indices of bits of $y_0 y_1 \ldots y_{k-1}$ held by Bob (each player possibly holds other bits as well from x, y, and i).

Consider the special case where all bits of x and y not in A and B (respectively) are 0's (as we are proving a lower bound, we are allowed to restrict the input). We are now going to fix the value of i in a way that yields high communication complexity between Alice and Bob. For this, let us first see what is the communication complexity for some *fixed* value of i. Denote

$$B_i = \{j \mid (i + j \bmod k) \in B\}.$$

We claim that for any fixed i, the communication complexity between A and B is at least $|A \cap B_i|$. To see this, we further restrict the input by letting $x_j = 0$ and $y_{i+j \bmod k} = 0$, for all $j \notin (A \cap B_i)$. Now, observe that the induced function is 1 iff for all $j \in (A \cap B_i)$, it holds that $x_j = y_{i+j \bmod k}$. Since $j \in A$ the bit x_j is held by Alice, and because $j \in B_i$, we have $(i + j \bmod k) \in B$, that is, the bit $y_{i+j \bmod k}$ is held by Bob (and none of these bits was already fixed). By the lower bound for EQ (Corollary 4) applied to strings of length $|A \cap B_i|$, the claim follows (that is, a better protocol for this problem implies a protocol for EQ whose communication complexity is better than the lower bound). Finally, it remains to show that for some i, $|A \cap B_i| = \Omega(m)$. For this, we write

$$\sum_i |A \cap B_i| = \sum_{j \in A} |\{i | j \in B_i\}|$$

(to see this, think of a matrix whose rows are indexed by $j \in A$ and columns are indexed by the sets B_i; the entry (j, B_i) is 1 if $j \in B_i$ and 0 otherwise. With this view, the left hand term of the equality counts the 1's of this matrix column-by-column while the right hand term counts the 1's row-by-row). Now,

$$\sum_{j \in A} |\{i | j \in B_i\}| = |A| \cdot |B| = d^2.$$

Hence, for some i we have $|A \cap B_i| \geq d^2/k = \Omega(k) = \Omega(m)$, as needed. $\square$

As the reader notices, proofs that need to give lower bounds for all possible partitions of the inputs are usually more complicated. However, as a result, lower bounds in the variable partition model can be applied to wider variety of problems.

2.5 Other Issues

The goal of this section is to briefly mention some additional topics related to the two-party model. For each of these topics we give a short description and pointers to the relevant literature.

Nondeterministic Protocols – [Lipton and Sedgewick 1981] defined the notions of *nondeterministic protocols* and *nondeterministic communication complexity.* Nondeterministic protocols allow the two parties to choose the messages that they send in a nondeterministic way. Essentially, it is only required that there is an execution of the protocol that leads to the correct answer (but in oppose to the randomized case, there is no requirement on the probability of such event). The combinatorial objects corresponding to such protocols are *overlapping rectangles.* A fundamental result due to [Aho et al. 1983] (which was later strengthened by [Halstenberg and Reischuk 1988]) relates the deterministic communication complexity of f to its nondeterministic communication complexity. It allows obtaining several results regarding deterministic protocols using the notion of nondeterministic protocols.

Rounds – In the definition of communication complexity we are interested only in the number of bits exchanged between Alice and Bob and we ignore the number of *rounds* (that is, the number of messages exchanged). We may ask how many rounds or how much *interaction* is really necessary to obtain a low communication protocol. In a sequence of results [Papadimitriou and Sipser 1982, Ďuriš et al. 1984, Nisan and Wigderson 1991] it was proved that for every $k \geq 1$ there are functions such that computing them with a k-round protocol is exponentially more expensive than computing the same function with a $k+1$-round protocol.

Amortized Communication Complexity – An interesting question regarding communication complexity is whether computing a function f on ℓ instances can be done more efficiently than just computing f on each of the ℓ instances separately. For example, suppose we want to compare ℓ files (strings of length n) $x^1, x^2, \ldots, x^\ell$ which are stored in some site in a network with ℓ files $y^1, y^2, \ldots, y^\ell$ which are stored in another site (the output is ℓ bits $b^1, \ldots, b^\ell$ where b^i indicates whether the strings x^i and y^i are equal). Can this be done more efficiently than ℓn bits which is the cost of computing the equality function ℓ times (on each pair (x^i, y^i)). Both positive and negative results on amortized communication complexity appear in [Feder et al. 1991, Karchmer et al. 1991, Karchmer et al. 1992].

The power of lower bound techniques – A lot of research was devoted to understanding the power of the lower bound techniques used in the area of communication complexity. In particular, examples of functions for which the rank method (Lemma 5) does not give tight lower bounds were given in [Alon and Seymour 1989, Razborov 1992, Raz and Spieker 1993, Nisan and Wigderson 1994]; in the work of [Dietzfelbinger et al. 1994] the fooling set method is considered – it is shown that this method can never give significantly better lower bounds than those that can be proved by the rank method (it is still the case however that the fooling set method is easier to use); in [Karchmer et al. 1992] a generalization of the fooling set method is considered. This method, called *the rectangle size method*, is shown to give lower bounds which are essentially equal to the *nondeterministic* communication complexity; in [Kushilevitz et al. 1996] an example is shown that shows that all the above techniques, which are based on giving lower bound on the number of rectangles needed to cover the space of inputs, cannot always give optimal lower bounds.

Other types of randomized protocols – In Section 2.3 we define the notion of a randomized protocol. Other definitions for this notion appear in the literature. These include the definition of *Las-Vegas* protocols, where the protocol is not allowed to make errors but we consider its *average case* behavior (on the worst input) instead of the worst case behavior [Mehlhorn and Schmidt 1982, Fürer 1987]; Another model is the public coin model, where the two parties share a random string which is known to both of them and can be used during the execution of the protocol. This model was considered in [Newman 1991] who shows that the complexity of computing a function in this model is about the same as in the standard model, defined in Section 2.3.

Relations – In this chapter we concentrated on the task of computing a two-argument function $f(x, y)$. In general, more complicated tasks can be considered. One such task is the computation of relations (or *search problems*). That is, Alice and Bob need to find (on input (x, y)) an output z such that x, y and z satisfy some relation R. For example, they may be interested in finding an index i $(1 \leq i \leq n)$ such that the i-th bits of x and y are different. Such an output may not exist at all, or there may be several possible outputs for certain input (x, y). The computation of relations was considered in several papers [Karchmer and Wigderson 1988, Karchmer et al. 1991, Gringi and Sipser 1991, Edmonds et al. 1991]. This is mainly due to surprising relations found between the communication complexity of this kind of computations and the depth complexity of boolean circuits [Karchmer and Wigderson 1988]. For an exposition on this subject see [Karchmer 1989].

Multiparty protocols – In the model we considered so far there were only two parties. We can consider extensions of this model where there are $k > 2$ parties. The natural extension is to assume that each party P_i $(1 \leq i \leq k)$ holds an input x_i and the k parties wish to compute the value $f(x_1, \ldots, x_k)$. It turns out that a different extension is more useful for applications of the type given in Section 3. This extension is called the "number on the forehead" model. Imagine k players sitting in a circle each of them with a number x_i written on his forehead. Therefore, player P_i sees all the inputs *but* x_i. This models in a convenient way situations where there is an overlap of information between players: each x_i is known to most of the players (all but P_i). The multi-party model (and its applications) was studied, for example, in [Chandra et al. 1983, Babai et al. 1989, Grolmusz 1994, Håstad and Goldmann 1990, Babai et al. 1995, Pudlák and Rödl 1993].

3 Two-Party Communication Complexity – Applications

In this section we show how lower bounds (and upper bounds) in the area of communication complexity can be used to prove lower bounds in other areas. Many of these lower bounds can be proved without using communication complexity at all, however the use of communication complexity allows giving a uniform treatment to many of these lower bounds and to focus on the real reason for difficulty which is the communication bottleneck. This is done (in most of the cases) for problems in which communication does not seem to appear in the problem at all.

3.1 Finite Automata

In this section we are concerned with (deterministic and nondeterministic) *finite automata*. A deterministic finite automaton $\mathcal{A}$ has a finite set of states, Q, and a transition function $\delta : Q \times \Sigma \rightarrow Q$ which determines for each state and for each character in the alphabet Σ what should be the next state. Every input $w = w_1 w_2 \ldots w_m$ defines a sequence of $(m+1)$ states as follows: start with the *initial state* (a pre-specified state $q_0 \in Q$), and in the i-th step according to the current state q and the character w_i move to the state defined by δ (that is, to the state $\delta(q, w_i)$). If at the end the state of the automaton belongs to the set of *accepting states* (a pre-specified set $F \subseteq Q$) then we say that w is accepted by $\mathcal{A}$ or that w belongs to the *language* of $\mathcal{A}$. Otherwise, we say that w is rejected by $\mathcal{A}$ or that w does not belongs to the language of $\mathcal{A}$.

Our goal is to prove lower bounds on the number of states required in an automaton for a given language L (for convenience we consider only $\Sigma = \{0, 1\}$). We use the following lemma which relates the number of states to an appropriate communication complexity problem.

Lemma 12: *Let $f : \{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$ be a function. Let $L \subseteq \Sigma^*$ be a language and let $\mathcal{A}$ be an automaton as above (in particular, denote by Q the set of states of $\mathcal{A}$). Assume that*

$$L \cap \{0,1\}^{2n} \quad = \quad \{xy : \ |x| = |y| = n \text{ and } f(x,y) = 1\}.$$

Then, $D(f) \leq \log |Q| + 1$.

Proof: A protocol for f will work as follows: On input (x, y) (each of length n) Alice simulates the path taken by the automaton $\mathcal{A}$ on her input x. She then sends the name of the last state q in this path to Bob ($\log |Q|$ bits). Then, Bob simulates the automaton $\mathcal{A}$, starting from state q, on his input string y. All together, they get a simulation of $\mathcal{A}$ on input xy. Therefore, by assumption, if xy is accepted then $f(x, y) = 1$ while if xy is rejected then $f(x, y) = 0$. □

The following example shows a possible way to use this lemma. For a fixed n consider the language $L_n = \{xx \ : \ |x| = n\}$. This language can be written differently by:

$$L_n = \{xy \ : \ |x| = |y| = n \text{ and } EQ(x,y) = 1\},$$

where EQ is the equality function. Note that L_n is finite so there exists an automaton for L_n. However, by the above lemma, such an automaton has a set of states of size

$$|Q| \geq 2^{D(EQ)-1}.$$

Since we proved in Corollary 4 that $D(EQ) \geq n$ (and in fact it can be shown that $D(EQ) = n+1$) then this implies that every automaton for L_n has size

$$|Q| \geq 2^n.$$

Since this holds for every n, it follows that the language $L = \{xx \ : \ x \in \Sigma^*\}$ has no finite automaton (that is, L is not a *regular language*). Note that a similar argument shows that every automaton for the language

$$L'_n = \{xy \ : \ |x| = |y| = n \text{ and } x \neq y\},$$

also has size at least 2^n. It is easy to construct a *nondeterministic automaton* for L'_n of size $O(n)$. These two facts together imply that the known transformation of nondeterministic automata to deterministic automata (which takes a nondeterministic automaton of ℓ states and produces a deterministic automaton of 2^ℓ states) is essentially optimal.

For formal definitions of finite automata and many of their properties, including the transformation of nondeterministic automata into deterministic automata, the reader is referred to any of the standard texts in this area (e.g., [Hopcroft and Ullman 1979]).

3.2 Turing Machines

In this section we consider *Turing machines*. We will show how to use communication complexity for proving *time-space* tradeoffs for such machines. (Again, for formal definitions of Turing machines, as well as the notions of time and space, the reader is referred to any of the standard texts in computability or complexity, e.g., [Hopcroft and Ullman 1979].)

For convenience we will concentrate on a particular example, which is the language of *palindromes*. That is,

$$L_{\text{pal}} = \{ww^R \ : \ w \in \{0,1\}^*\},$$

where w^R denotes the string w reversed. We will prove the following:

Lemma 13: *Let M be a Turing machine that accepts the language L_{pal} in time $T(m)$ using $S(m)$ Space. Then,*

$$T(m) \cdot S(m) = \Omega(m^2).$$

Proof: Let M be a Turing machine. The machine M has a read-only input tape and some (fixed number of) read/write work tapes (the space is defined as the number of cells used on these work tapes; in other words, the input itself is not counted as part of the space). We describe a protocol for Alice and Bob that computes the equality function EQ as follows.

On input $(x,y) \in \{0,1\}^n \times \{0,1\}^n$, Alice and Bob simulate the Turing machine M with input $x0^ny^R$ (of length $m = 3n$). The first observation is that such an input is in the language L_{pal} if and only if $EQ(x, y) = 1$. The difficulty is of course that Alice only knows x while Bob only knows y. To solve this, Alice and Bob do the following: Alice can be responsible for the simulation of M as long as the head of the machine M on the input tape is either in the "x-region" of the input tape (first n bits) or in the "0-region" of the input tape (next n bits). When the head moves to the "y-region" of the input tape (last n bits), the responsibility is transferred to Bob. Symmetrically, Bob can be responsible for the simulation of M as long as the head of the machine M on the input tape is either in the "y-region" of the input tape or in the "0-region" of the input tape. When the head moves to the "x-region" of the input tape, the responsibility is transferred back to Alice. The idea is that each player is responsible for the simulation in regions where he knows the input bits, and hence will not need to get this information from the other player.

In any step in which responsibility is transferred the two players have to exchange all the information which is required for continuing the simulation. This information includes: the state of the machine M (which is $O(1)$ bits) and the content of the work tapes ($O(S(m))$ bits). Finally, since in the 0-region both players can do the simulation, and since to move from the x-region to the y-region and vice versa it takes at least n time steps (as this is the size of the 0-region) then the number of times that the responsibility is exchanges is at most $T(m)/n$. All together the number of bits exchanged in the protocol is $O(S(m) \cdot T(m)/n)$. Since, $D(EQ) \geq n$ (Corollary 4) then $T(m) \cdot S(m) = \Omega(n^2) = \Omega(m^2)$, as needed. $\square$

This tradeoff is essentially tight: It is not difficult to see that the language of palindromes can be recognized in linear time using linear space (by copying the first half of the input to one of the read/write tapes and then checking that the first half matches the second half) and, alternatively, in quadratic time using logarithmic space (by verifying for each i that the i-th bit matches the $(m - i)$-th bit).

Time-Space tradeoffs for Turing machines were studied for many years. They were first proved by [Cobham 1966] and then by many others. Similar techniques, but using a certain type of *randomized* communication complexity can be used to prove *time lower bounds* for *one-tape* Turing machines. Extensions of these techniques can be found in [Babai et al. 1989].

3.3 Decision Trees

A *decision tree* is a binary tree such that each of its internal nodes is labeled by a variable from $x_1, \ldots, x_m$, from each internal node there are two edges going to the children of this node one is labeled by 0 and the other is labeled by 1, and each leaf is labeled by either 0 or 1.

A decision tree computes a function $f : \{0, 1\}^m \rightarrow \{0, 1\}$ in the following way: given an assignment to the m variables, we start at the root of the tree; whenever we reach a node labeled by some variable x_i we consider the value of x_i in the assignment (0 or 1) and we proceed by going on the edge which is labeled by this value. When we reach one of the leaves (labeled 0 or 1) we take this label as the value of f on the assignment.

For example, in Figure 2 a decision tree is presented which computes a function f of 3 variables x_1, x_2 and x_3. It can be seen that $f(x_1, x_2, x_3) = 1$ if and only if $x_1 = x_2 = x_3$.

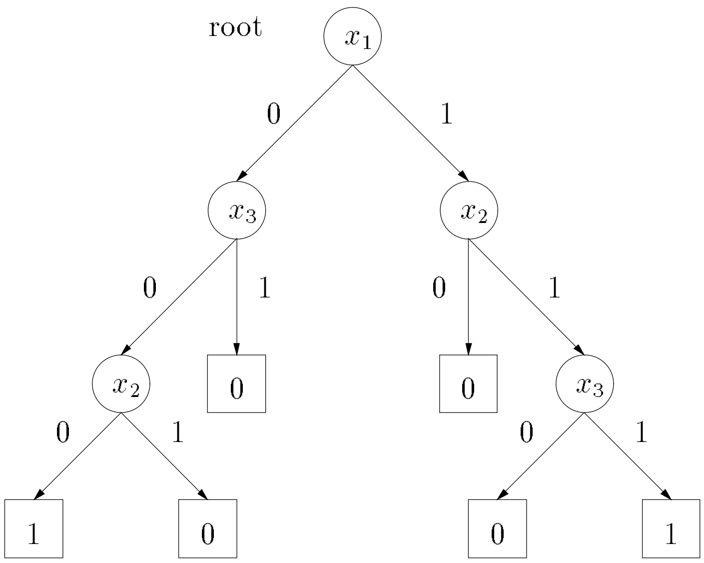

Figure 2: A Decision Tree

There are two interesting complexity measures with respect to decision trees: the *depth* (the length of the longest path from the root to a leaf) and the *size* (the number of nodes). Here we concentrate on the depth only. Clearly, for every function $f : \{0,1\}^m \rightarrow \{0,1\}$ there is a decision tree of depth m (simply by writing a complete binary tree of depth m, where all nodes in level i of the tree are labeled x_i; in this case each of the 2^m leaves corresponds to a single assignment; the label of the leaf is therefore the value of f on that assignment). The goal, of course, is to try to find decision trees of small depth. The following lemma allows getting lower bounds on the decision-tree depth using communication complexity lower bounds.

Lemma 14: *Let $m = 2n$ and $f : \{0,1\}^m \rightarrow \{0,1\}$ be a function. If f has a decision tree of depth d then the two-argument function*

$$f(x_1 \ldots x_n \ , \ x_{n+1} \ldots x_m)$$

has communication complexity $D(f) \leq d$.

Proof: Given a decision tree as above, Alice and Bob can simulate its computation. They start at the root. Whenever the simulation reaches an internal node of the tree the players look at the label x_j of the node and the player (Alice or Bob) that holds the value of this bit announces it. According to the value of x_j they determine the next node in the simulation. When the simulation reaches a leaf of the tree then the label of this leaf is the desired value of f. The number of bits exchanges is at most d. $\square$

The idea of proving lower bounds for decision trees using communication complexity lower bounds was introduced explicitly in [Nisan 1993] and implicitly in [Groger and Turan 1991]. This technique can be extended to more powerful decision trees that allow stronger operations in the nodes.

3.4 Ordered Binary Decision Diagrams (OBDDs)

A *Binary Decision Diagram* (BDD for short) is a directed acyclic graph, which consists of s nodes: $s-2$ nodes which are labeled by variables (from $x_1, x_2, \ldots, x_m$), one node labeled 0 and one node labeled 1. Each of the $s-2$ nodes labeled by variables has out-degree 2, where one of the edges going out of the node is labeled 0 and the other is labeled 1. The two nodes labeled by 0 and 1 are sinks (that is, have out-degree 0). There is a single node in the graph with in-degree 0 which is called the *root*. An *Ordered Binary Decision Diagram* (OBDD for short) is a BDD with the following property: there is a permutation (order) of the m variables $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ such that, on every path from the root to a sink, the order of variables is consistent with this permutation. An example of an OBDD appears in Figure 3.

A BDD (and in particular an OBDD) computes a function $f : \{0,1\}^m \rightarrow \{0,1\}$ in the following way: given an assignment to the m variables, we start at the root; whenever we reach a node labeled by some variable x_i we consider the value of x_i in the assignment (0 or 1) and we proceed by going on the edge which is labeled by this value. When we reach one of the sinks (labeled 0 or 1) we take this label as the value of f on the assignment. For example, the function f computed by the OBDD of Figure 3 is

$$f(x_1, x_2, x_3, x_4) = 1 \text{ iff } (x_1 = x_4) \text{ and } (x_2 = x_3).$$

OBDDs were introduced by [Bryant 1986]. They are widely used in logic design, formal verification and other fields (see [Bryant 1992]). The following lemma allows using communication complexity lower bounds to derive lower bounds on the size of OBDDs. Specifically, it uses the measure $D^{best}(f)$ introduced in Section 2.4.

Lemma 15: *Let $f : \{0,1\}^m \rightarrow \{0,1\}$ ($m = 2n$) be a function that can be computed by an OBDD of size s. Then*

$$s \geq 2^{D^{best}(f)-1}.$$

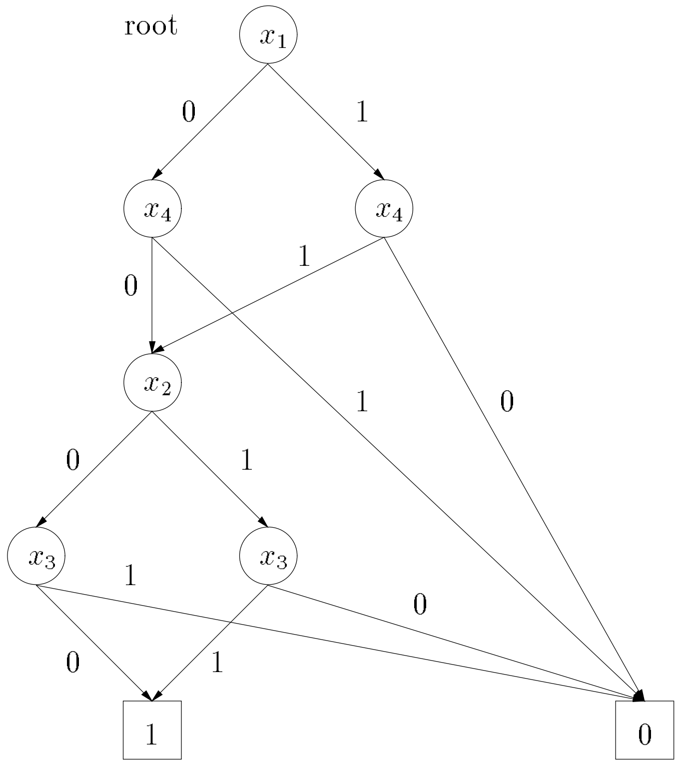

Figure 3: An OBDD with respect to the order x_1, x_4, x_2, x_3

Proof: Given an OBDD for computing f we first partition the input bits among Alice and Bob as follows: if the permutation used by the OBDD is $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ then we will give Alice the bits $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$ and give Bob the bits $x_{i_{n+1}}, x_{i_{n+2}}, \ldots, x_{i_m}$. We also fix a numbering of the s nodes in the OBDD. Alice starts the computation as defined by the OBDD: she evaluates each of the nodes which contains a variable which is in her possession. When she reaches a node labeled by a variable which she does not have (that is, x_{i_j} for $j > n$) she sends Bob the number of this node ($\log s$ bits). Bob can now proceed the computation until reaching one of the sinks (this is guaranteed by the assumption that the BDD is *ordered*). He sends Alice the output which is the label of this sink. All together, we get that $D^{best}(f) \leq \log s + 1$ as needed. □

The above lemma allows proving lower bounds on the OBDD size of functions for which we have good bounds on $D^{best}(f)$. For example, using Lemma 11, we get:

Corollary 16: *Every OBDD for the shifted-equality function, SEQ, has size*

$$s = 2^{\Omega(m)}.$$

This kind of technique can be further generalized to give so-called "width-length tradeoffs" for *oblivious branching programs*. See [Chandra et al. 1983, Alon and Maass 1986, Babai et al. 1989, Babai et al. 1990].

3.5 VLSI

In this section we show how communication complexity can be used for proving area-time tradeoffs for VLSI chips. A VLSI chip can be viewed as an $a \times b$ grid which has n input ports and one output port, there are gates on some of the vertices of the grid, and there are wires connecting the gates with other gates or with the ports (see Figure 4). We remark that this is not the most general VLSI model however the results presented here can be extended to more complicated (and realistic) models. See [Lengauer 1990] for a survey on VLSI and applications of communication complexity to VLSI.

The two most important complexity measures for a VLSI chip are A, its *area* (that is $A = a \cdot b$) and T, the *time* (number of steps) it takes from the time that the input is provided in the input ports to the time the result appears in the output port (in each step each gate computes some function of the values given on the wires entering this gate and sends this value on the outgoing wire). The designer of a chip that computes a function $f(x_1, \ldots, x_m)$ has to decide what gates to use, how to connect them and in particular he is free to choose which input x_i should be fed into each input port.

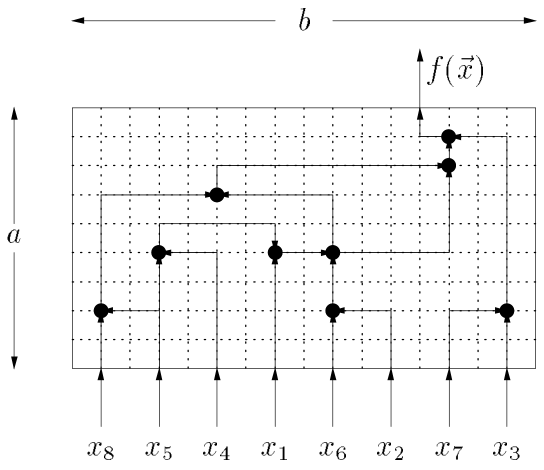

Figure 4: A VLSI chip (gates are denoted by $\bullet$)

The following lemma allows proving, for certain functions f, that every VLSI chip computing them either have a "large" area or requires "many" time steps. More precisely, we prove lower bounds on the quantity AT^2 of VLSI chips computing certain functions f, based on the communication complexity $D^{best}(f)$.

Lemma 17: *Suppose that we are given a VLSI chip as above with area A and time T to compute a function $f : \{0,1\}^m \to \{0,1\}$ ($m = 2n$). Then,*

$$AT^2 \geq (D^{best}(f))^2.$$

Proof: Given the chip, we construct a two-party protocol as follows. Assume, without loss of generality, that $a \leq b$. We can always cut the chip into two pieces in a way that partitions the input ports of the chip into two equal size sets, and that at most a wires are connecting the two pieces. We now construct a protocol for f that shows that $D^{best}(f) \leq \sqrt{A} \cdot T$.

For this, we assign each of Alice and Bob one of the two pieces of the chip. That is, we partition the input bits between Alice and Bob in the same way that the input bits are partitioned among the two pieces. Now, Alice and Bob simulate the computation of the chip. In step i of the protocol each of the two players evaluates the output of all the gates in his piece of the chip in time i. The only information that the two players exchange is the values which are going on the wires connecting the two pieces. Since the number of these wires is at most a, and the number of steps to compute the output is at most T, the total number of bits exchanged is at most $a \cdot T \leq \sqrt{A} \cdot T$, as needed. □

We can now prove results for functions for which we have good bounds on $D^{best}(f)$. For example, using Lemma 11, we get:

Corollary 18: *Every VLSI chip for the shifted-equality function, SEQ, satisfies*

$$A \cdot T^2 = \Omega(m^2).$$

3.6 Threshold Circuits

Threshold circuits are circuits whose gates are threshold elements. These circuits are used in what is called (discrete) neural computation. That is, in oppose to the more general concept of "neural network", which typically uses real numbers and each unit in the network (a "neuron") computes some real function ("sigmoid"), here each gate (unit) in the network gets as an input boolean variables and outputs a boolean variable which indicates whether the weighted sum of the inputs is larger than some threshold.

Formally, a threshold circuit is a directed acyclic graph with m nodes of in-degree 0 labeled by $x_1, \ldots, x_m$ and the other nodes are called *gates*. One of the gates, has out-degree 0, and is called the *output* gate. Now, each gate computes a weighted sum of the edges entering the gate and outputs 1 if and only this sum is larger than some value θ (that is, the gate has t edges entering it, $z_1, \ldots, z_t$, and there are $t+1$ real numbers associated with the gate $w_1, \ldots, w_t$ and θ; the gate computes the result of the comparison "$\sum_{i=1}^{t} w_i \cdot z_i > \theta$?").

Our goal is to give lower bounds for such circuits using communication complexity. In oppose to previous examples here we are using lower (and upper) bounds on *randomized* communication complexity (and not on the deterministic communication complexity). For convenience we will concentrate on a specific example – the inner product function:

Lemma 19: *Any threshold circuit that computes the inner-product function $IP(x_1 \ldots x_n, x_{n+1} \ldots x_m)$ (where $m = 2n$) has $\Omega(n/\log^2 n)$ gates.*

Proof: We first use a basic fact about threshold circuits which states that without loss of generality, all the w_i's and θ are $O(m \log m)$-bit integers [Muroga 1971]. Using this fact we can now construct a randomized protocol to compute the function IP. Assume that the circuit has $s \leq n$ gates (if not then we are done). Alice and Bob simulate the threshold gates one by one in a "bottom-up" manner. The last gate they evaluate is the output gate whose value is the desired value. To evaluate a particular gate, whose edges are denoted $z_1, \ldots, z_t$ they do the following: some of the z_i's are input bits which are known to one of the players, while others are output bits of previously simulated gates and are known to both players. Partition these output bits arbitrarily among Alice and Bob and let A be the set of i's such that z_i is known to Alice and let B be the set of i's such that z_i is known to Bob (that is, A and B are disjoint sets whose union is $1, \ldots, t$). Now, to make the comparison

$$\sum_{i=1}^{t} w_i \cdot z_i > \theta$$

it is convenient to make the comparison

$$\sum_{i \in A} w_i z_i > \theta - \sum_{i \in B} w_i z_i$$

since the left-hand term can be computed by Alice and the right-hand term can be computed by Bob. Moreover, each of the two terms is $O(m \log m)$-bit long so Alice and Bob can compare these two numbers using the randomized protocol of Lemma 8. The cost of using this protocol is $O(\log^2 m)$ bits. The error probability is at most $1/4n$ for each gate, so if there are less than n gates the error probability is at most $1/4$. All together, we have a randomized protocol to compute the function IP using $O(s \cdot \log^2 m)$ bits. By Lemma 9, $R(IP) = \Omega(m)$ which implies that $s = \Omega(m/\log^2 m)$, as needed. $\square$

The connection between communication complexity and the size of threshold circuits was shown in [Nisan 1993, Roychowdhury et al. 1994]. For a lot of information about the area of discrete neural computation and in particular some connections with communication complexity, see [Goldmann 1994, Siu et al. 1995].

3.7 Other Applications

In this section we briefly mention some other applications of communication complexity. Additional applications may be found using the sources appear in the bibliography.

Boolean Circuits – A boolean circuit is a circuit consisting of AND, OR and NOT gates. Proving "good" lower bounds on the size or depth of boolean circuits is a long standing open problem in computational complexity that goes back to [Shannon 1948]. For example, a super-polynomial lower bound on the size of circuits for any function in NP would imply that $P \neq NP$. [Karchmer and Wigderson 1988] show that boolean circuits and communication problems of a certain type are closely related. More specifically, for a function $f : \{0,1\}^n \rightarrow \{0,1\}$ they define the following communication problem. The input for one party is a string $x \in \{0,1\}^n$ such that $f(x) = 1$ and the input for the second party is a string $y \in \{0,1\}^n$ such that $f(y) = 0$. The communication problem is for the parties to find an index i such that $x_i \neq y_i$ (this type of problems is called "relations"; see Section 2.5). In particular, it was shown that the communication complexity of this communication problem equals the depth complexity of boolean circuits computing the function f. A similar relationships exist between another type of communication problems and *monotone circuits* (these are circuits which contain only AND and OR gates). Using the connection to communication complexity several lower bounds on monotone circuits depth were proved in [Karchmer and Wigderson 1988, Raz and Wigderson 1990, Gringi and Sipser 1991]. Another approach that allows using results in communication complexity to prove lower bounds on the depth of monotone circuits was shown in [Razborov 1990b]. Connections between communication complexity and monotone *constant depth* circuits were found in [Nisan and Wigderson 1991].

Networks – Communication complexity lower bounds in the two-party model can be used to prove time lower bounds for more complicated (multi-party) networks. Roughly speaking, the method consists of two stages. First, we partition the network into two "large" parts such that the number of edges connecting the two parts (sometimes called the *bandwidth*) is "small". For example, if the network is a linear array of p processors, a useful partition of the network is the first $p/2$ processors versus the last $p/2$ processors. The number of edges connecting the two parts in this case is 1. For a more sophisticated example, consider any planar network of p processors. A known result of [Lipton and Tarjan 1980] shows that any such network can be partitioned into two parts, each containing at least $p/3$ processors, such that the number of edges connecting the two parts if $O(\sqrt{p})$. In the second stage, we view the computation done by the network as a two party communication complexity problem, where each party gets as an input, all the inputs given to processors in one of the parts of the network. If the resulted two-party communication problem requires the submission of at least d bits then this implies that the time that it takes for the network to complete this computation is at least d divided by the bandwidth (the time it takes to submit a single bit over an edge is defined as the time unit). See [Leighton 1991] for applications of this method to proving lower bounds in various types of networks.

Data structures – [Miltersen 1994] discovered connections between communication complexity and data-structures. These connections are used to prove lower bounds for data-structures supporting certain types of queries, both static (where the data-structure is constructed once and then only queries are asked) and dynamic data-structures (where in addition the content of the data-structure can be modified) [Miltersen 1994, Miltersen et al. 1995]. To see the idea, consider the following typical (static) data structures problem: given a set $B \subseteq \{1, \ldots, n\}$ we want to store information in our memory in a way that will later allow answering questions of the type "is $i \in B$?". An implementation for this problem consists of a mapping that determines for each such set B the content of the memory, and a method for answering queries using the content of the memory. There are two measures to evaluate the quality of an implementation. The first measure is the space; that is, the number of memory cells (each consists of b bits) that are used. The second measure is called the time and is defined as the number of cells that should be read in order to answer a query. We can view this data structures problem as a communication complexity problem by considering two parties. One party gets as an input a set B and the other party gets as an input an element i. Their goal is to check whether $i \in B$. It can be shown that any implementation for the data structures problem implies the existence of a protocol for the communication complexity problem whose complexity depends on the time and space complexities of the data structure implementation. Therefore, bounds for the communication complexity problem yield time-space tradeoffs for the data structure problem.

4 Conclusions

We provide an introduction to theory of two party communication complexity. This theory is far more rich than what is presented in this short survey. It exhibits a beautiful mathematical structure who can be studied by using various mathematical tools from linear algebra, probability, combinatorics and other related fields.

Then, we argue that results in communication complexity can be applied to many problems in the theory of computation (and beyond that). By this, we claim that communication complexity proves itself to be a fundamental key in the understanding of computation.

Acknowledgments

I wish to thank Amos Beimel for his comments on an early draft of this chapter.

References

[Aho et al. 1983] A. Aho, J. Ullman, and M. Yannakakis, "On Notions of Information Transfer in VLSI Circuits", *Proc. of 15th STOC*, pp. 133-139, 1983.

[Alon and Maass 1986] N. Alon, and W. Maass, "Meanders and their Applications in Lower Bound Arguments", *JCSS*, Vol. 37, No. 2, pp. 118-129, 1988. (Early version in *Proc. of 27th FOCS*, pp. 410-417, 1986.)

[Alon and Seymour 1989] N. Alon, and P. D. Seymour, "A Counterexample to the Rank-Coloring Conjecture", *J. of Graph Theory*, Vol. 13, pp. 523-525, 1989.

[Babai et al. 1995] L. Babai, P. Kimmel, and S. V. Lokam, "Simultaneous Messages vs. Communication", *Proc. of 12th STACS*, LNCS 900, pp. 361-372, 1995.

[Babai et al. 1989] L. Babai, N. Nisan, and M. Szegedy, "Multiparty Protocols, Pseudorandom Generators for LOGSPACE, and Time-Space Trade-offs", *JCSS*, Vol. 45, No. 2, pp. 204-232, 1992. (Early version in *Proc. of 21st STOC*, pp. 1-11, 1989.)

[Babai et al. 1990] L. Babai, P. Pudlák, V. Rödl, and E. Szemeredi, "Lower Bounds to the Complexity of Symmetric Boolean Functions", *Theoretical Computer Science*, Vol. 74, pp. 313-323, 1990.

[Bryant 1986] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulations", *IEEE Transactions on Computers*, Vol. 35, pp. 677-691, 1986.

[Bryant 1992] R. E. Bryant, "Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams", *ACM Computing Surveys*, Vol. 24, No. 2, pp. 293-318, 1992.

[Chandra et al. 1983] A. K. Chandra, M. L. Furst, and R. J. Lipton, "Multyparty Protocols", *Proc. of 15th STOC*, pp. 94-99, 1983.

[Chor and Goldreich 1985] B. Chor, and O. Goldreich, "Unbiased Bits from Sources of Weak Randomness and Probabilistic Communication Complexity", *SIAM J. Computing*, Vol 17, No 2, pp. 230-261, 1988. (Early version in *Proc. of 26th FOCS*, pp. 429-442, 1985.)

[Cobham 1966] A. Cobham, "The Recognition Problem for the Set of Perfect Squares", Technical report RC-1704, IBM, 1966.

[Dietzfelbinger et al. 1994] M. Dietzfelbinger, J. Hromkovic, and G. Schnitger, "A Comparison of Two Lower Bound Methods for Communication Complexity", *Proc. of MFCS94*, LNCS 841, pp. 326-335, 1994.

[Ďuriš et al. 1984] P. Ďuriš, Z. Galil, and G. Schnitger, "Lower Bounds on Communication Complexity", *Information and Computation*, Vol. 73, No. 1, pp. 1–22, 1987. (Early version in *Proc. of 16th STOC*, pp. 81-91, 1984.)

[Edmonds et al. 1991] J. Edmonds, S. Rudich, R. Impagliazzo, and J. Sgall, "Communication complexity towards lower bounds on circuit depth", *Proc. of 32nd FOCS*, pp. 249-257, 1991.

[Feder et al. 1991] T. Feder, E. Kushilevitz, M. Naor, and N. Nisan, "Amortized Communication Complexity", *SIAM J. Computing*, Vol. 24, No. 4, pp. 736-750, 1995. (Early version in *Proceedings of the 32nd FOCS*, pp. 239-248, 1991.)

[Fürer 1987] M. Fürer, "The Power of Randomness for Communication Complexity", *Proceedings of the 19th STOC*, pp. 178-181, 1987.

[Goldmann 1994] M. Goldmann, "Communication Complexity and Lower Bounds for Threshold Circuits", Chapter 3 in *Theoretical Advances in Neural Computation and Learning*, V. Roychowdhury, K. Y. Siu, and A. Orlitsky (Eds.), Kluwer Academic Publishers, pp. 85-125, 1994.

[Gringi and Sipser 1991] M. Gringi, and M. Sipser, "Monotone Separation of Logarithmic Space from Logarithmic Depth", *JCSS*, Vol. 50, pp. 433-437, 1995. (Early version in *Proc. of 6th Structure in Complexity Theory*, pp. 294-298, 1991.)

[Groger and Turan 1991] H.D. Groger, and G. Turan, "On Linear Decision Trees Computing Boolean Functions", *Proceedings of 18th ICALP*, LNCS 510, pp. 707–718, 1991.

[Grolmusz 1994] V. Grolmusz, "The BNS Lower Bound for Multi-Party Protocols is Nearly Optimal", *Information and Computation*, Vol. 112, No. 1, pp. 51-54, 1994.

[Halstenberg and Reischuk 1988] B. Halstenberg, and R. Reischuk, "Different Modes of Communication", *SIAM J. Computing*, Vol. 22, No. 5, pp. 913-934, 1993. (Early version in *Proc. of 20th STOC*, pp. 162-172, 1988.)

[Håstad and Goldmann 1990] J. Håstad, and M. Goldmann, "On the Power of Small-Depth Threshold Circuits", *Proc. of 31st FOCS*, pp. 610-618, 1990.

[Hopcroft and Ullman 1979] J. E. Hopcroft, and J. D. Ullman, "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, 1979.

[Kalyanasundaram and Schnitger 1987] B. Kalyanasundaram, and G. Schnitger, "The Probabilistic Communication Complexity of Set Intersection", *SIAM J. Discrete Math.*, Vol. 5, No. 4, pp. 545-557, 1992. (Early version in *Proc. of 2nd Structure in Complexity Theory*, pp. 41-49, 1987.)

[Karchmer 1989] M. Karchmer, "Communication Complexity: A New Approach to Circuit Depth", The MIT Press, 1989.

[Karchmer et al. 1992] M. Karchmer, E. Kushilevitz, and N. Nisan, "Fractional Covers and Communication Complexity", *SIAM J. Discrete Math.*, Vol. 8, No. 1, pp. 76-92, 1995. (Early version in *7th IEEE Structure in Complexity Theory*, pp. 262-274, 1992.)

[Karchmer et al. 1991] M. Karchmer, R. Raz, and A. Wigderson, "On Proving Super-Logarithmic Depth Lower Bounds via the Direct Sum in Communication Complexity", *Proc. of 6th IEEE Structure in Complexity Theory*, pp. 299–304, 1991.

[Karchmer and Wigderson 1988] M. Karchmer, and A. Wigderson, "Monotone Circuits for Connectivity Require Super-Logarithmic Depth", *SIAM J. Discrete Math.*, Vol. 3, No. 2, pp. 255-265, 1990. (Early version in *Proc. of 20th STOC*, pp. 539-550, 1988.)

[Kushilevitz et al. 1996] E. Kushilevitz, N. Linial, and R. Ostrovsky, "The Linear-Array Conjecture of Communication Complexity is False", *Proc. of 28th STOC*, 1996.

[Kushilevitz and Nisan 1996] E. Kushilevitz, and N. Nisan, "Communication Complexity", Cambridge University Press, to appear.

[Leighton 1991] F. T. Leighton, "Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes", Morgan-Kaufmann, 1991.

[Lengauer 1990] T. Lengauer, "VLSI Theory", in *Handbook of Theoretical Computer Science*, Vol. A, Chapter 16, pp. 835-868, 1990.

[Lipton and Sedgewick 1981] R. J. Lipton, and R. Sedgewick, "Lower Bounds for VLSI", *Proc. of 13th STOC*, pp. 300-307. 1981.

[Lipton and Tarjan 1980] R. J. Lipton, and R. E. Tarjan, "Applications of A Planar Separator Theorem", *SIAM J. Computing*, Vol. 9, pp. 615-627, 1980.

[Lovász 1989] L. Lovász, "Communication Complexity: A Survey", in *Paths, Flows, and VLSI Layout*, edited by B. H. Korte, Springer Verlag, Berlin New York, 1990.

[Mehlhorn and Schmidt 1982] K. Mehlhorn, and E. Schmidt, "Las-Vegas is better than Determinism in VLSI and Distributed Computing", *Proc. of 14th STOC,* pp. 330-337, 1982.

[Miltersen 1994] P. B. Miltersen, "Lower Bounds for Union-Split-Find Related Problems on Random Access Machines", *Proc. of 26th STOC,* pp. 625-634, 1994.

[Miltersen et al. 1995] P. B. Miltersen, N. Nisan, S. Safra, and A. Wigderson, "On Data Structures and Asymmetric Communication Complexity", *Proc of 27th STOC*, pp. 103-111, 1995.

[Muroga 1971] S. Muroga, "Threshold Logic and its Applications", Wiley Interscience, 1971.

[Newman 1991] I. Newman, "Private vs. Common Random Bits in Communication Complexity", *Information Processing Letters* 39, pp. 67-71, 1991.

[Nisan 1993] N. Nisan, "The Communication Complexity of Threshold Gates", *Combinatorics, Paul Erdös is Eighty (Vol 1)*, Miklós, Sós and Szönyi (Eds.), Bolyai Math. Society, pp. 301-315, 1993.

[Nisan and Wigderson 1991] N. Nisan, and A. Wigderson, "Rounds in Communication Complexity Revisited", *SIAM J. Computing*, Vol. 22, No. 1, pp. 211-219, 1993. (Early version in *Proc. of 23rd STOC*, pp. 419-429, 1991.)

[Nisan and Wigderson 1994] N. Nisan, and A. Wigderson, "On Rank vs. Communication Complexity", *Combinatorica*, Vol. 15, No. 4, pp. 557-566, 1995. (Early version in *Proc. of 35th FOCS*, pp. 831-836, 1994.)

[Orlitsky and El-Gamal 1988] A. Orlitsky, and A. El-Gamal, "Communication Complexity", in *Complexity in Information Theory*, Y. S. Abu-Mostafa (ed), pp. 16-61, 1988.

[Papadimitriou and Sipser 1982] C. Papadimitriou, and M. Sipser, "Communication Complexity", *JCSS*, Vol. 28, No. 2, pp. 260-269, 1984. (Early version in *Proc. of 14th STOC*, pp. 196-200, 1982.)

[Pudlák and Rödl 1993] P. Pudlák, and V. Rödl, "Modified Ranks of Tensors and the Size of Circuits", *Proc of 25th STOC*, pp. 523-531, 1993.

[Raz and Spieker 1993] R. Raz, and B. Spieker, "On the "log rank" Conjecture in Communication Complexity", *Combinatorica*, Vol. 15, No. 4, pp. 567-588, 1995. (Early version in *Proc. of 34th FOCS*, pp. 168-176, 1993.)

[Raz and Wigderson 1990] R. Raz, and A. Wigderson, "Monotone Circuits for Matching Require Linear Depth", *JACM*, Vol. 39, No. 3, pp. 736-744, 1992. (Early version in *Proc. of 22nd STOC*, pp. 287-292, 1990.)

[Razborov 1990] A. A. Razborov, "On the Distributional Complexity of Disjointness", *Theoretical Computer Science*, Vol. 106, No. 2, pp. 385-390, 1992. (Early version in *Proc. of 17th ICALP*, LNCS 443, pp. 249-253, 1990.)

[Razborov 1990b] A. A. Razborov, "Applications of Matrix Methods to the Theory of Lower Bounds in Computational Complexity", *Combinatorica*, Vol. 10, No. 1, pp. 81-93, 1990.

[Razborov 1992] A. A. Razborov, "The Gap between the Chromatic Number of a Graph and the Rank of its Adjacency Matrix is Superlinear", *Discrete Math.*, Vol. 108, pp. 393-396, 1992.

[Roychowdhury et al. 1994] V. P. Roychowdhury, A. Orlitsky, and K. Y. Siu, "Lower Bounds on Threshold and Related Circuits via Communication Complexity", *IEEE Transactions on Information Theory*, Vol. 40, No. 2, pp. 467-474, 1994.

[Shannon 1948] C. E. Shannon, "A Mathematical Theory of Communication", *Bell System Tech. Jour.*, Vol. 27, pp. 379-423, 623-656, 1948.

[Siu et al. 1995] K. Y. Siu, V. Roychowdhury, and T. Kailath, "Discrete Neural Computation – A Theoretical Foundation", Prentice Hall, 1995.

[Thompson 1979] C. D. Thompson, "Area-Time Complexity for VLSI", *Proc. of 11th STOC*, pp. 81-88, 1979.

[Yao 1979] A. C. Yao, "Some Complexity Questions Related to Distributed Computing", *Proc. of 11th STOC*, pp. 209-213, 1979.