Abstract

In this work we try to approach one phenomenon from several different sides. The phenomenon is the relation between the computing power of a computational model and its capacity of transferring data between its parts.

We consider the following models:

1. Bounded depth circuits with Boolean gates ($\wedge$, $\vee$, $\neg$) and gates that output 0 or 1 according whether the number of 1's in their input is divisible by fixed m or not.
2. Bounded depth circuits with threshold gates
3. Bounded width branching programs
4. Raw communication models

Each of these models are capable of computing any Boolean function, but for certain functions such a computation is very costly. This cost is measured by the *size* in models 1 and 2, by the length in model 3 and by the number of bits necessary to exchange between the parts in the model 4. Our purpose is to find simple functions that are costly to compute.

Algebraic methods have been particularly successful in this area. We further develop these tools. We make use, among others, of character sum estimates (for a lower bound proof) and commutative semigroups (in a characterization theorem).

What we believe is fascinating is the interrelation between the models considered above. The raw communication model by its mathematical purity might be the key of understanding the behavior of all these models.

In Chapter 2 we discuss the Moebius transformation and the Fourier transformation of functions defined on $\{0,1\}^n$.

In Chapter 3 we give a lower bound for multiparty communication protocols, and mention numerous consequences on branching programs, time-space tradeoffs for Turing machines, pseudorandom sequences.

In Chapter 4 we characterize te functions with bounded symmetric communication complexity as well as bounded depth circuits combined from gates computing such functions.

In Chapter 5 we prove lower bounds on depth 2 threshold circuits that compute the INNER PRODUCT MOD 2 function.

Contents

Chapter 1

Introduction

1.1 Historical background

In this work we use techniques of classical algebra and number theory in order to gain a better understanding of certain problems of computational complexity.

Theoretical Computer Science and Mathematics have a history of fruitful interaction. Perhaps the first area of Mathematics that had a serious impact on CS was Mathematical Logic: TCS started with the work if logicians like Turing and von Neuman, who not only did the pioneering work in the study of basic capabilities of computational processes, but also were partly responsible for building some of the first computing devices. The interaction continues to these days. As an illustration it should be enough to note that the most celebrated open problem of computational complexity theory, the P vs. NP question was proposed in the context of theorem-proving procedures by S.A. Cook in 1974 **??** (and, as it became known recently, essentially the same question was proposed by one of the greatest logicians of the century, Kurt Godel in a letter to J. von Neuman in 1954).

Other areas of Mathematics that have been associated with Computer Science include Numerical Analysis (in large-scale scientific calculations), Linear Programming, and Combinatorics. The latter two areas have played an important role in the design of efficient algorithms and data structures.

Probabilistic methods have been used to improve the performance of algorithms (in areas as diverse as primality testing, message routing in networks, and sorting and retrieving information). Probabilistic methods were also used to prove the existence of combinatorial objects that are useful for

certain algorithms (e.g. expander graphs, superconcentrators).

Algebraic theories have played a role in the theory of automata and formal languages. Linear algebra, Fast Fourier Transformation and other algebraic techniques have been useful in the design of efficient algorithms. Strassen was the first to introduce methods of algebraic geometry in the study of lower bounds for arithmetic circuits fifteen years years ago. It is however a recent discovery that methods of classical algebra and number theory can be used to prove lower bounds in universal models of computation (Boolean circuits, Branching programs). Such methods were introduced in the seminal work of Razborov and generalized by Smolensky. Another recent accomplishment is Barrington's characterization of NC^1 as having power equal to that of bounded width branching programs, exploiting the unsolvability of the symmetric group of degree 5.

The theory of communication complexity, in the sense of Yao **??**, a major step towards a theory of complexity for parallel and distributed computations, has also proven to be a fertile ground for algebraic techniques. Certain measures of communication complexity can be bounded in terms of rank of associated matrices. The best lower bound for the probabilistic communication complexity of a boolean function was obtained through the use of the nontrivial Milnor-Thom theorem on the number of components of a real algebraic variety.

Particularly deep mathematical theorems from group representation have been used to explicitly construct expander graphs Lubotzky, Philips and Sarnak, and to estimate the mixing speed of Markov processes.

More recently, the Fourier transform of functions on Z_2^n has been used successfully to prove extremal properties of Boolean functions. Kahn, Kalai and Linial making a substantial use of Beckner's inequality proved that "Boolean functions always have small dominant set of variables".

The use of such algebraic techniques is appealing for several reasons. First, it is a new technique in an area that is notorious for lack of adequate tools. Second, many questions in TCS have an intrinsicly algebraic flavor: proving that a certain computational task cannot be done with a certain amount of resources has the same character as proving that certain constructions are impossible with a ruler and compass (of course the latter problem is THE classic success story of Algebra). Third, many of the TCS questions suggest algebraic problems that are interesting in their own right as algebraic problems. Finally, there are deep and powerful techniques in Algebra, that may shed some light on the difficult problems of TCS.

1.2 Embedding the Boolaen functions into function rings

The nonuniform complexity of a Boolean function — roughly speaking — is the number of *elementary operations* that are necessary to produce the Boolean function from the elementary functions $x_1, x_2, ..., x_n$. We get different complexity measures by defining the notion of 'elementary operation' in different ways. We can also impose some restrictions on the depth of the formula that results the Boolean function in question.

The study of the *algebraic structure* of the Boolean functions with respect to these operations can help us to compute the complexity of single Boolean functions.

The Boolean functions on n Boolean variables form a Boolean algebra with respect to the pointwise Boolean operations ($\wedge$, $\vee$, $\neg$). We can also consider them as the elements of the algebra $(GF(2))^{2^n}$ over GF(2) where the operations are the modulo 2 addition and the logical and (which plays the role of the multiplication).

Razborov in **??** considered the latter structure in order to achieve exponential *size*-lower bound on the majority function for bounded depth circuits. In this model unbounded fan-in $\wedge$, $\vee$ and mod2 gates are allowed and the longest path from the input gates to the output gate is bounded by a constant.

Smolensky extended his methods and proved that if the circuit contains mod$p$ gates[1] for an arbitrary prime p rather than mod2 gates then for a different prime q no mod q function can be computed in subexponential size.

In his paper {30 Smolensky considers the Boolean functions as elements in the algebra $\mathbf{F}^C$, where F is a *Galois*-field and $\mathbf{F}^C$ is the algebra of the functions $\{0,1\}^n \to F$ over F. There is a natural isomorphism between $\mathbf{F}^C$ and the multilinear polynomials in n variables over F which makes it possible to assign a natural number to every element of $\mathbf{F}^C$, namely the *degree* of the corresponding polynomial.

Razborov and Smolensky in their argument use degree estimations that they combine with the *approximation technique* introduced by Razborov.

In section 2.1 we describe some special properties of $\mathbf{R}^C$.

The notion sensitivity....

In section 2.2 we study some approximation schemes and prove that

[1] We understand that a mod$p$ gate outputs *zero* if the number of the ones in its input is congruent to *zero* modulo p and it outputs *one* otherwise.

any function approximating a function with high sensitivity must have high degree.

In section 2.3 we use the Fourier-transformation approach which utilizes the isomorphism between $\mathbf{F}^C$ and the group algebra of Z_2^n over F. We give a new and simpler proof of the uncertainity principle stated in [].

In section 2.4 the Boolean functions correspond to subsets of certain abelian groups. We prove that AC^0 circuits with a single $\bmod 6$ gate on the top cannot compute the majority function.

1.3 Multiparty protocols

The original version of the communication game introduced by A.Yao **??** is a two player game choreographed by a *protocol* agreed upon by both parties before the game starts. The players cooperatively evaluate a Boolean function $f(\mathbf{x}, \mathbf{y})$ the input of which is distributed between the players in such a way that Player 1 gets $\mathbf{x}$ and Player 2 gets $\mathbf{y}$. The name of the game is *message exchange*. Both players have infinite computation power so that they can compute the next message the protocol prescribes for them. The only resource that counts is the number of bits they have to send to each other. If the protocol is optimal this value is the *communication complexity* of f.

In Chapter 3 we study the version of the above game when k players wish to evaluate the value of a Boolean function $f(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. They are sitting around a table and the i^{th} player has $\mathbf{x}_i$ written on his forehead for $i = 1, \ldots, k$ The game is very similar to the two player game. The protocol tells which player sends the next message. All the players can hear all the messages and they can remember them. We suppose that the very last message is the value of f.

The k party game was introduced first by Chandra Furst and Lipton **??**. They proved that if $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are numbers from 1 up to p and f tells whether their sum taken modulo p is 0 then the complexity of a multiparty protocol that computes f is at least $\sqrt{\log p}$. Using this result they were able to prove that bounded width branching program that has linear length cannot compute the majority function.

Let us denote the function that a protocol P computes also by P. We also introduce the shorthand notion $Pr(P = f)$ for the probability of the event that a protocol P computes a function f for a random string $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$. The ε-distributional communication complexity of f is the length of the

shortest protocol P that achieves $Pr(P = f) \geq 1/2 + \varepsilon$. We denote this value by $C_\varepsilon(b)$.

Let $\mathbf{x}_1, \ldots, \mathbf{x}_k$ integers between 0 and $p-1$ for a prime p. Let f be the Boolean function that takes value 1 iff $\sum_{i=1}^{n} x_i$ is quadratic residue modulo p.

In Chapter 3 we prove that

$$C_\epsilon(f) \geq \Omega(\frac{\log p}{2^k} + log\varepsilon).$$

This lower bound and a similar one for the GENERALIZED INNER PRODUCT MOD 2 function has several applications.

As the first we mention that a pseudorandom generator for Logspace can be constructed. The generator produces in polynomial time pseudorandom sequences of length $\exp(c\sqrt{\log n})$ that no Logspace Turing maschine will be able to distinguish from truly random sequences. As a corollary we give an explicit construction of universal traversal sequence of length $\exp(\exp(c\sqrt{\log n}))$ for arbitrary undirected graphs on n vertices.

The multiparty protocol lower bounds can be applied to derive to derive several new time-space tradeoffs. In **??** we give a tight time-space tradeoff of te form $TS = \Theta(n^2)$, for general, k-head Turing Machine. In the same paper we give a new length-width tradeoff for oblivious branching programs, or on the size of Boolean formulas (over an arbitrary finite base).

1.4 Functions with bounded symmetric communication complexity

In Chapter 4 we consider Boolean functions which have bounded symmetric communication complexity, i.e. bounded communication complexity under all partitions of the input. We give an explicit characterization of these classes of functions by showing that they can be represented in a natural way by commutative semigroups of bounded size. As a consequence we deduce that unbounded fanin circuits with gates with bounded symmetric communication complexity can be simulated by circuits with AND, OR and MOD_m gates for bounded m, with only linear increase in size and depth. This shows the robustness of the concept of bouned depth, poly size circuits with such gates suggesting a possible new approach to lower bounds for such circuits.

1.5 Introduction

The results of Ajtai [], Furst-Saxe-Sipser [], Hastad [], Razborov [], Smolensky [] give us that with respect to the bounded dept polynomial size reducibility ($\leq_{bp}$) [], $\emptyset <_{bp}$ PARITY $<_{bp} MAJORITY$.

The next step in extending this hierarchy would be to study bounded depth, polynomial size circuits where MAJORITY is also allowed as a single gate.

In this chapter we consider these circuits and the complexity class TC^0 of Boolean functions computed by them. This class was first studied by Parberry-Schnitger []. The definition we use allows gates computing *threshold functions* of the form $T_k^{\vec{\alpha}}(y_1, y_2, \ldots, y_m)$ iff $\sum_{i=1}^{m} \alpha_i y_i \geq k$ (where $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$).

Circuits with threshold gates are also related to several computational brain models (e.g. Boltzmann machines []) and to other models related to pattern recognition and learning, some of which have been studied since the 1950's (e.g. perceptrons [], []).

We shaw that depth 2 threshold circuits computing INNER PRODUCT MOD 2 must have exponential size. As INNER PRODUCT MOD 2 can be computed by polynomial size depth 3 threshold circuits this gives a separation of the corresponding complexity classes.

Chapter 2

Representation of Boolean Functions

2.1 Basic properties of $\mathbf{R}^C$ (the real case)

2.1.1 Definitions and the isomorphism theorem

Throughout this subsection the field $\mathbf{F}$ is arbitrary unless we state otherwise.

For $n \geq 0$ let us denote by $C = C_n$ the n dimensional hypercube. C_n consists of the vectors $(\epsilon_1, \epsilon_2, ..., \epsilon_n)$ where $\epsilon_i \in \{0, 1\}$ for $n = 1, 2, ..., n$. We need the following definitions:

Definition 2.1.1 *For a field* $\mathbf{F}$ *we define:*

$$\mathbf{F}^C = \{f \mid f : C_n \to \mathbf{F}\}.$$

In other words $\mathbf{F}^C$ *is the algebra over* $\mathbf{F}$ *of the* $\mathbf{F}$*-valued functions on the* n*-dimensional hypercube.*

Definition 2.1.2 *The algebra-homomorphism* Φ *from* $\mathbf{F}[x_1, x_2, ..., x_n]$ *to* $\mathbf{F}^C$ *is defined by:*

$$\Phi(P)(\epsilon_1, \epsilon_2, ..., \epsilon_n) = P(\epsilon_1, \epsilon_2, ..., \epsilon_n). \tag{2.1}$$

Theorem 2.1.3 *(Smolensky []) For an arbitrary field* $\mathbf{F}$ *and for the algebra-homomorphism* Φ *from* $\mathbf{F}[x_1, x_2, ..., x_n]$ *to* $\mathbf{F}^C$ *defined above we have:*

1. Φ *is a surjection.*

2. $\mathrm{Ker}\Phi$ *is the ideal generated by the polynomials* $x_1^2 - x_1, x_2^2 - x_2, ..., x_n^2 - x_n$.

3. *Each class in* $\mathbf{F}[x_1, x_2, ..., x_n]/\mathrm{Ker}\Phi$ *can be represented unambiguously by a multilinear polynomial. (A polynomial is multilinear if it is linear in all of its variables).*

Proof: The polynomial

$$P = \prod_{i=0}^{n} (\epsilon_i x_i + (1 - \epsilon_i)(1 - x_i)).$$

vanishes on all the points of C except on $(\epsilon_1, \epsilon_2, ..., \epsilon_n)$ where $P(\epsilon) = 1$. Since these functions form a basis for $\mathbf{F}^C$ the surjectivity of Φ follows.

For proving property 2. first realize that since the polynomials $x_i^2 - x_i$ for $i = 1, 2, ..., n$ are in $\mathrm{Ker}\Phi$ so is the ideal they generate:

$$(x_1^2 - x_1, x_2^2 - x_2, ..., x_n^2 - x_n) \subset \mathrm{Ker}\Phi \tag{2.2}$$

The surjective property of Φ gives us that

$$\dim(\mathbf{F}[x_1, x_2, ..., x_n]/\mathrm{Ker}\Phi) \geq 2^n.$$

This together with $\dim \mathbf{F}[x_1, x_2, ..., x_n]/(x_1^2 - x_1, x_2^2 - x_2, ..., x_n^2 - x_n) = 2^n$ imply property 2.

Property 3 is a consequence of property 2. ∎

Definition 2.1.4 *Let us denote the set of the monomials of the form* $m_\delta = \prod_{\delta_i=1} x_i$ *by* M *where* δ *ranges over the elements of* $\{0, 1\}^n$.

We identify the set of multilinear polynomials with the set of functions $\mathbf{F}^M$ *by identifying the polynomial* $\sum_{\delta \in \{0,1\}^n} \lambda_\delta m_\delta$ *with the function in* $\mathbf{F}^M$ *that takes the value* λ_δ *on* m_δ.

Note that the multiplication in the algebra $\mathbf{F}^M$ is the multiplication inherited from $\mathbf{F}[x_1, x_2, ..., x_n]$ rather than the pointwise multiplication.

Consequence 2.1.5 *The map* Φ *induces a map* φ *from* $\mathbf{F}^M$ *to* $\mathbf{F}^C$. *The map* φ *is an algebra isomorphism.*

For a function $f \in \mathbf{F}^C$ *we denote* $\varphi^{-1}(f)$ *by* P_f. *Note that the even though* P_f *depends on the field* $\mathbf{F}$, *the formula expressing it in many cases does not (see samples below). We, therefore, do not specify the field* $\mathbf{F}$ *in the notion of* P_f. *From the context it will always be clear what is the field (fields) in question.*

Let us illustrate the correspondence between $\mathbf{F}^C$ and $\mathbf{F}^M$ with some samples.

1. For the function $AND = \bigwedge_{i=1}^{n} x_i$ we have that $P_{AND} = \prod_{i=1}^{n} x_i$.

2. For the function $OR = \bigvee_{i=0}^{n} x_i$ we have that $P_{OR} = 1 - \prod_{i=1}^{n}(1 - x_i)$.

3. We call the following function the *address function* and denote it by *ADDR*:

   The address function has $n + 2^n$ Boolean input variables $y_1, y_2, ..., y_n$ and x_ϵ for any $\epsilon \in C_n$.

   For determining the value of the address function we first look at the input setting of the variables $y_1, y_2, ..., y_n$. This determines an $\epsilon \in C_n$. The value of the function is the value of x_ϵ.

   Now $P_{ADDR} = \sum_{\epsilon \in C_n} x_\epsilon \prod_{i=1}^{n}(\epsilon_i y_i + (1 - \epsilon_i)(1 - y_i))$.

4. For a given number q we want to determine the polynomial $P_{\bmod q}$. We understand that the Boolean function mod$q$ outputs *zero* if the number of the ones in the input is congruent to *zero* modulo q and outputs *one* otherwise.

   We compute $P_{\bmod q}$ in the real case, but like in the case of other Boolean functions this gives us information about the polynomial $P_{\bmod q}$ over an arbitrary field.

   Define the polynomial Q as $Q = 1/q(x^{q-1} + x^{q-2} + ... + 1) + 1$. Q has the property that it takes each q-th root of unity into 0 except the 1, which it takes into 0. We shaw that for an elementary q-th root of unity ξ:

$$P_{\bmod q} \equiv Q(\prod_{i=0}^{n}((\xi - 1)x_i + 1)) \pmod{\mathrm{Ker}\Phi}. \tag{2.3}$$

   Indeed, on the hypercube C_n :

$$\prod_{i=0}^{n}((\xi - 1)x_i + 1) = \xi^{\sum_{i=1}^{n} x_i}. \tag{2.4}$$

   Composing the rhs. of equation2.4 with the polynomial Q we get the mod$q$ function on the hypercube. The multilinear polynomial that

corresponds to the rhs. of equation 2.3 is

$$P_{\bmod q} = \sum_{i=1}^{n} -1/q(\lambda_1^i + \lambda_2^i + ... + \lambda_{q-1}^i)\sigma_i, \tag{2.5}$$

where σ_i is the i'th elementary symmetric polynomial and $\lambda_j = \xi^j - 1$ for $j = 1, ..., q-1$. We can express the coefficients $a_i = -1/q(\lambda_1^i + \lambda_2^i + ... + \lambda_{q-1}^i)$ in either of the two following forms:

- $a_0 = 0$, $a_i = \sum_{j=0}^{\lfloor \frac{n}{q} \rfloor} (-1)^{i-qj+1} \left(\begin{array}{c} i \\ qj \end{array} \right)$ for $i = 1, ..., n$.
- $a_0 = 0$, $a_i = -1^{i+1}$ for $i = 1, ..., q-1$, and for $i \geq q$ a_i is given by the following linear recursion:

$$a_i \left(\begin{array}{c} q \\ q \end{array} \right) + a_{i-1} \left(\begin{array}{c} q \\ q-1 \end{array} \right) + ... + a_{i-q+1} \left(\begin{array}{c} q \\ 1 \end{array} \right) = 0.$$

Note that the value of a_i is independent of n.

2.1.2 Further facts

The map φ from $\mathbf{F}^M$ to $\mathbf{F}^C$ defined in subsecion 2.1.1 can be formulated explicitly.

Write the elements of $\mathbf{F}^M$ in the form $\sum_{\delta \in \{0,1\}^n} x_\delta m_\delta$ and the elements of $\mathbf{F}^C$ in the form $\sum_{\epsilon \in \{0,1\}^n} y_\epsilon f_\epsilon$. Here f_ϵ is defined as the function on C that is *zero* everywhere except in ϵ where it takes the value *one*.

The map φ and its inverse establish the following transformation on the coefficients x_δ and y_ϵ:

$$(\epsilon) \qquad \sum_{\epsilon \geq \delta} x_\delta = y_\epsilon \qquad (\epsilon \in \{0,1\}^n) \tag{2.6}$$

$$(\delta) \qquad \sum_{\epsilon \leq \delta} (-1)^{|\delta - \epsilon|} y_\epsilon = x_\delta \qquad (\delta \in \{0,1\}^n) \tag{2.7}$$

These equations hold no matter what is the field $\mathbf{F}$.

Equation (2.7) provides us with the following pieces of information about P_b.

1. The coefficients of P_b are integers.
2. The coefficients of P_b in absolute value cannot exceed 2^{n-1}

3. P_b depend only on the characteristics of the field $\mathbf{F}$.

4. The polynomial P_b for the field GF(p) where p is a prime is exactly the polynomial in the real case except the coefficients are taken modulo p.

Consider the distribution $\mathcal{D}(\nu_1, \nu_2, ..., \nu_n)$ on the points of C_n where for $i = 1, ..., n$ we make the i-th coordinate completely independently from the other coordinates equal to *one* with probability ν_i. For a function $f \in \mathbf{R}^C$ let us denote by $E_{\mathcal{D}}(f)$ the expected value of f with respect to $\mathcal{D} = \mathcal{D}(\nu_1, \nu_2, ..., \nu_n)$.

Proposition 2.1.6 *For $f \in \mathbf{R}^C$ we have $E_{\mathcal{D}}(f) = P_f(\nu_1, \nu_2, ..., \nu_n)$, where $\mathcal{D} = \mathcal{D}(\nu_1, \nu_2, ..., \nu_n)$.*

Proof. For the functions f_ϵ defined in the beginning of this subsection $P_{f_\epsilon}(\nu_1, \nu_2, ..., \nu_n) = \prod_{i=0}^{n}(\epsilon_i\nu_i + (1-\epsilon_i)(1-\nu_i)) = E_{\mathcal{D}}(f_\epsilon)$.

For every $f \in \mathbf{R}^C$ there are $\lambda_\epsilon \in \mathbf{R}$ (for $\epsilon \in C_n$) such that $f = \sum_{\epsilon \in C_n} \lambda_\epsilon f_\epsilon$. Now

$$P_f(\nu_1, \nu_2, ..., \nu_n) = P_{(\sum_{\epsilon \in C_n} \lambda_\epsilon f_\epsilon)}(\nu_1, \nu_2, ..., \nu_n) =$$

$$\sum_{\epsilon \in C_n} \lambda_\epsilon P_{f_\epsilon}(\nu_1, \nu_2, ..., \nu_n) = \sum_{\epsilon \in C_n} \lambda_\epsilon E_{\mathcal{D}}(f_\epsilon) = E_{\mathcal{D}}(f). \quad \square$$

Consequence 2.1.7 *For a Boolean function b we have $0 \leq P_b \leq 1$ if the argument of P_b is taken from the hypercube $[0,1]^n$*

2.1.3 The symmetric case

For $\epsilon \in \{0,1\}^n$ its weight $|\epsilon|$ is the number of *ones* that occur in $\epsilon = (\epsilon_1, \epsilon_2, ..., \epsilon_n)$.

A function $f \in \mathbf{F}^C$ is *symmetric* if it takes the same values on the elements of C with the same weight.

A typical example for a symmetric function is the mod$q$ function.

A symmetric function f can be written as $\sum_{i=0}^{n} \lambda_i f_i$, where f_i is the function that is *zero* everywhere except at the points of weight i where it is *one*. So e.g. the MAJORITY function is expressed as $\sum_{i=0}^{\lfloor n/2 \rfloor} f_i$.

The sequence $\lambda_0, \lambda_1, ..., \lambda_n$ is the sequence of the values it takes on the points of weight 1,2,... and n respectively.

Proposition 2.1.8 *The following four statements are equivalent over an arbitrary field $\mathbf{F}$:*

1. *$f \in \mathbf{F}^C$ is symmetric.*

2. *The polynomial P_f is symmetric.*

3. *The polynomial P_f is a linear combination of the elementary symmetric polynomials $\sigma_1, \sigma_2, ..., \sigma_n$.*

4. *$f = \Phi(P)$ for some symmetric polynomial P.*

If $f = \sum_{i=0}^{n} \lambda_i f_i$ and $P_f = \sum_{i=0}^{n} \mu_i \sigma_i$ then the vectors $\vec{\lambda} = (\lambda_0, \lambda_1, ..., \lambda_n)$ and $\vec{\mu} = (\mu_1, \mu_2, ..., \mu_n)$ can be expressed from one another by the equations $\vec{\lambda}^T = \mathbf{M}\vec{\mu}^T$ and $\vec{\mu}^T = \mathbf{M}^{-1}\vec{\lambda}^T$ where the matrix $\mathbf{M}$ and its inverse are

$$\mathbf{M} = \begin{pmatrix} \binom{0}{0} & 0 & \cdots & 0 \\ \binom{1}{0} & \binom{1}{1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \binom{n}{0} & \binom{n}{1} & \cdots & \binom{n}{n} \end{pmatrix} \tag{2.8}$$

and

$$\mathbf{M}^{-1} = \begin{pmatrix} (-1)^{0+0}\binom{0}{0} & 0 & \cdots & 0 \\ (-1)^{1+0}\binom{1}{0} & (-1)^{1+1}\binom{1}{1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^{n+0}\binom{n}{0} & (-1)^{n+1}\binom{n}{1} & \cdots & (-1)^{n+n}\binom{n}{n} \end{pmatrix} \tag{2.9}$$

If $p = \mathrm{char}(F)$ is not *zero* we understand that we take the entries matrices modulo p.

That $\mathbf{M}$ transforms $\vec{\mu}$ into $\vec{\lambda}$ follows from

$$\varphi(\sigma_i) = \sum_{j=0}^{n} \binom{j}{i} f_j \ \ (i = 0, 1, ..., n).$$

In order to see that the product of the matrices in equation (2.8) and equation (2.9) is $\mathbf{E}_n$ we consider the functions $g_i = (1 + \frac{1}{x})^i$ and $h_j = \frac{1}{1+x}(\frac{1}{1+\frac{1}{x}})^j$ for $i, j \in \{0, 1, 2, ..., n\}$. It is easy to see that the scalar product of the i'th row of $\mathbf{M}$ and the j'th column of $\mathbf{M}^{-1}$ is constant term in the *Laurent* expansion of $g_i h_j$. This verifies our claim.

Proposition 2.1.9 *For any symmetric polynomial $Q \in \mathbf{R}[x_1, x_2, ..., x_n]$ there exists a P polynomial in one variable that*

1. $Q \equiv P(x_1 + x_2 + ... + x_n) \pmod{\text{Ker}\Phi}$
2. $\deg P \leq \deg Q = k$

Proof: In the light of Proposition 2.1.8 it is enough to prove the theorem for the elementary symmetric polynomials.

We proceed by induction on the degree k.

The case $k = 0$ is obvious. For $k > 0$ the inductional hypothesis yields a polynomial P_{k-1} of degree $k - 1$ which is congruent to σ_{k-1} modulo the ideal KerΦ. One can check that

$$\sigma_{k-1} \sum_{i=1}^{n} x_i \equiv k\sigma_k + (k-1)\sigma_{k-1} \qquad \pmod{\text{Ker}\Phi}.$$

Hence the polynomial $P_k = \frac{x}{k} P_{k-1} - \frac{k-1}{k} P_{k-1}$ satisfies 1 and 2 for $Q = \sigma_k$. ∎

Definition 2.1.10 *For a function $f \in \mathbf{R}^C$ we define the symmetric function $\tilde{f} \in \mathbf{R}^C$ as follows:*

$$\tilde{f} := \frac{1}{k!} \sum_{\sigma \in S_n} f(\epsilon_{\sigma(1)}, \epsilon_{\sigma(2)}, ..., \epsilon_{\sigma(n)}).$$

We call the above transformation symmetrization.

For a polynomial $P \in \mathbf{R}[x_1, x_2, ..., x_n]$ we can also define:

$$\tilde{P} := \frac{1}{k!} \sum_{\sigma \in S_n} P(x_{\sigma(1)}, x_{\sigma(2)}, ..., x_{\sigma(n)}).$$

For any $f \in \mathbf{R}^C$ we have that $P_{\tilde{f}} = \tilde{P}_f$.

2.2 The L_∞ approximation scheme

Lemma 2.2.1 *Suppose the graph of a polynomial P above the interval $[x_1, x_2]$ is in the box $[x_1, x_2] \times [y_1, y_2]$. Suppose moreover that $\sup\{P'(x) \mid x \in [x_1, x_2]\} \geq D$. Then $\deg P \geq \sqrt{\frac{D|x_2 - x_1|}{|y_2 - y_1|}}$*

The proof of this lemma uses the well known fact from analysis that among those polynomials of degree d that are not greater in absolute value than 1 in the interval $[0, 1]$ the derivative of Chebisev polynomials is the largest in L_∞ norm. From this the lemma easily follows.

Lemma 2.2.2 *Suppose for a single variable polynomial P over R the following conditions hold:*

1. *There is a point x in $[0, 1]$ where $P'(x) \geq C$*
2. *For $i \in \{0, 1, ..., k\}$ $0 \leq P(i) \leq 1$.*

Then $\deg P \geq \lfloor \sqrt{Ck} \rfloor$.

Proof: First inductively construct the sequence of polynomials $P_k, P_{k+1}, P_{k+2}, ...$ as follows:

$$P_k = P; \text{ For } l > k P_l(x) = \frac{x}{n+1} P_{l-1}(x-1) + \frac{n-x+1}{n+1} P_{l-1}(x).$$

The following statements can be proven easily by induction on l:

1. For $l = k, k+1, ...$ $degP_l = k$.
2. For $l = k, k+1, ...$ and for $i = 0, 1, ..., l$ $|P_l(i)| \leq 1$
3. $P_l(0) = 0$; $P_l(1) \geq Ck/l$.
4. There is a point $x \in [0, 1]$ for which $P'_l(x) \geq Ck/l$.

We prove that for any δ the inequality $\deg P \geq \sqrt{Ck - \delta}$ holds. Chose $\varepsilon = \delta/2Ck$ and $l = \lceil 2(Ck)^{\frac{3}{2}}(1 + 2\varepsilon)/\varepsilon$. If P_l never goes out of the box $[0, l] \times [-1-\varepsilon, 1+\varepsilon]$ on the interval $[0, l]$ then we get contradiction with 2.2.1 or 4.

If P_l intersects the boundary of the above box at least $2\sqrt{Ck}$ times then P_l takes one of the value $1 + 2\varepsilon$ at least $\sqrt{Ck}$ times or else it takes the value

$1+2\varepsilon$ at least $\sqrt{Ck}$ times. Since P is non-constant the statement of the theorem is settled in this case.

We are left with the case when P_l intersects the boundary of the box $[0,l]\times[-1-\varepsilon,1+\varepsilon]$ at most $2\sqrt{Ck}$ times. The longest distance between the x-coordinates of two neighboring intersections (or an intersection and 0 or l) should be at least $Ck(1+2\varepsilon)/\varepsilon$. Suppose these two intersections are x_1 and x_2. Again, we can use Lemma 2.2.1 for the box $[x_1,x_2]\times[-1-\varepsilon,1+\varepsilon]$ since we can easily get a $|P_l'(x_0)|\geq\varepsilon$ estimation from 2 of the derivative of P_l either for some $x_0\in[x_1,\lceil x_1\rceil]$ or for some $x_0\in[\lfloor x_2\rfloor,x_2]$ using the theorem of Lagrange. □

For a function f in $\mathbf{R}^C$ its L_∞ norm is defined as $\max\{|f(\epsilon)|\mid\epsilon\in C_n\}$.

Theorem 2.2.3 *Suppose that for the Boolean function b on n variables the value that it takes on the point $(0,..,0)$ differs from the value it takes at any point of weight 1. Suppose moreover, that for a function $f\in\mathbf{F}^C$ we have that $|f-b|_\infty\leq 1/3$. Then the degree of the polynomial P_f is at least $1/5\sqrt{n}$*

Proof: We can suppose that $b(0,..0)=0$ and therefore b outputs *one* on all the inputs of weight *one*. The polynomial $\tilde{P}_f$ has the following properties:

1. $\tilde{P}_f(0,..,0)\leq 1/3$
2. $\tilde{P}_f(\epsilon)\geq 2/3$ if $|\epsilon=1$
3. $-1/3\leq\tilde{P}_f(\epsilon)\leq 4/3$ for any $\epsilon\in C_n$

Since $\tilde{P}_f$ is symmetric according to Proposition 2.1.9 there exists a polynomial P with $\deg P\leq\deg\tilde{P}_f$ such that P and $\tilde{P}_f$ agree on C_n. Applying Lemma 2.2.2 to the polynomial $3/5(P(x)-P(0))$ we obtain that

$$1/5\sqrt{n}\leq\deg P\leq\deg\tilde{P}_f=\deg P_f.\quad\square$$

For an $\epsilon=(\epsilon_1,\epsilon_2,...,\epsilon_n)\in C_n$ and for a set of coordinates $B\subset\{0,1,...,n\}$ we call the 0-1 vector ϵ_B which differs from ϵ exactly in those coordinates that belong to B.

Definition 2.2.4 *(Noam...) For a Boolean function b and for an input ϵ we define the* block-sensitivity *$S_\epsilon(b)$ of the function b on* input *ϵ as follows:*

$$S_\epsilon(b)=\max\{k\mid\exists B_1,B_2,...,B_k\subset\{0,1,...,n\}:$$
$$B_i\bigcap B_j=\emptyset\,(1\leq i\leq j\leq k);\ b(\epsilon)\neq b(\epsilon_{Bi})\,(1\leq i\leq k)\}$$

The block-sensitivity of a Boolean function b is defined as $S(b)=\max\{S_\epsilon(b)\mid\epsilon\in C_n\}$.

Theorem 2.2.5 *Suppose for a function $f \in \mathbf{R}^C$ there exists a Boolean function b with $|f - b|_\infty \leq 1/3$. Then* $\deg P_f \geq 1/5\sqrt{S(b)}$.

Proof Suppose the point ϵ is such that $S_\epsilon(b) = S(b) = k$ and the corresponding blocks are $B_1, B_2, ..., B_k$. From $b(x_1, x_2, \ldots, x_n$ we construct a Boolean function $b'(y_1, y_2, \ldots, y_k)$ as follows:

$$\text{We replace } x_i \text{ in } f \text{ by } \begin{cases} \epsilon_i & \text{if } \; i \notin \bigcup_{j=1}^k B_j \\ y_j & \text{if } \; i \in B_j \text{ and } \epsilon_i = 0 \\ 1 - y_j & \text{if } \; i \in B_j \text{ and } \epsilon_i = 1 \end{cases}$$

We obtain the function f' from the function f in the same way. One can check that the assumptions of Theorem 2.2.5 hold for b' and f'. The application of Theorem 2.2.5 and $\deg f' \leq \deg f$ complete the proof. ∎

2.3 The Fourier transform approach

For a group G and for a field F let us define $G(\mathbf{F})$ as the associative algebra of the formal sums $\sum_{g \in G} \lambda_g g$, where the algebra-elements $g_i \in G$ and $g_j \in G$ are multiplied as in G.

The following theorem which is well known from the representation theory of the finite abelian groups describes the structure of $G(\mathbf{F})$ in the case when G is finite abelian and the field F contains an m'th root of unity where m is the exponent of the group G. The exponent m of a group G is the l.c.m. of the orders of the elements of G.

Theorem 2.3.1 *Let G be a finite abelian group with $|G| = n$ and F a field which contains an m'th root of unity where $m = m(G)$ is the exponent of G. Then there is an map ϕ from $G(\mathbf{F})$ to F^n where the i'th copy of F corresponds to the i'th character of G and for $g \in G$ $\phi(g) = (\chi_1(g), \chi_2(g), ..., \chi_n(g))$. This map is an algebra isomorphism.*

In the following discussion G and F will always be like in Theorem 2.3.1. We identify F^n and F^C where C is the set of the characters of G over F. We also identify F^G with the elements of $G(\mathbf{F})$. We call the map ϕ Fourier transform.

For a function $f \in F^G$ supp f denotes $\{g \in G | f(g) \neq 0\}$.

Jeff Kahn and Roy Meshulam in [] proved the following relation between $|\text{supp} f|$ and $|\text{supp}\phi(f)|$:

Theorem 2.3.2 *If $f \in F^G$ is not identically 0, then*

$$|suppf||supp\phi(f)| \geq |G| \tag{2.10}$$

Below we give an argument to this fact that does not use the structure theorem for the finite abelian groups.

Proof: Let us denote by $\mathcal{I}$ the ideal generated by f in F^G, and let $\mathcal{I}^{'} = \phi(\mathcal{I})$. Clearly $\dim \mathcal{I}^{'} = \dim \mathcal{I} = \text{supp}f$. We can suppose that $\phi(f) = \sum_{i=1}^{k} \mu_i \chi_i$, where $0 \neq \mu_i \in F$ for $i = 1, ..., k$.

Since $k = \text{supp}\phi(f)$, what we have to prove is that $k \dim \mathcal{I}^{'} \geq |G|$.

Since the elements $\chi_1, \ldots, \chi_n$ linearly span the algebra F^C, the elements $\chi_1\phi(f), \ldots, \chi_n\phi(f)$ linearly span $\mathcal{I}^{'}$. Chose indices $j_1, \ldots, j_l$ Such that $\chi_{j_1}\phi(f), \ldots, \chi_{j_l}\phi(f)$ independently linearly span $\mathcal{I}^{'}$. Suppose kl were less than n. Then there would be a character χ which occurs with *zero* coefficient in every element of $\mathcal{I}^{'}$ since only the characters $\chi_s\chi_{j_t}$ for $s = 1, .., k$; $t = 1, .., l$ can occur with *non-zero* coefficient in some element of $\mathcal{I}^{'}$.

This contradicts to the fact that in $\chi\chi_1^{-1}\phi(f) \in \mathcal{I}^{'}$ χ occurs with the coefficient $\mu_1 \neq 0$.

The theorem follows from $j = \dim \mathcal{I}^{'}$. ∎

2.4 The embedding into abelian groups

2.4.1 The Three Notions of Degree

Consider the following system of equations over the elements of an abelian group A:

$$(\epsilon) \qquad \sum_{\delta \leq \epsilon} x_\delta = y_\epsilon \qquad (\epsilon \in C_n) \tag{2.11}$$

The set of vectors $\{(y_\epsilon)_{\epsilon \in C_n} \mid y_\epsilon \in A\}$ and the set of vectors $\{(x_\delta)_{\delta \in D_n} \mid x_\delta \in A\}$ form abelian groups with respect to the pointwise addition. We denote these abelian groups by $\mathcal{A}$ and $\mathcal{A}^*$ respectively. There is a homomorphism φ from $\mathcal{A}^*$ to $\mathcal{A}$ established by (2.11).

The system

$$(\delta) \qquad \sum_{\delta \geq \epsilon} (-1)^{|\delta - \epsilon|} y_\epsilon = x_\delta \qquad (\delta \in D_n) \tag{2.12}$$

gives us a homomorphism ψ from $\mathcal{A}$ to $\mathcal{A}^*$ and an easy calculation shows that $\varphi \circ \psi = id_{\mathcal{A}}$; $\psi \circ \varphi = id_{\mathcal{A}^*}$.

Proposition 2.4.1 *The homomorphisms $\varphi : \mathcal{A}^* \to \mathcal{A}$ and $\psi : \mathcal{A} \to \mathcal{A}^*$ are an abelian group isomorphisms and inverses one of another.*

Definition 2.4.2 *For $\alpha \in \mathcal{A}$ define:*

$$\deg \alpha = \max\{|\delta| \mid x_\delta = (\phi(\alpha))_\delta \neq 0\}.$$

For a given d the elements of $\mathcal{A}$ with degree at most d form a sugroup of $\mathcal{A}$ that we call $\mathcal{A}_d$. Note that if the abelian group A is the additive structure of a field F and *alpha* $\in \mathcal{A}$ then *deg*α is exactly the degree of the polynomial P_α.

Definition 2.4.3 *If $\beta \in \mathcal{A}$ is* nonzero *when a Boolean function b is* one *and* zero *when b is* zero *then we say that β* represents *b. We define:*

$$\deg_H b = \min\{\deg \beta \mid \beta \textit{ represents } b\}.$$

Definition 2.4.4 *For a set $H \subset C_n$ $d = \deg H$ denotes the smallest d such that*

$$(\epsilon) \qquad \sum_{\delta \leq d; \delta \leq \epsilon} x_\delta = y_\epsilon \qquad (\epsilon \in H) \tag{2.13}$$

has solution for any $\vec{y}$ from $\{(y_\epsilon)_{\epsilon \in H} \mid y_\epsilon \in A\}$. This is equivalent to saying that any $\gamma \in A^H$ has an extension in $\mathcal{A}_d$

Proposition 2.4.5 *For a set $H \in C_n$ $\log |H| \geq \deg H \geq k$, where k is the largest integer such that*

$$\binom{n}{0} + \binom{n}{1} + ... + \binom{n}{k} \leq |H|$$

.

Proof: For the lower bound we realize if an equation over an abelian group like in 2.13 is always solvable then the number of the variables should be at least as big as the number of equations. We omit the proof of the upper bound which is an easy induction on n. ∎

For the sets in the following two samples we get as small degree as Proposition 2.4.5 allows:

1. We want to determine the degree of the set $S_d = \{\epsilon \in C_n \mid |\epsilon| \leq d\}$. Consider the equations:

$$(\epsilon) \qquad \sum_{\delta \leq \epsilon} x_\delta = y_\epsilon \qquad (\epsilon \in C_n; |\epsilon| \leq d) \tag{2.14}$$

and

$$(\delta) \qquad \sum_{\delta \geq \epsilon} (-1)^{|\delta - \epsilon|} y_\epsilon = x_\delta \qquad (\delta \in D_n; |\delta| \leq d). \tag{2.15}$$

Similarly to the systems (2.11) and (2.12) we can verify that the systems (2.14) and (2.15) are equivalent in the sense that they are satisfied by the same $(\vec{x}, \vec{y})$ pairs. This verifies that $\deg S_d = d$.

2. For an arbitrary number $0 \leq d \leq n$ we are interested in $\deg C_n \setminus S_d$. First notice that for any $\alpha \in \mathcal{A}$ which is constant in the subcube $\overline{x}_1 \wedge \overline{x}_2 \wedge \ldots \wedge \overline{x}_k$ and *zero* anywhere else $\deg \alpha = k$. Now from the previous sample we can derive that $\deg C_n \setminus S_d = n - d$.

Proposition 2.4.6 *Suppose the abelian group A is the additive group of a field F. Suppose furthermore that b a boolean function and for the sets H_0 and H_1 $H_0 \subset b^{-1}(0)$ and $H_1 \subset b^{-1}(1)$. Then*

$$\deg H_0 \bigcup H_1 \leq \max\{\deg H_0\,, \deg H_1 + \deg_{(H_0 \bigcup H_1)} b\}.$$

Proof: Define: $\deg H_0 = d_0$; $\deg H_1 = d_2$; $\deg_{(H_0 \bigcup H_1)} b = d_2$. Suppose $\beta \in \mathcal{A}$ represents b and $\deg \beta = d_2$ Let γ be an arbitrary function in $A^{H_1 \bigcup H_2}$. It is enough to prove that there exist $\alpha \in \mathcal{A}$ with $\deg \alpha \leq d_0$ and $\zeta \in \mathcal{A}$ with $\deg \zeta \leq d_1$ such that $\alpha + \zeta\beta$ coincides with γ on $H_0 \bigcup H_1$.

By definition there exist an $\alpha \in \mathcal{A}_{d_0}$ such that $\gamma|_{H_0} = \alpha|_{H_0}$. Find $\zeta \in \mathcal{A}_{d_1}$ such that ζ agrees with $\frac{\gamma - \alpha}{\beta}$ on H_1.

Only the expression above requires some explanation. Since A is the additive structure of the field F, division by non zero elements make sense in A. The function constructed above by pointwise division is defined everywhere in H_1 since β is *non-zero* on H_1. ∎

Theorem 2.4.7 *Let A be the additive structure of a field F. Let the b boolean function be the* majority *function (i.e. the function that takes* zero *on $S_{\lfloor \frac{n}{2} \rfloor}$ and it takes* one *otherwise). If $|V| = 2^{n-1} - \Omega(2^n)$ then $\deg_{(C_n \setminus V)} b = \Omega(\sqrt{n})$.*

Proof: Let us denote the set $S_{\lfloor \frac{n}{2} \rfloor} \setminus V$ by H_0 and the set $C_n \setminus (S_{\lfloor \frac{n}{2} \rfloor} \bigcup V)$ by H_1. We have the following estimations:

1. $\deg H_0 \leq \deg S_{\lfloor \frac{n}{2} \rfloor} = \lfloor \frac{n}{2} \rfloor$
2. $\deg H_1 \leq \lfloor \frac{n}{2} \rfloor$
3. $\deg(H_0 \bigcup H_1) \geq \frac{n}{2} + \Omega(\sqrt{n})$

Items 1 and 2 comes from the monotonicity of the degree of the sets and from samples 1 and 2. For proving item 3 we notice that $|H_0 \bigcup H_1| \geq 2^{n-1} = \Omega(2^{n-1})$ and we use Proposition 2.4.5 and basic facts about the binomial distribution. Now if we use Proposition 2.4.6 for the sets H_0 and H_1 and the Boolean function b, the above estimations on the degrees of H_0, H_1 and $H_0 \bigcup H_1$ immediately give the $\Omega(\sqrt{n})$ lower bound on $\deg_{C_n \setminus V} b$. ∎

Theorem 2.4.8 *Let A be an arbitrary but fixed finite abelian group. Let the b boolean function be the* majority *function If $|V| = o(2^n)$ then* $\deg_{C_n \setminus V} b = \Omega(\sqrt{n})$.

We omit the proof.

2.4.2 AC^0 circuits with a mod6 gate on top

Theorem 2.4.9 *Suppose that $\mathcal{C}_n$ computes the majority function b_n of n Boolean variables and:*

1. *The input gates are variables or their negations.*
2. *The output gate is a* mod6 *gate.*
3. *The gates in $\mathcal{C}_n$ other than the output gate are unbounded fan-in $\wedge$ and $\vee$ gates.*
4. *$\mathcal{C}_n$ is of depth d for a constant d independent of n.*

Then $S(\mathcal{C}_n) \geq 2^{n/2d}$, where $S(\mathcal{C}_n)$ is the size of $\mathcal{C}_n$.

Proof Let us denote the subcircuits of $\mathcal{C}_n$ by $\mathcal{D}_1$,$\mathcal{D}_2$,...,$\mathcal{D}_k$.

Denote the output of $\mathcal{D}_i$ $(i = 1, \ldots, k)$ by y_i. By the definition of the circuit $\mathcal{C}_n$ we have that $\sum_{i=1}^{k} y_i \equiv 0 \pmod 6$ if the number of the ones in the input is less or equal than $\lfloor \frac{n}{2} \rfloor$ and $\sum_{i=1}^{k} y_i \not\equiv 0 \pmod 6$ otherwise. This implies that for the sets of inputs $H_2 := \{\epsilon \mid \sum_{i=1}^{k} y_i \not\equiv 0 \pmod 2\}$

and $H_3 := \{\epsilon \mid \sum_{i=1}^{k} y_i \not\equiv 0 \pmod 3\}$ the equation $H_2 \bigcup H_3 = C_n \setminus S_{\lfloor \frac{n}{2} \rfloor}$ holds. Since $|C_n \setminus S_{\lfloor \frac{n}{2} \rfloor}| = 2^{n-1}$ either $|H_2|$ or $|H_3$ is at least of size 2^{n-2}. W.l.g. we can suppose that it is H_3. Appling Theorem 2.4.7 for $S_{\lfloor \frac{n}{2} \rfloor} \bigcup H_3$ we get that the function $\sum_{i=1}^{k} y_i \pmod 3$ cannot even be approximated by a polynomial of degree $o(\sqrt{n})$ over $GF(3)$. From the paper [] of Smolensky we know that this implies a $2^{n/2d}$ size lower bound. □

Chapter 3

Communication Schemes

3.1 Introduction

The notion of the communication complexity was first introduced by H. Abelson [1] in 1978 and was designed to measure the information transfer among processors that cooperate in solving a computational problem.

The definition that we use now goes back to the STOC paper of A. Yao [87] in 1979.

The area has important applications outside Theoretical computer science: perhaps the most important is that it provides a tool to prove tradeoffs between area and number of cycles for synchronous VLSI chips. See Lighton [] for details.

By now communication complexity theory has become a frequently studied branch of the complexity theory with interesting and nontrivial techniques using linear algebra and combinatorics. There are proofs of explicit lower bounds on communication complexity. These results hold a promise of application to other areas of the theory of computation. A recent overwiew of the topic is the survey article of L. Lovász [52].

Several variants of the original model have been defined like the nondeterministic version introduced by Lipton and Sedgewick [] or the multiparty version studied by Chandra Furst and Lipton [23]

In the model for deterministic communication complexity two parties try to jointly determine the value of a Boolean function f. Both of them have infinite computing power, but each of them has access only to half of the input bits.

They send *messages* one to another. The players know the function f and

they also agree upon a *protocol* beforehand that direct their communication.

The object of the game is to compute the particular value of f using as small number of bits of communication as possible.

Before putting the above notion into a mathematically more precise form, let us give an example of a very simple communication game: The Boolean function in question is the mod2 of $2n$ Boolean variables $x_1, \ldots, x_{2n}$. The two parties — let us call them Alice and Bob — agree to the following protocol:

The protocol consists of a single round: Alice first computes the modulo two sum of the bits she has and send it over to Bob. The game then finishes.

Note that Bob is able to compute the modulo two sum of all the $2n$ bits relying only on the information sent by Alice and the part of the input what he possesses.

The sequence of the bits they get is not arbitrary: Alice always gets the bits $x_1, \ldots, x_n$ and Bob always gets the bits $x_{n+1}, \ldots, x_{2n}$.

We say that the *length* of the above protocol is one bit, the bit that Alice has to send. We disregard the effort that Alice and Bob have to make in order to add the digits modulo two.

Formally a deterministic protocol is a pair of functions (A, B) which tell to player one (Alice) and player two (Bob) what message is to send next. We suppose that the messages have a special format, namely a 0-1 string followed by an & symbol. The & symbol at the end of the message is required so that the next party knows when the message terminates. Later when we compute the length of the messages we take into account only the length of the 0-1 sequence and disregard the one extra symbol at the end.

Definition 3.1.1 *A protocol P is a pair of functions (A, B) and a set STOP. $A, B : \{0, 1, \&\}^* \to$ MESSAGES. The set MESSAGES is the regular set $(0, 1)^*\&$; STOP is a subset of $\{0, 1, \&\}^*$ with the property that $w \in$ STOP implies that for any word $w' \in \{0, 1, \&\}^*$ we have $ww' \in$ STOP.*

For an $(\mathbf{x}, \mathbf{y})$ pair the the sequence of the messages $M_0, M_1, \ldots$ produced by the protocol P is defined by $M_0 = A(x)$; $M_{2k+1} = B(\mathbf{y}\&M_0M_1 \ldots M_{2k})$ (for $k = 0, 1, \ldots$); $M_{2k} = A(\mathbf{x}\&M_0M_1 \ldots M_{2k-1})$ (for $k = 1, 2, \ldots$).

The messages $M_0, M_2, \ldots$ are are sent by the first player (Alice) and the messages $M_1, M_3, \ldots$ are sent by the second player (Bob). The parties stop sending messages as soon as the concatenation of the sequence of the messages sent over the communication chanel by both parties first belongs to the set STOP. The above definition does not exclude the possibility that

the parties exchange messages without reaching an end. Naturaly when we design a protocol for a Boolean function f we do not want to allow this to happen for any pairs of input $\mathbf{x}, \mathbf{y}$. The following definition is crucial in telling us when we consider a protocol appropriate in computing the value of a Boolean function f.

Let C_i be the concatenation of the messages that the parties send each other upto the i^{th} round. Formally for a pair of inputs $(\mathbf{x}, \mathbf{y})$ we define the sequence $C_0, C_1, \ldots$ for $i = 0, 1, \ldots$ where $C_i = M_0 M_1 \ldots M_i$ $(i = 0, 1, \ldots)$. We call the elements of the sequence C_i communication.

Definition 3.1.2 *The protocol is* finite *if for every pair* $(\mathbf{x}, \mathbf{y})$ *there exists a natural number* $i(\mathbf{x}, \mathbf{y})$ *such that for any* $j \geq i(\mathbf{x}, \mathbf{y})$ *the sring* C_j *belongs to the set STOP.*

The length of a finite protocol P is the smallest number c that every communication of length at least c is in STOP. When computing the length of a communication C we do not take the & symbols into account.

Definition 3.1.3 *We say that a finite protocol* P *is good for a boolean function* $b(\mathbf{x}, \mathbf{y})$ *if:*

1. *If a word* $w \in STOP$ *is a* C_i *for two different input pairs* $(\mathbf{x}, \mathbf{y})$ *and* $(\mathbf{x}, \mathbf{y}')$ *and* i *is odd then* $b(\mathbf{x}, \mathbf{y}) = b(\mathbf{x}, \mathbf{y}')$.

2. *If a word* $w \in STOP$ *is a* C_i *for two different input pairs* $(\mathbf{x}, \mathbf{y})$ *and* $(\mathbf{x}', \mathbf{y})$ *and* i *is even then* $b(\mathbf{x}, \mathbf{y}) = b(\mathbf{x}', \mathbf{y})$.

Let us notice that in the previous definition item 1 corresponds when Alice knows the value of f and item 2 corresponds when Bob knows the value of f. Since the set STOP depends only on the messages exchanged thru the communication chanel in both cases the other player is aware that the other player knows the value of f. Later we give another definition to a protocol being effective in computing a function when the last bit exchanged should be the value of f. The notion what we use above is the one that Lovász suggests using in his survey [52], but the two different notions only slightly differ since in an extra round the player possessing the knowledge of the value of f can anounce it.

Definition 3.1.4 *The communication complexity of a Boolean function* $b(\mathbf{x}, \mathbf{y})$ *is the length of the shortest protocol that is good for* b.

In the following sections we take a closer look at various types of communication schemes.

We define the symmetric communication complexity of a Boolean function f as the maximal communication complexity of f where the maximum is taken over all subdivisions of the input into two parts. In section **??** we succeed giving a complete characterization of the functions with bounded symmetric communication complexity.

Noam Nisan in 1988 discovered a whole new area of trading hardness for pseudo-randomness. It turnes out that complexity lower bounds of the so called k-ways communication complexity when k players try to evaluate the value of a Boolean function with k input stings the i^{th} of which is written on the i^{th} player forehead can be effectively turned into pseudorandom-generators for Logspace mashines. The lower bounds for the k-ways communication complexity of certain functions given in the paper of Babai, Nisan and Szegedy [13] turne out to be applicable in giving lower bounds for different types of branching programs as well as proving time space tradeof for muliple head Turing mashines. In details this can be found in section **??**.

The newly introduced communication scheme by Gereb-Graus is a very interesting experiment to transtate the phenomena of information compressing that takes place in the Boolean circuits into a purely communicational problem. An and/or gate in a Boolean circuit 'compresses' two less useful pieces information into a shorter, but more useful one. Unlike in the case of previos schemes no explicit function was found which is provably hard, and the problem looks rather complicated. We suspect that proving such bounds might lead to nontrivial bounds in circuit complexity. In section **??** we shaw that a random function has high communication complexity when using the scheem of Gereb-Graus, that somewhat analogous to the situation that proving that a random function has high circuit complexity is much easier than finding an explicite function.

3.2 The multiparty games

3.2.1 The concept of k-ways communication

Chandra, Furst and Lipton in [] introduce a version of the communication game for the purposes of getting lower bounds on bounded width branching programs. Their scheme is a generalization of the two ways communication scheme.

Let $g(x_1, \ldots, x_k)$ be a Boolean function that k parties wish to collaboratively evaluate. The i^{th} party knows each input argument except x_i; and each party has unlimited computational power. They share a blackboard, viewed by all parties, where they can exchange messages. The objective is to minimize the number of bits written on the board.

Note that if each x_i has n bits then n bits of communication obviously suffice. (Player #1 prints x_2 on the board and player #2 announces the result.)

The function in their paper is the EXACTLY p function, i.e. $g(x_1, \ldots, x_k) = 1$ if and only if when $x_1 + \ldots + x_k = p$ where $x_1, \ldots, x_k$ are numbers in the $[1, N]$ interval. Chandra at al. in their paper prove that worst-case upper and lower bounds on k-ways communication complexity of certain types of functions can be translated into problems of Ramsey theory.

Nisan in 1988 discovered a number of new relations between complexity problems and the problem of constructing effective pseudorandom generators for various types of computational models. He also noticed new ways of gaining lower bounds for different computational models making use of k-ways communication complexity. Inthe paper with him and Laszlo Babai [] we prove a lower bound of $\Omega(n2^{-k})$ for the minimum number of bits to be exchanged on an average instance of certain functions g if the parties wish to have a 1% advantage at guessing the value of g on random inputs. As a consequence, we construct a pseudorandom sequence of length n from a random seed of length $\exp(c\sqrt{\log n})$ that a given logspace-machine will not be able to distinguish from truly random sequences. From this result we deduce an explicit traversal sequence of length $\exp(\exp(c\sqrt{\log n}))$ for arbitrary undirected graphs on n vertices. We presen our technique in subsection **??** where we prove lower bounds on k-ways communication complexity where the function in question is the quadratic character modulo the prime p of $x_1 + \ldots + x_k$ where $0 \leq x_i \leq p-1$. The proof uses character sum estimates.

Below we give the precise definitions of *k-ways protocol*, *k-ways communication complexity* and the *ε-distributional complexity* that we intend to work with.

Definition 3.2.1 *The protocol P for the k ways communication is a set of functions $(A_1, A_2, \ldots, A_k)$ and a set STOP. $A_1, A_2, \ldots, A_n : \{0, 1, \&\}^* \to$ MESSAGES. The set MESSAGES is the regular set $(0, 1)^*\&$; STOP is a subset of $\{0, 1, \&\}^*$ with the property that $w \in$ STOP implies that for any word $w' \in \{0, 1, \&\}^*$ we have $ww' \in$ STOP. The protocol P also contains a function $f : \{0, 1, \&\}^* \to \{1, \ldots, k\}$.*

For a k-tuple$(\mathbf{x}_1, x_2, \ldots, x_n)$ *the the sequence of the messages* $M_0, M_1, \ldots$ *produced by the protocol* P *is defined by* $M_0 = A_1(x)$*;*

$M_i = A_j(\mathbf{x}_1 \& \ldots \mathbf{x}_{j-1} \& \mathbf{x}_{j+1} \& \ldots \mathbf{x}_k \& M_0 M_1 \ldots M_{i-1})$ *(for* $k = 1, 2, \ldots$*),*

where $j = f(M_0 M_1 \ldots M_{i-1})$*.*

Notce that the definition is slightly different from the one that Chandra, Furst and Lipton use in [].

In the introduction of the present chapter we defined when we consider a protocol *good* for Boolean function $f(\mathbf{x}, \mathbf{y})$. Instead of generalizing this notion for the k-ways case we give the definition of a protocol *yielding* a Boolean function $f(x_1, \ldots, x_k)$. The only difference is that in the later case the player knowledgeable with the result should announce it. We do this in order to be consistent with the notions of the paper of Babai, Nisan, Szegedy [].

Definition 3.2.2 *We say that a* k*-ways communication protocol* P *yields a function* $g(\mathbf{x}_1, \ldots, \mathbf{x}_k$ *if for any* k*-tuple* $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ *the elements of the sequence* $C_i = M_0 M_1 \ldots M_i$ *(*$i = 0, 1, 2, \ldots$*) belong to STOP from some index* $i(\mathbf{x}_1, \ldots, \mathbf{x}_k)$*. Moreover the last three digits in* $C_{i(\mathbf{x}_1, \ldots, \mathbf{x}_k)}$ *are* $\&g(\mathbf{x}_1, \ldots, \mathbf{x}_k)\&$*. The length of the protocol* P *is* $\max(i(\mathbf{x}_1, \ldots, \mathbf{x}_k))$ *where the maximum is taken over all the* k*-tuples* $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$*.*

Definition 3.2.3 *The communication complexity of a Boolean function* $b(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ *is the length of the shortest* k*-ways protocol that yields* b*.*

Definition 3.2.4 *We define the* ε*-distributional complexity of a boolean function* $b(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ *as the shortest length of a* k*-ways protocol that yields a function* g *with the property that* $Prob(g = f) \geq 1/2 + \varepsilon$*. We denote the* ε*-distributional complexity of* b *by* $C_\epsilon(b)$*.*

3.2.2 Lower bound techniques

A cylinder in the i^{th} coordinate in the Cartesian product $A_1 \times \cdots \times A_k$ is the Cartesian product of A_i and of an arbitrary subset of the Cartesian product of the remaining $k - 1$ of the A_j.

The collection of the k-tuples of the form $(x_1, \ldots, x_k)$ can be looked as a Cartesian product of the sets $A_1, \ldots, A_n$, where A_i the collection of the possible values of x_i.

The lower bound argument both in Chandra Furst Lipton [] and in Babai, Nisan, Szegedy [] use the following observation:

Proposition 3.2.5 *Let $b(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is a Boolean function and let P be a k-ways communication protocol. Let $X(C)$ be the set of those k-touples that result a communication $C = C_i$. We claim that $X(C)$ is an intersection of cylinders in the k different coordinates.*

Proof: First notice that a set X is a cylinder-intersection set if and only if $\{(\mathbf{x}'_1, \ldots, \mathbf{x}_k), \ldots, (\mathbf{x}_1, \ldots, \mathbf{x}'_k)\} \subseteq X$ implies that $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in X$ We prove by induction on i that the above property holds for $X(C_i)$. For $i = 0$ the statement is straightforward since $X(C_0)$ is the set of all the k-tuples. For $i > 0$ suppose that $\{(\mathbf{x}'_1, \ldots, \mathbf{x}_k), \ldots, (\mathbf{x}_1, \ldots, \mathbf{x}'_k)\} \subseteq X(C_i)$. Then $X(C_{i-1}) \supseteq X(C_i)$ and the inductional hypothesis together yields us that $(\mathbf{x}_1, \ldots, \mathbf{x}_k) \in X(C_{i-1})$. Let j be $f(C_{i-1})$ i.e. the index of the party that sends the i^{th} message (for the definition of f see the previous section). Then the k-tuples $(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ and $(\mathbf{x}_1, \ldots, \mathbf{x}'_j, \ldots, \mathbf{x}_k)$ result the same message in the i^{th} step. Now from $(\mathbf{x}_1, \ldots, \mathbf{x}'_j, \ldots, \mathbf{x}_k) \in X(C)$ we get that $(\mathbf{x}_1, \ldots, \mathbf{x}_j, \ldots, \mathbf{x}_k) \in X(C)$ ∎

As a consequence of lemma **??** we get that if P is a valid protocol that yield a Boolean funcion f then the set $A_1, \ldots, A_n$ can be written as the disjoint union of monochromatic cylinder-intersections where each cylinder-intersection corresponds to some running of protocol P. This observation leads us to the following lemma:

Lemma 3.2.6 *Suppose that for a Boolean function b the smallest number of the monochromatic cylinders that disjointly cover the space $A_1 \times \cdots \times A_k$ is x. Let us denote the size of $A_1 \times \cdots \times A_n$ by a. Then we have:*

$$C(b) \geq \log \frac{a}{x}$$

What can we say if the protocol P yields a function g which only approximates the function f in the sense that $Prob(f = g) \geq 1/2 + \varepsilon$? It turnes out that it is still possible to get lower bound on $C_\varepsilon(f)$ just by simply getting estimation on the distribution of b on the cylinder-intersection sets. Toward this goal let us define the discrepancy of a Boolean funcion f.

Definition 3.2.7 *Let f be a Booean function on $A_1 \times \cdots \times A_n$. The* discrepancy $\Gamma(f)$ *of f is defined as*

$$\Gamma(f) = \max \left\{ S \mid \frac{|S \cap f^{-1}(1)| - |S \cap f^{-1}(0)|}{a} \right\}.$$

Here a denotes the size of $A_1 \times \cdots \times A_n$.

Now we are ready to give the lower found on $C(f)$.

Lemma 3.2.8 *For any Boolean function f we have*

$$C_\varepsilon(f) \geq \log \frac{\varepsilon}{\Gamma(f)}.$$

Proof: Consider the bias a certain protocol achieves on f. The bias is certainly bound on from above by the biases achieved on the various components corresponding to different communnication patterns. The bias achieved on any component is bound from above by $\Gamma(f)$, so in order to achieve the total bias of ϵ, a total of $\epsilon/\Gamma(f)$ different communication patterns are needed. The total number of bit exchanges must be the logarithm of the number of communication patterns.

In the next subsection we prove upper bound for $\Gamma(f)$ on a particular function f.

3.2.3 A Character Sum Estimate

Let p be an odd prime and $\mathbf{F}_p$ the field of p elements. A *multiplicative character* over $\mathbf{F}_p$ is a homorphism tweenom the multiplicative groups $\mathbf{F}_p^\times$ and $\mathbf{C}^\times$, extended by 0 to all of $\mathbf{F}_p$. The *order* of a multiplicative character $\chi : \mathbf{F}_p \to \mathbf{C}$ is the smallest positive integer d such that $(\chi(x))^d = 1$ for all $x \in \mathbf{F}_p^\times$. Let χ be a multiplicative character of order $d > 1$ over $\mathbf{F}_p$ ($d|p-1$). (In our applications $d = 2$ and $\chi(x) = (\frac{x}{p})$ is the Legendre symbol.) Let $f(x)$ be a polynomial over $\mathbf{F}_p$. We say that f is a constant times a full d^{th} power if $f(x) = c(h(x))^d$ for some $c \in \mathbf{F}_p$ and $h(x) \in \mathbf{F}_p[x]$. We shall use the following estimate from the theory of character sums (cf. Schmidt [73], p.43, Theorem 2C').

Theorem 3.2.9 *If $f \in \mathbf{F}_p[x]$ has m distinct roots in the algebraic closure of $\mathbf{F}_p$ and f is not a constant times a full d^{th} power then*

$$\Big| \sum_{x \in \mathbf{F}_p} \chi(f(x)) \Big| \leq (m-1)\sqrt{p}. \tag{3.1}$$

Our chief tool will be a generalization of this result. We replace x by $\sum_{i=1}^k x_i$ and extend the summation over a subdomain of $\mathbf{F}_p^k$ only. The subdomain is defined as the intersection of k cylinders. A cylinder in the i^{th} coordinate in the Cartesian product $A_1 \times \cdots \times A_k$ is the Cartesian product of A_i and of an arbitrary subset of the Cartesian product of the remaining

$k-1$ of the A_j. Below we shall conveniently represent cylinders by their characteristic functions; by definition, each depends on $k-1$ of the variables only.

Let $\varphi_1, \ldots, \varphi_k$ be arbitrary $(0,1)$-valued functions in $k-1$ variables from $\mathbf{F}_p$. For convenience we shall view the φ_i as functions in k variables:

$$\varphi_1 = \varphi_1(x_1, x_2, \ldots, x_k),$$

with the stipulation that φ_i *does not depend* on the variable x_i. Such a function is clearly the characteristic function of a cylinder in the i^{th} coordinate in $\mathbf{F}_p^k$.

Theorem 3.2.10 *Let d, s and k be positive integers, p an odd prime, and assume $d \geq 2$, $d|p-1$, and $s < p$. Let $\chi : \mathbf{F}_p \to \mathbf{C}$ be a multiplicative character of degree d and let $f \in \mathbf{F}_p[x]$. If f has at most s distinct roots in the algebraic closure of $\mathbf{F}_p$ and f is not a constant times a full d^{th} power then for arbitrary $(0,1)$-valued functions $\varphi_i : \mathbf{F}_p^k \to \{0,1\}$ $(i = 1, \ldots, k)$ such that φ_i does not depend on the i^{th} variable, we have*

$$\Big| \sum_{x_1 \in \mathbf{F}_p} \sum_{x_2 \in \mathbf{F}_p} \cdots \sum_{x_k \in \mathbf{F}_p} \chi(f(x_1 + \ldots + x_k))\varphi_1 \cdots \varphi_k \Big| \leq c(s,k) p^{k - 2^{-k}}, \quad (3.2)$$

where $c(s,k) = (2^{k-1}s - 1)^{2^{-(k-1)}}$.

Remark. Observe that for $s < k$, we have $c(s,k) < 2$. In our applications, $s = 1$. We need the more general result for the sake of the proof by induction.

Proof. We first observe that f must satisfy the following condition.

For some root α of f, not all elements $\alpha + t$, $t \in \mathbf{F}_p$, have the same multiplicity modulo d. $(*)$

Indeed, this follows from the two assumptions that $s < p$ and f is not a constant times a full d^{th} power. For the inductive proof, we shall drop these two assumptions and assume $(*)$ only. (This, of course, still implies that f is not a constant times a full d^{th} power but weakens the $s < p$ condition.)

Now we proceed by induction on k.

The case $k = 1$ is precisely the character sum estimate (3.1).

Assume $k \geq 2$. Let S denote the sum to be estimated. Then, clearly

$$|S| \leq \sum_{x_1, \ldots, x_{k-1} \in \mathbf{F}_p} \Big| \sum_{x_k \in \mathbf{F}_p} \chi(f(x_1 + \ldots + x_k))\varphi_1\varphi_2 \cdots \varphi_{k-1} \Big|.$$

We use the Cauchy-Schwartz inequality to estimate the right hand side:

$$|S| \le p^{\frac{k-1}{2}}(\sum_{x_1,\ldots,x_{k-1}\in \mathbf{F}_p} (\sum_{x_k\in \mathbf{F}_p} \chi(f(x_1+\ldots+x_k))\varphi_1\varphi_2\cdots\varphi_{k-1})^2)^{\frac{1}{2}}. \quad (3.3)$$

We expand the squares on the right hand side. The following notation will be convenient.

$$F^{u,v}(x) = f(x+u)f(x+v);$$

$$\psi_i^{u,v}(x_1,\ldots,x_{k-1}) = \varphi_i(x_1,\ldots,x_{k-1},u)\varphi_i(x_1,\ldots,x_{k-1},v) \quad (i=1,\ldots,k-1).$$

Observe that $\psi_i^{u,v}$ does not depend on x_i. With this notation the right hand side of (3.3) expands to

$$p^{\frac{k-1}{2}} \sum_{u,v\in \mathbf{F}_p} S(u,v), \quad (3.4)$$

where

$$S(u,v) = \sum_{x_1,\ldots,x_{k-1}\in \mathbf{F}_p} \chi(F^{u,v}(x_1+\ldots+x_{k-1}))\psi_1^{u,v}\cdots\psi_{k-1}^{u,v}. \quad (3.5)$$

(We have used u and v to denote the two occurrences of x_k and moved the summation over $(x_1,\ldots,x_{k-1})$ inside.)

For $u \ne v$ we observe that $S(u,v)$ is just an instance of the sum S in (3.2), with $k-1$ in the place of k and $2s$ in the place of s.

In order to justify this claim, all we have to see is that $F^{u,v}$ satisfies $(*)$.

Let R be the multiset of roots of f in the algebraic closure of $\mathbf{F}_p$. Then R has a member α defined under $(*)$.

The set of roots of $F^{u,v}$ is the multiset $(R-u)\cup(R-v)$. Let μ_i denote the multiplicity of $\alpha+i(u-v)$ in R. Now if $F^{u,v}$ does not satisfy $(*)$ then for some ν, $\mu_{i-1}+\mu_i \equiv \nu \bmod d$ for every integer i. Since p is odd, it follows that $d|2\nu$ and $\mu_0 \equiv \mu_1 \equiv \ldots \equiv \mu_{p-1} \equiv \nu$, contrary to the choice of α. This observation sets up the conditions for an inductive step.

We infer by induction that for $u \ne v$

$$|S(u,v)| \le c(2s,k-1)p^{k-1-2^{-(k-1)}}.$$

For the case $u=v$ we use the trivial estimate

$$|S(u,u)| \le p^{k-1}.$$

Substituting into eqn. (3.4) we obtain

$$\begin{aligned} |S| &\le p^{\frac{k-1}{2}}(p\cdot p^{k-1} + p(p-1)c(2s,k-1)p^{k-1-2^{-(k-1)}})^{\frac{1}{2}} \\ &\le c(2s,k-1)^{\frac{1}{2}}p^{k-2^{-k}} = c(s,k)p^{k-2^{-k}}. \end{aligned}$$

This completes the proof of Theorem 3.2.10. ∎

3.2.4 Results on multyparty prococols

From the result of the previous subsection and from lemma **??** we easily obtain:

Theorem 3.2.11 *Let $f : [0,p]^k \to \{0,1\}$ be the function which for an input $x_1, \ldots, x_k$ takes the value 1 if $\sum_{i=1}^{k} x_k$ is a quadratic residue modulo p and otherwise it takes the value 0. Then:*

$$C_\epsilon(f) \geq \Omega(\frac{\log p}{2^k} + log\varepsilon)$$

With similar technique essentially the same bound can be obtained for the GENERALIZED INNER PRODUCT (GIP) function, where $GIP(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is the parity of the number of those cordinates where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ all agree and they are all 1.

Theorem 3.2.12 *Let f be the generalized inner product function (GIP($\mathbf{x}_1, \ldots, \mathbf{x}_k$)). Then*

$$C_\epsilon(f) \geq \Omega(\frac{n}{4^k} + log\varepsilon).$$

Here n denotes the the number of digits in each $\mathbf{x}_i$ (i=1,. . . ,k).

The major open problem is to prove lower bounds that do not decrease exponencially in k for some explicit function. Such bound will give considerable improvements in the applications.

3.2.5 Applications

As an applicatin of the above theorem in [] we prove:

Theorem 3.2.13 *There exists an explicitely given pseudorandom generator for Polylog-space $G = G_n$ which stretches $2^{O(\sqrt{\log n})}$ bits to n bits. Moreover G can be computed in polynomial time (actually even in Logspace, given multiple acces to the input bits).*

Further consequence of the above theorem is that a sequence can be explicitely given which simulates a random walk on an arbitrary graph.

Theorem 3.2.14 *There exist an (explicitely given) universal traversal sequence for general undirected graph of length $2^{2^{O(\sqrt{\log n})}}$, where n is the number of the vetices in the graph.*

Theorem **??** and **??** can be

3.2.6 Pseudorandom generators for Logspace

In the last few years there has been much research directed towards the construction of *pseudorandom* sequences – sequences that "appear" to be random to some large class of statistical tests. Blum-Micali ([BM]), Yao [Y1], and recently Impagliazzo, Levin, and Luby [ILL], have designed sequences that pass all polynomial time computable tests, under some unproven "hardness" assumptions. Reif and Tygar [RT] constructed sequences that pass all NC tests, again under some unproven "hardness" assumption. Ajtai and Wigderson [AW] give sequences that pass all AC^0 tests, using the known "hardness" results for AC^0. Nisan and Wigderson [NW1] give general constructions of sequences that pass all tests from a complexity class C, given any function that is "hard" for C. In particular, they construct pseudorandom sequences for AC^0 using the "hardness" of the parity function [NW2].

Unfortunately, no lower bounds, "hardness" results, are known for any complexity class but AC^0. Thus in particular no sequences are provably pseudorandom for any class but AC^0.

In [] we show how to construct sequences that pass all statistical tests that can be performed by Logspace Turing machines (or generally any "small"-space machines). We do this without relying on any unproven "hardness" assumption, but rather by using the multiparty communication complexity lower bounds.

An exact definition of pseudorandom generators for Logspace appears in []. In general, the definition insures that the output of a pseudorandom generator for Logspace may always be used instead of truly random sequences in any randomized Logspace computation.

Theorem 3.2.15 *There exists an (explicitly given) pseudorandom generator G for Logspace which stretches a seed of $2^{O(\sqrt{\log n})}$ random bits to n bits. Moreover G can be computed in polynomial time (actually, even in Logspace, given multiple access to the input bits).*

A particularly interesting class of tests that can be performed in RL are walks on graphs. The output of our generator will behave just like a random walk on any graph. This allows us to give an explicit construction of universal traversal sequences for general graphs.

Theorem 3.2.16 *There exists an (explicitly given) universal traversal sequence for general undirected graphs of length $2^{2^{O(\sqrt{\log n})}}$. Moreover the se-*

quence can be constructed by a Turing machine running in space logarithmic in the length of the sequence.

3.2.7 Time-space tradeoffs for Turing machines

The first time-space tradeoff we give is for general Turing machines. The machines have a read-only input tape, as well as an arbitrary number of read-write work tapes which count as space. For Turing machines that have only 1 head on the input tape, a classic result gives a time-space tradeoff of $TS = \theta(n^2)$ for the recognition of, e.g., the language $\{ww \mid w \in \{0,1\}^n\}$. However, when the Turing machine is allowed to have 2 or more heads on the read-only tape, the situation is more complicated.

For multihead Turing machines the known time-space tradeoffs are not tight and are significantly sub-quadratic. Duris and Galil [DG] and Karchmer [Ka] exhibit languages that require a time-space tradeoff of $T^2S \geq \Omega(n^3)$. Gurevich and Shelah [GS] give a language that requires a time-space tradeoff of $T^2S \geq \Omega(n^{2.5})$ on deterministic machines, but can be solved in linear time and logarithmic space nondeterministically.

In [] we give a very simple proof, that relies on our multiparty communication complexity lower bounds, of a tight, quadratic, time-space tradeoff for multihead Turing machines. We actually prove a tight separation between the power of k-head and $(k+1)$-head Turing machines:

Theorem 3.2.17 *The $(k+1)$-wise inner product function over n bit strings requires a time-space tradeoff of $TS = \Theta(n^2)$ on any k-head Turing machine.*

Note that this function can be computed in linear time and constant space on a $(k+1)$-head Turing machine. Note also that this lower bound, as well as the separation proved between k-head and $(k+1)$-head Turing machines, is tight as even a 1-head Turing machine can simulate a $(k+1)$-head Turing machine with only a quadratic penalty in time and no penalty in space. Our bounds apply to the nondeterministic, probabilistic, and average complexities as well.

3.2.8 Time-space tradeoff for branching programs and formula size

The next time-space tradeoff in **??** is for Boolean branching programs. Although the improvments we have over known results are rather modest (an

extra $\log n$ factor), we believe that the techniques are interesting. In particular, significantly stronger bounds would follow using our techniques if one could improve the lower bounds for multiparty communication complexity.

A branching program is a DAG with one source, and with two kinds of vertices: vertices that are labeled by an input variable, in this case they must have out-degree two, and one of the out-going edges must be labeled with 0 and the other with 1; the other kind of vertices are sinks and they are labeled with 1 or 0. The branching program computes a Boolean function in the natural way. The *size* of the branching program is the number of vertices. A branching program is called *leveled* if the vertices are partitioned into subsets, $L_1, \ldots, L_l$, called the levels, such that all the edges out of L_i go to L_{i+1}. The number of levels is called the *length* of the branching program, and the size of the largest level is called the *width*. A leveled branching program is called *oblivious* if all the vertices on a single level are labeled by the same variable.

The log of the width of a branching program corresponds to the space, and the length corresponds to the time. In [CFL] it was observed that lower bounds for multiparty communication complexity imply lower bounds on the length of constant width branching programs. Their techniques, however, do not give any nontrivial bounds when the width is allowed to be linear. Currently there is no known result which would give even a $2n$ lower bound on the length of general *polynomial width* branching programs. The best result along these lines is due to Alon and Maass [AM] which gives an *nlogn* lower bound on the length of polynomial width *oblivious* branching programs.

In [] we generalize the techniques of [AM] as to take advantage of multiparty communication complexity lower bounds, and achieve stronger bounds. We present an explicit function f^* (in P) such that:

Theorem 3.2.18 *Any oblivious branching program of length $o(n \log^2 n)$ computing f^* requires exponential size.*

This time-space tradeoff implies some new lower bounds for general branching program size and for Boolean formula size. So far there was only one lower bound technique known that achieves bounds above *nlogn*, either for the size of formulas over an arbitrary basis, or for general branching program size. This is the nearly quadratic bound due to Nečiporuk [Ne]. In [] we give a new technique for proving such lower bounds, a technique that works for functions that Nečiporuk's technique does not apply to.

Consequence 3.2.19 *Any branching program computing f^* requires $\Omega(n \log^2 n)$ size.*

To obtain lower bounds on formula size, in [] we show that formulas over an arbitrary basis can be converted to branching programs of polynomial size, and of length similar to that of the formula.

Consequence 3.2.20 *Any Boolean formula over an arbitrary finite basis that computes f^* requires $\Omega(n \log^2 n)$ size.*

3.3 The communication scheme of Gereb-Graus

Definition 3.3.1 *Two players intend to evaluate the Boolean function $b(\vec{x}, \vec{y})$. Let $\vec{y'}$ be a vector of some input bits ocurring in $\vec{y}$ (we say $\vec{y'} \subset \vec{y}$). Let us denote $|\vec{x}|$ by x, $|\vec{y}|$ by y and $|\vec{y'}|$ by y'. We define the* $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity *of b as follows:*

First we define a protocol. The protocol contains a function $F : \{0,1\}^{x+y'} \to \{0,1\}^x$ and a two party communication protocol P for strings of length x and y. Player A first computes $F(\vec{x}, \vec{y'}) = \vec{x'}$ and then player A and player B run the protocol P on the strings x' and y. (Note that A cannot 'remember' x after computing x'). The protocol should result the value of b. The complexity of the protocol is the worst case complexity of P.

The $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity of b is the complexity of an optimal communication scheme described above i.e. when F and P are chosen so that the above protocol requires the smallest number of bit-exchange in the worst case.

Note that the value of $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity for a Boolean function b cannot exceed the value of the classical $(\vec{x}, \vec{y})$-communication complexity, because we can chose F as the projection $F(\vec{x}, \vec{y}) = \vec{x}$.

Proposition 3.3.2 *Suppose that $y - y' - \log x \gg 1$. Then for a randomly chosen b Boolean function the $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity of b is at least $x + y - \max(x, y) - \log(x + y) - O(1)$.*

Proof: We show that the number of Boolean functions with small communication complexity is small compare to the number of all Boolean functions. The number of functions with $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity at most l is bounded from above by the number of possible $(\vec{x}, \vec{y}, \vec{y'})$-communication schemes that yield a communication of length l in the worst case.

The number of possible choices for the function F is $2^{x2^{x+y'}}$ and the number of possible choices for P is at most $2^{l2^{\max(x,y)+l}}$. For $l < x + y - \max(x, y) - \log(x + y) - O(1)$ we obtain that

$$2^{x+y} - x2^{x+y'} - l2^{\max(x,y)+l} \gg 1$$

which establishes our claim. □

If we do not assume in the proposition 3.3.2 that $y - y' > \log x$ the proposition does not hold. Indeed, for $y - y' = \log x$ the function F can encode all the possible anwers for the different values of $\vec{y} \setminus \vec{y'}$. Then in the communication part it is enough if player B sends over the bits in $\vec{y} \setminus \vec{y'}$. Here A and B communicate altogether $y - y' = \log x$ bits, which is in the case of the usual setup when $x = y = n$ significantly less than $n - \log n - O(1)$ that would be provided by the proposition.

Let us now concentrate on the case when $x = y = n$ and $y - y' = o(n)$. What we want is to construct an explicit Boolean function b such that the $(\vec{x}, \vec{y}, \vec{y'})$-communication complexity of b is $O(n)$. It turnes out that most of the explicitly given functions that are proved to have high classical communation complexity (e.g. INNER PRODUCT MOD 2) are not good for our purposses. In fact we do not have any real good candidate for such a b.

Chapter 4

Functions of Bounded Communication

4.0.1 Introduction

In this section we define a hierarchy of functions that have very low complexity with respect to almost any known computational model.

We define the hierarchy in the terms of communication complexity, but later we prove that a number of different definitions result the same notion.

Let us be given a Boolean funcion $f : 2^X \to \{0,1\}$, and a partition of the set of the input variables $X = Y \dot{\cup} Z$. Player 1 receives the values of the variables in Y and Player 2 receives the values of the variables in Z. The *communication complexity* of f with respect to the partition (Y, Z) is the number of bits the parties have to exchange in the worst case when jointly trying to evaluate the value of f according to an optimal protocol agreed upon by the parties before the game.

For more details about communication complexity we refer the reader to the paper of Lovasz [].

Definition 4.0.1 *Let f be a Boolean function on the set of Boolean variables X. If for every partition of X into two sets Y and Z the communication complexity of f remains below a constant c than we say that the* symmetric *communication complexity of f is at most c.*

Let us denote by $B(c)$ the class of functions with symmetric *communication complexity* at most c.

Construction. Let S be a commutative semigroup, $\psi : S \to \{0,1\}$ a map and $\phi_i : \{0,1\} \to S$ a map for every variable $x_i \in X$. From these data

we define a Boolean function $f : 2^X \to \{0,1\}$ by the formula

$$f(x_1, \ldots, x_n) = \psi(\sum_{i=1}^{n} \phi_i(x_i)). \tag{4.1}$$

We will say that f is represented by the semigroup S if there exists maps ψ and ϕ_i as above such that 4.1 holds.

It is easy to see that if f can be represented by S then it belongs to $B(\lceil \log |S| \rceil)$. On the other hand our main result says (see theorem 4.0.17):

Theorem 4.0.2 *For any c there exists a c' such that any function in $B(c)$ can be represented by a commutative semigroup of size at most c'.*

A less explicit characterization of the classes $B(c)$ is given in terms of the number of subfunctions $N(f)$ to a Boolean function f.

For a Boolean function f with variables $X = \{x_1, \ldots, x_n\}$, $A \subset X$ and a truth assignment $\alpha : X \setminus A \to \{0,1\}$, $f|_\alpha$ denotes the subfunction of f on A obtained by fixing variables $x_i \in X \setminus A$ to $\alpha(x_i)$. $N(f, A)$ is the number of different subfunctions of f on A. We define $N(f) = \sum_{A \subseteq X} N(f, A)$.

We will shaw that for any fixed c there is a c' such that if for a Boolean function of n variables $N(f) \leq c2^n$ then $f \in B(c')$. On the other hand it is a consequence of basic facts about communication complexity that for any c there is a c' such that for any $f \in B(c)$ the inequation $M(f) \leq c'2^n$ holds.

Gy. Turán in his paper about restricted Boolean circuits [] in 1988 remarks that there is a very interesting, but apparently less known result of Uhlig [], 1973. Uhlig showed that if for a Boolean function f of n variables $N(f) \leq c2^n$ for some constant c then the circuit complexity of f is linear.

We call a circuit *synchronous* if its inputs and gates are divided into levels so that the input gates are on level 0, and edges can go only from a level to the next one. The size of the smallest synchronous circuit with fanin ≤ 2, computing f is $C_s(f)$, the synchronous complexity of f.

Gy. Turán observes [] that Uhlig's proof is strong enough to prove $C_s(f) \leq c'n$ for functions with at most $c2^n$ subfunctions, first noticing that such functions have a bounded width linear size oblivious, read once only branching program. In subsection 4.0.6 we give an argument that sheds a new light on this statement.

Define:

$$N'(f, a) = \left(\begin{array}{c} n \\ a \end{array} \right)^{-1} \sum_{\substack{A \subseteq X \\ |A| = a}} \log N(f, A)$$

is the average of $\log N(f, A)$ taken over all the a-element subsets of X.

Turán's ideas were motivated by the the following theorem of Harper and Savage [], that makes connection between the values of $C_s(n)$ and $N'(f, a)$.

Theorem 4.0.3 *(Harper-Savage []). If* $0 < \delta < 1$, $1 \leq a \leq n$ *and* $d = \lfloor \log(\delta(n-a+1)/(2a+\delta)) \rfloor \leq D(f)$, *(*$D(f)$ *is the depth of* f*), then* $C_s(f) \geq (1-\delta)dN'(f, a)$.

The most studied branch of circuit complexity theory is the theory of circuits with bounded depth. Ajtai [] and independently Furst, Saxe and Sipser [] were the first ones who gave a superpolynomial lower bound on the size of bounded depth circuits that compute the parity function (or the majority function). Later by improving the method of Razborov [], Smolensky [] proved that bounded depth circuits with AND, OR and MOD_m gates must have exponential size if they compute the majority function assuming m is a prime power. (A MOD_m gate outputs 1 if $m| \sum x_i$ and outputs 1 otherwise.)

The difference between m being a prime power or an arbitrary natural number may seem insignificant, but as a number of attempts show the behavior of circuits with MOD_m gates for a compound number m is much less known.

Observe that a MOD_m gate computes a function that is in $B(c)$ for $c = \lceil \log m \rceil$. The AND an OR functions belong to $B(1)$. We speculate that the reason why these circuits are unable to compute the majority function is the poor ability of the gates to collect the data necessary for the computation. We thus might conjecture that no family of polynomial size circuits equipped with gates that compute functions from $B(c)$ can compute the majority function.

In section 4.0.7 we show that bounded depth circuits with gates that compute functions from $B(c)$ (c is fixed) are no more powerful than AC^0 circuits with MOD_m gates where m depends only on c.

With this argument we believe give a new reason to study the class of circuits described above.

4.0.2 Definitions and the preparation of the main theorem

For the later discussion we have to allow functions of the form $f : \{0,1\}^X \to \{1, \ldots, C\}$. The notions of communication complexity, $N(f, A)$, etc. are analogous to those in the case of Boolean functions.

A restricted version of communication protocols is introduced.

The *one-way* communication complexity of a function $f : 2^X \to \{0,1\}$ with respect to a partition of the input bits into two parts $X = Y \dot\cup Z$ is the minimal number of bits Player 1 (the one who knows the values of the variables in Y) has to send over to Player 2, who immediately can compute the value of F without further message exchange. Here the *direction* of the communication is from Y to Z. Observe that this number is $\lceil \log N(Z) \rceil$.

Proposition 4.0.4 *Let S be a commutative semigroup and the f function defined as in Construction in section 4.0.1. For any $Y \subseteq X$ we have $N(f,Y) \leq |S|$.*

Proof: The value of $\sum_{x_i \in X \setminus Y} \phi_i(x_i)$ determines the subfunction on Y. □

As a consequence of this proposition we get that the one-way communication complexity of $\psi(\sum_{x \in X} \phi_x(x))$ is at most $\lceil \log |S| \rceil$ for any partition of the set X.

Proposition 4.0.5 *For a function $f : \{0,1\}^X \to \{1,\ldots,C\}$ let c be the communication complexity of f when taking a partition of the input variables $X = Y \dot\cup Z$, and c' be the one way commmunication complexity of f from Y to Z. We have:*

$$c \leq c' \leq c2^c$$

Proof: The first inequality is obvious. For the second notice that for Player 1 it is a good strategy if he sends over all the communications possible from his point of wiew. □

For a function $f : \{0,1\}^X \to \{1,\ldots,C\}$ define $M(f) = \max_{Y \subseteq X} N(f,Y)$. From Proposition 4.0.5 directly follows:

Proposition 4.0.6 *For every c there is a c' such that $f \in B(c)$ implies $M(f) \leq c'$.*

The *symmetric* functionts are those that do not change their value under any permutation of their input bits. The value of these functions depend only on the number of 1's in the input, i.e. only on the value of $\sum_{i=1}^n x_i$. We are going to define a hierarchy of symmetric functions.

Definition 4.0.7 *We say that a symmetric function* $f : 2^{\{x_1,\ldots,x_n\}} \to \{0,1\}$ *is* quasiperiodical *with period* (c_1, c_2, c_3) *if whenever*

$$c_1 + c_3 \leq c_3 + \sum_{i=1}^{n} \epsilon_i = \sum_{i=1}^{n} \epsilon_i{}' \leq n - c_2$$

holds we have $f(\epsilon_1, \ldots, \epsilon_n) = f(\epsilon_1{}', \ldots, \epsilon_n{}')$ *The class* $Q(c)$ *is defined as the class of symmetric functions that are* (c_1, c_2, c_3) *quasiperiodical for* $c_1 + c_2 + c_3 \leq c$

Lemma 4.0.8 *Let* f *be a symmetric function.Then* $f \in Q(c)$ *for* $c = M(f)$.

Proof: Let $A \subseteq X$ a set of variables that has size C (C cannot be bigger than n). When setting the variables in A the value of $\sum_{x_i \in A} x_i$ can take any value in the range $\{0, \ldots, C\}$. Since the settings of the variables in A may result at most C different subfunctions on $X \setminus A$ there should be two settings of the variables in A, α_1 and α_2, such that they result the same subfunction on $X \setminus A$, but when evaluating the sum $\sum_{x_i \in A} x_i$ for the two settings we get r_1 and r_2 respectively that are not equal. It is easy to see that f is quasiperiodical with period $(r_1, r_2 - r_1, C - r_2)$. ∎

Lemma 4.0.9 *For any* $f \in Q(c)$ *there is a commutative semigroup* S *of size at most* c^2 *that represents* S.

Proof: Let (c_1, c_2, c_3) be the period of the function f. Define the commutative semigroup S_1 by the relation $g^{c_1+c_2} = g^{c_1}$ for a single generator element g. Define the commutative semigroup S_2 by the relation $h^{c_3+1} = h^{c_3}$ for a single generator element h. It is easy to see that $S_1 \times S_2$ represents f.

Lemma 4.0.8 and lemma 4.0.9 together provides us with a proof for theorem 4.0.2 for symmetric functions. In order to prove the characterization theorem for arbitrary $f \in B(c)$ we decompose the variable set of f into a small number of subsets (a number depending only on c) such that in each of these subsets the variables are interchangable with each other, i.e. we find the 'symmetric parts' of the function. Once we know this we can apply Lemma 4.0.8 and Lemma 4.0.9 to each equivalence class.

Definition 4.0.10 *For a function* $f : \{0,1\}^X \to \{1, \ldots, C\}$ *we define an equivalence relation on the set of the variables* X *as follows. Two variables* $x_1, x_2 \in X$ *are equivalent* ($x_1 \sim_f x_2$) *if in any situation exchanging the values of the variables* x_1 *and* x_2 *the value of* f *remains the same.*

The relation $\sim_f$ is clearly an equivalence relation. Our main lemma says that the number of equivalence classes for a function in $B(c)$ is bounded

Main Lemma *Let M be an arbitrary positive integer, X be a set of Boolean variables, $X' \subseteq X$ and f is a function with the property that for any $X' \subseteq Y \subseteq X$, $|Y \setminus X'| \leq M^2$ we have that $N(f,Y) \leq M$. Then the number of equivalence classes in $X \setminus X'$ according to $\sim_f$ is at most 2^{2^M}.*

The proof of this lemma can be found in section 4.0.5. We need a few technical statements in order to be able to make the derivation from the general case to the symmetric case. We collected these statements into one lemma.

Definition 4.0.11 *For a function $f : \{0,1\}^X \to \{1,\ldots,C\}$ and a set $Y \subseteq X$ we define $\{f,Y\}$ as the set of set of subfunctions on Y when we fix the variables in $\overline{Y} = X \setminus Y$ in all possible ways.*

Definition 4.0.12 *For a function $f : \{0,1\}^X \to \{1,\ldots,C\}$ and a set $Y \subseteq X$ we define $f_Y : 2^Y \to \{f,\overline{Y}\}$ as the function on the variables in Y that for a truth assignment α on Y gives the value $f|_\alpha \in \{f,\overline{Y}\}$.*

Lemma 4.0.13 *For a functions $f : \{0,1\}^X \to \{1,\ldots,C\}$ the following statements hold:*

1. *Let $X = X_1 \dot\cup \cdots \dot\cup X_k$ an arbitrary decomposition of the variable set X. Then there exists a function g such that*

   $$f = g(f_{X_1},\ldots,f_{X_k}).$$

   In other words the value of the function f can be gotten from the values of the functions $f_{X_1},\ldots,f_{X_k}$.

2. *Let $Y \subseteq X$. then $M(f_Y) \leq M(f)$.*

3. *Suppose that the set $Y \subseteq X$ has the property that the elements in it are equivalent with respect to the relation $\sim_f$ (for any $y_1, y_2 \in Y$ we have $y_1 \sim_f y_2$). Then the function f_Y is symmetric.*

4.0.3 Average vesus maximum

In section 4.0.1 we defined $N(f)$ as the number of subfunctions of f. For a function of n variables $N(f)$ can range from 2^n to almost 3^n. A function with $M(f) \leq c$ also has $N(f) \leq c2^n$. Surprisingly one can convert the statement:

Proposition 4.0.14 *For any c there is a c' such that if $N(f) \leq c$ then $M(f) \leq c'$.*

The proof of this proposition just as the implications 8→1 and 8→1 in Theorem 4.0.17 come from a general idea:

Idea: *For a set X let $\mathcal{H} \subseteq 2^X$ such that every Y can be produced by a fixed length straightline program from $\mathcal{H}$, where the operations are the usual set operations. Then if a function $N : 2^X \to \mathbf{R}^+$ has 'smoothness properties' with respect to the set operations then the value $\max_{H \in 2^X} N(H)$ can be be estimated by a function of $\max_{H \in \mathcal{H}} N(H)$. On 'smoothness property' we mean unequations of the type $N(\overline{H}_1) \leq g(N(H_1))$, $N(H_1 \cup H_2) \leq h(N(H_1), N(H_2))$, etc. for some functions g, h etc.*

Lemma 4.0.15 *For a function $f : \{0,1\}^X \to \{1, \ldots, C\}$ and for $Y, Z \subseteq X$ we have the following unequalities:*

1. $N(f, \overline{Y}) \leq C^{N(f,Y)}$
2. $N(f, Y \cap Z) \leq N(f,Y)N(f,Z)$
3. $N(f, Y \cup Z) \leq 2^{(N(f,Y)+N(f,Z)) \log(N(f,Y)+N(f,Z))}$
4. $N(f, Y \bigtriangledown Z) \leq 2^{\log C \cdot N(f,Y)N(f,Z)+(N(f,Y)+N(f,Z)) \log(N(f,Y)+N(f,Z))}$

We omit the proof.

Proof of Lemma 4.0.14: Lemma 4.0.15 provides us with smoothness properties for the function $N(f, X)$ required by the Idea. According to the Idea it is enough to find a set $\mathcal{H} \subseteq 2^X$ such that for every $H \in \mathcal{H}$ the value of $N(f, H)$ remains below a bound that depend only on c and that every element in 2^X can be produced by a bounded length straightline program from $\mathcal{H}$

Let us define: $\mathcal{H} := \{H \subseteq 2^X \mid N(f, H) \leq 3c\}$. We have $|\mathcal{H}| \geq \frac{2}{3}2^n > 2^{n-1}$, since otherwise the average value of $N(f, A)$ would be bigger than c. As the following lemma shows such an $\mathcal{H}$ is enough for our purposses.

Lemma 4.0.16 *Let $\mathcal{H}$ be a collection of the subsets of $X = \{x_1, \ldots, x_n\}$ of size at least $2^{n-1} + 1$. Then $\mathcal{H} \bigtriangledown \mathcal{H} = 2^X$. ($\mathcal{H} \bigtriangledown \mathcal{H}$ is the collection of the subsets of X of the form $X \bigtriangledown Z$ where both X and Z are from the set $\mathcal{H}$.)*

Proof of lemma 4.0.16: Let $Y \subseteq X$ be arbitrary. There is a $Z \subseteq X$ such that Z and $Y \bigtriangledown Z$ are both in $\mathcal{H}$, otherwise $\mathcal{H}$ could contain no more

than half of all the subsets of the set 2^X. Now $Y = Z \bigtriangledown (Y \bigtriangledown Z)$ proves $Y \in \mathcal{H} \bigtriangledown \mathcal{H}$. $\square$

Consider the function $K : 2^X \to \mathbf{N}$ where $K(f, Y)$ is the communication complexity of f with respect to the partition $(Y, X \setminus Y)$. Smoothness properties of F analogous to those in Lemma 4.0.15 can be proven, and hence the argument in Lemma 4.0.14 goes thru establishing a relationship between the average value and the maximal value of K.

4.0.4 The main theorem

We can collect our results into the following theorem:

Theorem 4.0.17 *For a language L the following things are equivalent:*

1. *There is a c that $L_n \in B(c)$ for every n.*
2. *There is a constant c that $M(L_n) \leq c$ for every n*
3. *There exists a constant c such that for every n there is a partition of the indices $\{1, 2, , \ldots, n\} = H_1 \dot\cup \ldots \dot\cup H_c$ such that f can be written in the form*
$$f = g(h_1, \ldots, h_c) \tag{4.2}$$
where $h_i : 2^{H_i} \to \{1, \ldots, c\}$ belong to $Q(c)$ for $i = 1, \ldots, c$.
4. *There exist a constant c that for every n there is a commutative semigroup S_n of size at most c such that L_n can be represented by S_n in the way given by the construction.*
5. *There exist a finite commutative semigroup S that L_n can be represented by S.*
6. *For every ordering of the indices in $\{1, 2, \ldots, n\}$ there exists a read once levelled branching program with a bound c on the width which computes L_n and which reads the bits in the input-word according to the ordering in question.*
7. *There is a constant c that $N(L_n) \leq c2^n$ for every n*
8. *There is a constant c that no matter how we cut the input of L_n into two parts of sizes l_n and $n - l_n$ ($0.1n \leq l_n \leq 0.9n$) the communication complexity is at most c between the parts.*

9. *The communication complexity of L_n for an average partition is no more than c.*

Proof:

1→2: See Proposition 4.0.6.

2→3: If we apply the Main Lemma for L_n we get classification of the indeces in $\{1,\ldots,n\}$ corresponding to the symmetry classes of L_n. The number of this classes bounded by a constant depending only on c. For the partition gotten above we aply the first item of Lemma 4.0.13. Hence we obtain a decomposition of L_n as in Equation 4.2. We are left with proving that the functions that occur in the right hand side belong to $Q(c')$ for a fixed c'. This is provided by items 2 and 3 of Lemma 4.0.13 and by Lemmasymmlemma.

3→4: Applying Lemma 4.0.9 to the functions $h_1,\ldots,h_n$ we get the commutative semigroups $S_1,\ldots,S_c$, where the size of each S_i $(i=1,\ldots,c)$ is bounded by a constant that depends only on c. It is easy to see now that $S_1\times\cdots\times\S_c$ represents L_n.

4→5: For every c there is a fixed finite commutative semigroup S that contain all the commmutative semigroups of size at most c. This will obviously represent L_n for any n.

5→1: Each L_n belong to $B(\log(\lceil|S|\rceil))$.

We have a proof so far that items 1–5 are equivalent. The equivalence of 2 and 6 is proven in Section 4.0.7 (see Proposition 4.0.22 and Proposition 4.0.24). The statements 7, 8 and 9 are obviously stronger than 2, 1 and 1 respectively, but they can be reversed using the ideas in section 4.0.3, where the implication 7→2 is explicitely stated in Proposition 4.0.14.

4.0.5 The main lemma

In this section we prove the Main lemma stated in section 4.0.2. The proof is an improved version of the proof of D. Uligh that any function with a small number ($c2^n$) of subfunctions have linear circuit size. We need two lemmas of Uhlig (see [])

Lemma 4.0.18 (*D. Uhlig) Suppose that for $Y'\subseteq X$ and $x\in X\setminus Y$ we have that $N(f,Y\cup\{x\})<N(f,Y)$. Then $\{f,Y\}\setminus\{f|_{x=0},Y\}$ and $\{f,Y\}\setminus\{f|_{x=1},Y\}$ are not empty.*

Lemma 4.0.19 (*D. Uhlig) Suppose that for some $Y'\subseteq X$ and $x\in X\setminus Y$ we have that $N(f,Y\cup\{x\})=N(f,Y)$ and also that $\{f,Y\}=\{f|_{x=0},Y\}$ (resp.*

$\{f, Y\} = \{f|_{x=1}, Y\}$*). Then there exists a map* $\pi : \{f|_{x=0}, Y\} \to \{f|_{x=1}, Y\}$ *(resp.* $\pi : \{f|_{x=1}, Y\} \to \{f|_{x=0}, Y\}$*) such that for any* $f' \in \{f, Y \cup \{x\}\}$ *we have that* $f'|_{x=1} = \pi(f'|_{x=0})$ *(resp.* $f'|_{x=0} = \pi(f'|_{x=1})$*).*

We proceed by induction on M. If $M = 1$ then f can be written as $g(X')$ and hence all the variables in $X \setminus X'$ are equivalent.

For $M > 1$ we prove that we can extend X' there is a set $X \supseteq X'' \supseteq X'$ with $|X'' \setminus X| \leq M$, that one of the following two cases hold:

1. *We can find a variable* x *in* $X \setminus X''$ *that the inequalities* $N(f|_{x=0}, Y) \leq N(f, Y) - 1$*,* $N(f|_{x=1}, Y) \leq N(f, Y) - 1$ *hold for every* $X'' \subseteq Y \subseteq X \setminus \{x\}$*.*

   From this and from the assumption of the theorem on $N(f, Y)$ we get that $N(f|_{x=0}, Y) < M - 1$ (resp. $N(f|_{x=1}, Y) < M - 1$) when $X'' \subseteq Y \subseteq X \setminus \{x\}$, $|Y \setminus X''| \leq (M-1)^2$.

   This observation allows us to use the inductional hypothesis for the functions the $f|_{x=0}$ and $f|_{x=1}$ when X' is replaced by $X'' \cup \{x\}$.

   If we have $x_i \sim_{f|_{x=0}} x_j$ and $x_i \sim_{f|_{x=1}} x_j$ for two variables $x_i, x_j \in X \setminus (X'' \cup \{x\})$ then $x_i \sim_f x_j$ is also true. This fact lets us estimate the number of equivalence classes produced by the relation $\sim_f$ on $X \setminus (X'' \cup \{x\})$ by the product of the number of classes that are produced by the relations $\sim_{f|_{x=0}}$ and $\sim_{f|_{x=1}}$.

2. *For every variable* $x_i \in X \setminus X''$ *there is a value* $\alpha_i \in \{0, 1\}$ *and a map* $\pi_i : \{f, X''\} \to \{f, X''\}$ *that if two settings* α' *and* α'' *of the variables in* $X \setminus X''$ *agree everywhere except at* x_i *where they differ, and* α' *sets* x_i *to* α_i *then* $f|_{\alpha''} = \pi_i(f|_{\alpha'})$*.*

   If for two elements $x_i, x_j \in X \setminus X''$ we have $\alpha_i = \alpha_j$ and $\pi_i = \pi_j$ then $x_i \sim_f x_j$. From here we get an upper bound on the number of the equivalence classes of the relation $\sim_f$ on $X \setminus X''$ that is two times the number of the maps from a M element set to itself.

Let us denote 2^{2^M} by $l(M)$. We can now deduce the lemma from the inequality

$$l(M) \leq M + \max\{1 + l^2(M-1), 2M^M\},$$

where the two amounts in the argument of max correspond to the two cases discussed above.

We are left with proving the existence of a set $X'' \subseteq X$ with either of the properties required above. We give an algorithm that finds such a set

of variables. The algorithm adds elements to the set X' one by one till it finds a set that qualifies for X''. The algorithm terminates within M steps. When it terminates it reaches either *branch 1* or *branch 2*. We prove that if we reach *branch 1* then the first of the above cases hold and if we reach *branch 2* the second of the above cases hold.

The algorithm: First take $N(f, X')$ and compare it with every $N(f, X' \cup x)$ where x is taken from $X \setminus X'$. If we find an $x \in X \setminus X'$ such that $N(f, X' \cup \{x\}) < N(f, X')$ then we reach at *Branch 1.*

If for every $x \in X \setminus X'$ ve have $N(f, X' \cup \{x\}) = N(f, X')$ then we separate two cases. Either there exists an $x \in X \setminus X'$ such that both $\{f, X'\} \setminus \{f|_{x=0}, X'\}$ and $\{f, X'\} \setminus \{f|_{x=1}, X'\}$ are not empty in which case we reach *branch 1* or for every $x_i \in X \setminus X'$ there exists a $\alpha_i \in \{0, 1\}$ such that $\{f, X'\} = \{f|_{x_i=\alpha_i}, X'\}$. In this case we reach *branch 2*

Finaly if none of the previous cases hold then we find an $x \in X \setminus X'$ for which $N(f, X' \cup \{x\}) > N(f, X')$. Add the element x to X' and do the previous procedure again with the extended X'.

After at most M iteration we get out of the cycle, because each time the value of $N(f, X')$ increases by at least one. We denote by X'' the final value of X'.

Lemma 4.0.20 *If we reach at branch 1 then there is an $x \in X''$ that the inequalities $N(f|_{x=0}, Y) \leq N(f, Y) - 1$, $N(f|_{x=1}, Y) \leq N(f, Y) - 1$ hold for every $X'' \subseteq Y \subseteq X$, where $x \notin Y$*

Proof: We reach branch 1 either by finding an $x \in X \setminus X''$ such that $N(f, X'' \cup \{x\}) < N(f, X'')$ or by finding an element $x \in X \setminus X''$ for which $N(f, X'' \cup \{x\}) = N(f, X'')$ and

$$\{f, X''\} \setminus \{f|_{x=0}, X''\} \neq \emptyset; \tag{4.3}$$

$$\{f, X''\} \setminus \{f|_{x=1}, X''\} \neq \emptyset \tag{4.4}$$

simultaniously hold. From Lemma 4.0.18 we obtain that Inequations 4.3 and 4.4 hold in the first case too. This is going to be the property we use.

Let us notice that Inequation 4.3 (resp. Inequation 4.4) implies that for any $Y \subseteq X \setminus (X'' \cup \{x\})$ we have $N(f|_{x=0}, Y) < N(f, Y)$ (resp. $N(f|_{x=1}, Y) < N(f, Y)$). ∎

Lemma 4.0.21 *If we reach branch 2 then for every variable $x_i \in X \setminus X''$ there is a value $\alpha_i \in \{0, 1\}$ and a map $\pi_i : \{f, X''\} \to \{f, X''\}$ that if for two settings α' and α'' of the variables in $X \setminus X''$ α' sets x_i to α_i and α'' sets x_i to $\overline{\alpha}_i$, but they agree otherwise, then $f|_{\alpha''} = \pi_i(f|_{\alpha'})$.*

Proof: If we reach *branch 2* then for every $x_i \in X \setminus X''$ we have $N(f, X'' \cup \{x_i\}) = N(f, X'')$ Moreover for every i there is an $\alpha_i \in \{0,1\}$ that $\{f, X''\} = \{f|_{x_i=\alpha_i}, X''\}$. Lemma 4.0.19 provides us with maps $\pi_i : \{f|_{x_i=\alpha_i}, Y\} \to \{f|_{x_i=\overline{\alpha}_i}, Y\}$ such that for any function $f' \in \{f, X'' \cup \{x_i\}\}$ we have

$$f'|_{x=\overline{\alpha}_i} = \pi_i(f'|_{x=\alpha_i}) \tag{4.5}$$

Take two settings α' and α'' of the variables in $X \setminus X''$ that are different only on x_i and α' sets x_i to α_i. Let $f' \in \{f, X'' \cup \{x_i\}\}$ be the subfunction on $X'' \cup \{x_i\}$ that sets the variables in $X \setminus (X'' \cup \{x_i\})$ the same way as α' and α''. From Equation 4.5 we obtain:

$$f|_{\alpha''} = f'|_{x=\overline{\alpha}_i} = \pi_i(f'|_{x=\alpha_i}) = \pi_i(f|_{\alpha'}).$$

□

4.0.6 Branching programs an synchronous circuits

Functions with small number of subfunctions can be computed by bounded width, linear size branching programs of very special form and also by linear size synchronous circuits. This fact was first observed by Gy. Turán [].

In [] the authors define: A branching program is a DAG with one source, and with two kinds of vertices: vertices that are labeled by an input variable, in this case they must have out-degree two, and one of the out-going edges must be labeled with 0 and the other with 1; the other kind of vertices are sinks and they are labeled with 1 or 0. The branching program computes a Boolean function (or for our purposses a function $f : \{0,1\}^X \to \{1, \ldots, C\}$) in the natural way. The *size* of the branching program is the number of vertices. A branching program is called *leveled* if the vertices are partitioned into subsets, $L_0, \ldots, L_l$, called the levels, such that all the edges out of L_i go to L_{i+1}. The number of levels is called the *length* of the branching program, and the size of the largest level is called the *width*. A leveled branching program is called *oblivious* if all the vertices on a single level are labeled by the same variable. An oblivios branching program is *read once only* if none of the variables occur at more than one level.

Let $x_{i_1}, \ldots, x_{i_n}$ be a permutation of the variables in the set X. We say that an oblivious read once only branching program *computes f in the order* $x_{i_1}, \ldots, x_{i_n}$ if the variables appear in this order in the consecutive levels.

Proposition 4.0.22 *For a function $f : \{0,1\}^X \to \{1, \ldots, C\}$ and an arbitrary permutation of the variables $x_{i_1}, \ldots, x_{i_n}$ There is an oblivious read*

once only branching program with width at most $M(f)$ that computes the function in the order given above.

Proof: Let us define $Y_i := \{x_1, \ldots, x_i\}$ for $i = 1, \ldots, n$. For $i = 1, \ldots, n$ the set of the nodes in the i^{th} level will be $L_i = \{f, X \setminus Y_i\}$. An edge with label 0 (1) points from $f' \in \{f, X \setminus Y_i\}$ to $f'' \in \{f, X \setminus Y_{i+1}\}$ if $f'' = f'|_{x_{i+1}=0}$ ($f'' = f'|_{x_{i+1}=0}$). It is straightforward that the branching program defined above computes f. □

We have already defined synchronous circuits in section 4.0.1 as leveled circuits where edges can go from one level to the next one. Unlike in the case of general circuits where no super-linear lower bound known, $n \log n$ lower bound can be obtained for synchronous circuits using the inequality of Harper and Savage (see theorem 4.0.3). Below we give an upper bound for any function f in terms of $M(f)$.

Proposition 4.0.23 *A function $f : \{0,1\}^X \to \{1, \ldots, C\}$ can always be computed by a synchronous circuit of size $\frac{M^2(f)}{\log M(f)} n$ and depth $K^2 \log \frac{n}{\log M(f)}$.*

We omit the simple proof that uses divide and conquer strategy and Lemma 4.0.13.

As we have the relation between $M(f)$ and $N(f)/2^n$ we can easily turn the bounds in Proposition 4.0.22 and Proposition 4.0.23 into upper bounds for the complexity of functions with small number of subfunctions in the computational models dicussed above. This conversion is however not optimal. We suspect that better bounds can be gained e.g. in the case of Theorem 4.0.22 it might be possible to take the advantage of the fact that order of the variables in which we read them in is arbitrary.

Another question is whether we could reverse Proposition 4.0.22 or Proposition 4.0.23. It is easy to see that there are Boolean functions with $M_f = 2^{n/2}$ that can be computed by linear size synchronous circuits in depth $2 + \log n$. We can immediately see however that Proposition 4.0.22 can be reversed:

Proposition 4.0.24 *If for a function $f : \{0,1\}^X \to \{1, \ldots, C\}$ and for an arbitrary permutation of the variables $x_{i_1}, \ldots, x_{i_n}$ There is an oblivious read once only branching program with width at most M that computes the function in the order given above then $M(f) \leq M$.*

4.0.7 AC^0 circuits with $\mathbf{mod}_m$ gates

There has been much work done on bounded depth, unbounded fanin circuits. Due to Ajtai [], Furst, Saxe and Sipser [], Yao [] and Hastad [] and recently to Lineal, Nisan [] we have a good grasp of the behavior of the circuits that are equipped with AND and OR gates. Using similar ideas to Razborov [], Smolensky [] proved that adding the power of MOD_m gates to these circuits still do not help in computing such simple functions like the MAJORITY function if te size of the circuit is bounded by a polynomial (a MOD_m gate outputs 0 if the nuber of 1's in its input is divisible by m and outputs 1 otherwise). Smolensky's proof that uses Moebius transformation over finite fields requires that m be a prime power. So far non of the similar approaches has been successful any compound m.

In this paper we suggest a communication complexity approach. The AND, OR and MOD_m gates all compute functions that belong to the class $B(\lceil \log m \rceil)$. We can drop every special assumption and ask:

What is the power of bounded depth polynomial size circuits where the only restriction on the gates is that they compute functions that belong to $B(c)$ for a fixed c?

It might be surprising for the first glance that these circuits can compute the same classes of functions that the circuits with AND, OR and MOD_m gates.

(Define: $E(c) = 2^{2^{2^{2^c}}}$.)

Theorem 4.0.25 *Let $\mathcal{C}$ be a circuit of depth d, size s of gates that compute any function from $B(c)$. Then there is a circuit with AND, OR and $MOD_{c!}$ gates with depth $dE(c)$, size ... that compute the the same function.*

Proof: It is enough to prove that a single function from $B(c)$ of a set of m input variables X can be computed by a circuit with AND, OR and $\text{MOD}_{c!}$ gates with depth $dE(c)$ and size ....

We go back to the characterization of $B(c)$ in terms of quasiperiodical functions. According to this characterization there is a number $a \leq 2^{2^C}$ and a partition of the input variables $X = X_1 \dot{\cup} \ldots \dot{\cup} X_a$ such that f can be written in the form

$$f = g(h_1, \ldots, h_a)$$

where $h_i : 2^{X_i} \to \{1, \ldots, a\}$ belong to $Q(c)$ for $i = 1, \ldots, c$.

Since each h_i belong to $Q(c)$ the size of their range cannot be bigger than c. Hence we can compute g by a Boolean circuit of size and depth bounded

by c^a. We also have to find circuits that compute h_i. The following lemma provides us with small circuits that compute functions in $Q(c)$.

Lemma 4.0.26 *For a function $f \in Q(c)$ of m variables there is a circuit with Boolean and MOD_m gates that compute f and has size at most ... and depth ....*

Proof: Let (c_1, c_2, c_3) be the period of f. When the number of input 1's is more than c_1 or the number of input 0's is more than c_3 the function f is perodical with period c_2. In this interval the value of f can be computed by at most c_2 MOD_{c_2} gates and a Boolean circuit on top of size $O(c_2)$ We have to construct circuits in order to copute the exact number of input 1's up to c_1 and the exact number of input 0's up to c_3. ... From these data we easily compute te value of f using $O(c)$ additional Boolean gates. □

Composing the size and depth bounds we got so far we can easily establish the statement of the theorem. □

4.0.8 Open problems

The communication coplexity approach, we believe, gives a new motivation to finding lower bounds for bounded depth circuits with Boolean and MOD_m gates, but the original problem remains widely open. We have however interesting new subproblems that might be worthwile to study. Even though the 'bounded depth' class of the models with Boolean and MOD_m gates, respectively with $B(c)$ gates coincide, the classes of functions that can be computed by depth 2, 3, etc. circuits in these models are different. An interesting step toward the final goal would be to give a lower bound on the size of circuits in the 'bounded communication' model with depth 2 that compute the MAJORY function.

Another problem is the characterization of the class $B(c)$ when c is not constant. Our multiple eponential bound on the size of the commutative semigroup that represents a function $B(c)$ yields only trivial estimates even in the case of very slowly growing functions. We can explicitely give an $f \in B(c)$ for which the number of equivalence classes produced by the 'symmetricity relation' $\sim_f$ is at least 2^{2^c}.This implies a $2^{2^{c-1}-1}$. lower bound on the the size of the smallest commutative semigroup that represents f. On the other hand we can shaw by a modification of our argument that a function f can be represented by a commutative semigroup of size $2^{2^{M(f)}}$. This tightens the gap by an exponent, but there is still a three exponent gap between the lower and upper bound.

We suspect that the bounds on the interrelation between the constants in theorem 4.0.17 can be improved and we believe it would be very interesting to do so.

Chapter 5

Threshold circuits

5.1 Introduction

The results of Ajtai [], Furst-Saxe-Sipser [], Hastad [], Razborov [], Smolensky [] give us that with respect to the bounded dept polynomial size reducibility ($\leq_{bp}$) [], $\emptyset <_{bp}$ PARITY $<_{bp} MAJORITY$.

The next step in extending this hierarchy would be to study bounded depth, plynomial size circuits where MAJORITY is also allowed as a single gate.

In this chapter we consider these circuits and the complexity class TC^0 of Boolean funcyions computed by them. This class was first studied by Parberry-Schnitger []. The definition we use allowes gates computing *threshold functions* of the form $T_k^{\vec{\alpha}}(y_1, y_2, \ldots, y_m)$ iff $\sum_{i=1}^{m} \alpha_i y_i \geq k$ (where $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m)$).

Circuits with threshold gates are also related to several computational brain models (e.g. Boltzmann mashines []) and to other models related to pattern recognition and learning, some of which have been studied since the 1950's (e.g. perceptrons [], []).

We shaw that depth 2 threshold circuits computing INNER PRODUCT MOD 2 must have exponential size. As INNER PRODUCT MOD 2 can be computed by polynomial size depth 3 threshold circuits this gives a separation of the corresponding complexity classes.

5.2 Definitions

Definition 5.2.1 *The* threshold function $T_k^{\vec{\alpha}}(y_1, y_2, \ldots, y_m)$ *is a Boolean function defined by*

$$T_k^{\vec{\alpha}}(y_1, y_2, \ldots, y_m) = \begin{cases} 1 & if \sum_{i=1}^m \alpha_i y_i \geq k \\ 0 & otherwise \end{cases}$$

where the α_i*'s are the* weights, $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_m) \in \mathbf{Z}^m$ *is the* weight vector *and* $k \in \mathbf{Z}$ *is the* threshold value.

We use the notation $\alpha = \sum_{i=1}^m |\alpha_i|$. We shall also use the notations $T_k^m(y_1, y_2, \ldots, y_m)$, $T_{\leq k}^m(y_1, y_2, \ldots, y_m)$ for the functions "at least k 1's", at most k 1's."

A *threshold circuit* $\mathcal{C}$ is a Boolean circuit where every gate computes a threshold function. We assume that the gates are on levels. Level 0 contains the nodes $x_1, x_2, \ldots, x_n$. Every eedge connects a node from level l to a node on level $n+1$. A cicuit is a formula if every gate has outdegree ≤ 1.

The *size* of the circuit is the nuber of gates, its *depth* is the is the length of the longest path. The *weight* is the maximal absolute value of the weights occuring in in gates of the circuit.

If v is a gate of $\mathcal{C}$ and $\vec{x}$ is an input, we write $\mathcal{C}_v(\vec{x})$ for the value computed at v for input $\vec{x}$. $\mathcal{C}(\vec{x})$ is the output value.

TC_d^0 is the class of languages $L \subseteq \{0,1\}^\star$ for which there is a polynomial p and a sequence $(\mathcal{C}_n)_{n \in \mathbf{N}}$ of depth d threshold circuits such that $\mathcal{C}_n$ computes $L \cap \{0,1\}^n$ and both the size and the weight of $\mathcal{C}_n$ are bounded by $p(n)$.

$TC^0 = \bigcup_{d \in \mathbf{N}} TC_d^0$ is the class of languages recogniszble by threshold circuits of bounded depth such that the size and the weight are bounded by a polynomial. (As for AC^0, one can define uniform variants as well).

Definition 5.2.2 *The* symmetric function $S_H^{\vec{\beta}}(y_1, y_2, \ldots, y_m)$ *is a Boolean function defined by*

$$S_H^{\vec{\beta}}(y_1, y_2, \ldots, y_m) = \begin{cases} 1 & if \sum_{i=1}^m \beta_i y_i \in H \\ 0 & otherwise \end{cases}$$

where the β_i*'s are the* weights, $\vec{\beta} = (\beta_1, \beta_2, \ldots, \beta_m) \in \mathbf{Z}^m$ *is the* weight vector *and* $H \subset \mathbf{Z}$.

Obviously the threshold functions are special symmetric functions. Clearly not every symmetric funtion can be expressed as a threshold function, but we have the following:

Proposition 5.2.3 *Let $S_H^{\vec{\beta}}(y_1, y_2, \ldots, y_m)$ be a symmetric function where $H = \{h_1, h_2, \ldots, h_l\}$. Then the circuit*

$$T_{l+1}^{2l}(T_{h_1}^{\vec{\beta}}(\vec{x}), T_{\leq h_1}^{\vec{\beta}}(\vec{x}), \ldots, T_{h_l}^{\vec{\beta}}(\vec{x}), T_{h_l}^{\vec{\beta}}(\vec{x}))$$

Define the INNER PRODUCT MOD 2 as the $2n$-variable function $\vec{x} \cdot \vec{y} = x_1 y_1 \oplus x_2 y_2 \oplus \cdots \oplus x_n y_n$. Proposition 5.2.3 implies that this function is in TC_3^0. It will be shown not to belong TC_2^0.

5.3 The Pudlak lemma

Definition 5.3.1 *Let $\mathcal{C}$ be a circuit with n inputs and $A, B \subseteq C_n$ be disjoiint sets. Let P_a (resp. P_B) denote the uniform probability distribution on A (resp. B). Then $\mathcal{C}$ is an* ϵ-discriminator *for A and B if*

$$|P_A(\mathcal{C}(\vec{x}) = 1) - P_B(\mathcal{C}(\vec{x}) = 1)| \geq \epsilon$$

If f is an n-variable Boolean function, $\mathcal{C}$ is an *ϵ-discriminator for f* if it is an ϵ-discriminator for $A = f^{-1}(1)$ and $B = f^{-1}(0)$.

Lemma 5.3.2 *Let $T_k^{\vec{\alpha}}(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m)$ be a circuit with m inputs, $\alpha = \sum_{i=1}^m |\alpha_i|$ and $A, B \subseteq C_n$ be disjoint sets such that the circuit accepts A and rejets B. Then some subcircuit $\mathcal{C}_i$ $(1 \leq i \leq m)$ is a $1/\alpha$ discriminator for A and B.*

Proof: Let the random variable $C_i^A(\vec{x})$ (resp. $C_i^B(\vec{x})$) be the output of $\mathcal{C}_i$ when $\vec{x}$ is distributed uniformly on A (resp. B). Then

$$\sum_{i=1}^m \alpha_i E(C_i^A(\vec{x})) \geq k$$

and

$$\sum_{i=1}^m \alpha_i E(C_i^B(\vec{x})) \leq k - 1.$$

Taking expectations and rearranging we get:

$$1 \leq \sum_{i=1}^m \alpha_i (E(C_i^A(\vec{x})) - E(C_i^B(\vec{x}))) \leq \alpha \cdot \max_{1 \leq i \leq m} |P_A(C_i(\vec{x}) = 1) - P_B(C_i(\vec{x}) = 1)|.$$

□

The lemma holds for arbitrary probability distributions. The lemma can be slightly generalized:

Lemma 5.3.3 *Let S be a symmetric function with at most k alternation. Suppose $S^{\vec{\alpha}}(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m)$ is a circuit with n inputs, $\alpha = \sum_{i=1}^{m} |\alpha_i|$ and $A, B \subseteq C_n$ be disjoint sets such that the circuit accepts A and rejets B. Then there exist indices $1 \leq i_1 < i_2 < \cdots < i_l \leq n$ such that $l \leq k$ and the circuit $\mathcal{C}_{i_1} \wedge \mathcal{C}_{i_2} \wedge \cdots \wedge \mathcal{C}_{i_l}$ is an $\frac{1}{(2^k-1)\alpha^k}$ discriminator for A and B.*

Proof: First we prove that the function $S^{\vec{\alpha}}(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m)$ can be expressed as $T_j^{\vec{\beta}}(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s)$ where

1. $s = \left(\begin{array}{c} m \\ 1 \end{array} \right) + \left(\begin{array}{c} m \\ 2 \end{array} \right) + \cdots + \left(\begin{array}{c} m \\ k \end{array} \right)$
2. The circuits $\mathcal{D}_i$ are ranging among all the circuits of the form $\mathcal{C}_{i_1} \wedge \mathcal{C}_{i_2} \wedge \cdots \wedge \mathcal{C}_{i_l}$.
3. $\beta = \sum_{i=1}^{s} |\beta_i| \leq (2^k - 1)\alpha$.

If the alternation points of S are $j_1, j_2, \ldots, j_k$ then the polynomial $Q = (-1)^{\nu}(x - j_1)(x - j_2)\cdots(x - j_k)$ is such that it is nonnegative in where S is 1 and negative where S is 0 for either $\nu = 1$ or $\nu = -1$. The function $T_0^1(Q(\sum_{i=1}^{m} \alpha_i \mathcal{C}_i))$ easily corresponds to a threshold function which satisfy 1 and 2. In order to see 3 we notice that $|Q(\alpha) - Q(0)| \geq (2^k - 1)\alpha^k$.

The Proof is completed by aplying lemma 5.3.2 to the function $T_j^{\vec{\beta}}(\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_s)$. □

5.4 Lower bounds for depth 2 circuits

Lemma 5.4.1 *Suppose there is a communication protocol P which is an ε discriminator for a function f. Then the $\varepsilon/8$-distributional complexity of f is at most the length of P.*

Proof: What we have to prove is that there is a protocol P' such that

$$||Pr(f(\mathbf{x}, \mathbf{y}) = P'(\mathbf{x}, \mathbf{y}))| - |Pr(f(\mathbf{x}, \mathbf{y}) = P'(\mathbf{x}, \mathbf{y}))|| \geq \varepsilon/8. \qquad (5.1)$$

If $|Pr(f = 1) - Pr(f = 0)| \geq \varepsilon/8$ then a protocol with length 1 will do (that always outputs the majority value of f). Otherwise the protocol P or the protocol P'

Theorem 5.4.2 *Let $f(\vec{x},\vec{y})$ be a Boolean function with $|f^{-1}(0)-1/2|\geq\delta$. Suppose that f is computable by a two level ciruit with a majority gate of weight $k<1/4\delta$ on the top and symmetric gates on the first level. Then the weight of one of these symmetric gates is at least $2^{C_\varepsilon(f)}$ where $\varepsilon=\frac{1}{2k}-2\delta$ and $C_\varepsilon(f)$ is the ε-distributional complexity of f introduced in section...*

Proof: Lemma 5.3.2 gives us that one of the symmetric gates on the first level say T should be an ε-discriminator for f. From

$$|Pr(T(\vec{x},\vec{y})=f(\vec{x},\vec{y}))-Pr(T(\vec{x},\vec{y})\neq f(\vec{x},\vec{y}))|>\frac{1}{2k}-2\delta$$

we get that any communication protocol that yields T requires at least $C_\epsilon(f)$ bits of communication in the worst case. Consider the following protocol:

We first begin to compute the linear form by computing the corresponding sum for $\vec{x}$. Then we send it over to the party that knows $\vec{y}$ which can finish the evaluation of f. We obtain that the coefficients of the linear form belonging to $\vec{x}$ sum up to at least $2^{C_\epsilon(f)-1}$. Since the same is true for the coefficients belonging to $\vec{y}$, the lower bound on the weight of T is established. ∎

Lemma 5.4.3 ***Lemma 5.4.4*** *ipl The ε-distributional communication complexity of the INNER PRODUCT MOD 2 function is at least $n/2+\log\varepsilon$*

Proof: The proof will be an easy consequence of a special case of a lemma due to Lindsey (see [], [], []):

Lemma 5.4.5 *(Lindsey [], [], []) For every $X,Y\subseteq\{0,1\}^n$*

$$||\{(\vec{x},\vec{y})\in X\times Y\mid\vec{x}\cdot\vec{y}=1\}|-|\{(\vec{x},\vec{y})\in X\times Y\mid\vec{x}\cdot\vec{y}=0\}||\leq\sqrt{|X|\cdot|Y|\cdot 2^n}.$$

If P is an arbitrary protocol, then each running of it results a product set $A\times B$. In such a set the Lemma of Lindsey provides us with a $\sqrt{|A|\cdot|B|\cdot 2^n}\leq 2^{\frac{3}{2}n}$ upper bound for the discrepancy of f. (the discrepancy of a 0-1 function f on a subset S of its domain is $||S|-\sum_{s\in S}f(s)|$.

Let T be the number of possible runnings of the protocol. From the bound of the discrepancy in the case of a single running we easily get an estimation on the performance of P:

$$T2^{-\frac{1}{2}n}=||P(f(\mathbf{x},\mathbf{y})=P(\mathbf{x},\mathbf{y}))|-|P(f(\mathbf{x},\mathbf{y})=P(\mathbf{x},\mathbf{y}))||.\geq\varepsilon \qquad (5.2)$$

∎

Theorem 5.4.6 *Assume that $\mathcal{C}$ is a depth 2 threshold corcuit computing the INNER PRODUCT MOD 2 of two n bit strings. Then the size of $\mathcal{C}$ is at least $2^{(1/4-\epsilon)n}$.*

Proof: Let f denote the INNER PRODUCT MOD 2 function. We have that $|Pr(f^{-1}(0)) - 1/2| = \delta = 1/2^{n+1}$. Let w be the weight of the output gate. From Lemma 5.4.2 we get that there is a gate in the first level with weight at least $2^{C_{1/2w-2\delta}(f)}$. If $w \geq 2^{n/4}$ we are immediately done. We can suppose therefore that $w \leq 2^{n/4}$. Using the estimation for $C_\epsilon(f)$ provided by Lemma **??** we get the folloing estimation for the size of $\mathcal{C}$:

$$W(\mathcal{C}) \geq \max\{w, 2^{n/2-2-\log w}\}. \tag{5.3}$$

The unequality immediately implies what we were to prove. ∎

Bibliography

[1] H. Abelson, *Lower bounds on information transfer in distributed computing,* Proc 19th IEEE Symp. on Foundation of Computer Science, (1978), pp. 151–158

[2] A.V. Aho, J.D. Ullman. M. Yanakakis, *On notions of information transfer in VLSI circuits,* STOC '83

[3] M. Ajtai, Σ_1^1 *formulae on finite structures,* Annals of Pure and Appl. Logic, 24 (1984), pp. 1–48

[4] M. Ajtai, L. Babai, P. Hajnal, J. Komlós, P. Pudlák, V. Rödl, E. Szemerédi, Gy. Turán, *Two lower bounds for branching programs,* Proc. 18th ACM STOC (1986), pp. 30–38

[5] M. Ajtai and M. Ben-Or. *A theorem on probabilistic constant depth computations,* STOC '84, pp. 471-474.

[6] M Ajtai, Y. Gurevich, *Monotone versus Positive,* ACM Oct 1987, pp. 1004–1015

[7] M. Ajtai, A Wigderson, *Deterministic simulation of probabilistic constant depth circuits* 26th FOCS (1985), pp. 11–19

[8] R. Aleliunas, R.M. Karp, R.J. Lipton, L. Lovász, C. Rackoff, *Random walks, universal traversing sequences and the complexity of maze problems,* FOCS '79

[9] N Alon, W Maass, *Meanders, Ramsey theory and lower bounds for branching programs,* 27th FOCS pp. 410-417

[10] D. Angluin. *On counting problems and the polynimial-time hierarchy,* Theoretical computer Science, vol. 12, (1980), pp. 161-173.

[11] L. Babai, P. Frankl, J. Simon, *Complexity classes in communication complexity theory,* FOCS '86, pp. 337–347

[12] L. Babai, P. Hajnal, E. Szemerédi and Gy. Turán, *A lower bound for read-only-once branching programs,* J.C.S.S., vol. 35 (1987), pp. 153–162

[13] L. Babai, N. Nisan, M. Szegedy, *Multiparty protocols an Logspace-hard pseudorandom sequences* STOC '1989, pp. 1–11

[14] L. Babai, P. Pudlák, V. Rödl and E. Szemerédi, *Lower bounds to the complexity of symmetric Boolean functions,* submitted for publication

[15] D. A. Barrington, *Width-3 permutation branching programs,* draft (1985), MIT

[16] D. A. Barrington, *Bounded-width polynomial size branching programs recognize exactly those languages in NC^1,* (1986), Proc. 18th ACM STOC, pp. 1–5

[17] P.W. Beame, S.A. Cook, H.J. Hoover, *Log depth circuits for division and related problems* SIAM J. Comp. 15 (1986), pp. 994–1003

[18] M. Ben-Or, *Lower bounds for algebraic computational trees,* (1983), Proc. 15th ACM STOC, pp. 247–248

[19] M. Blum, S. Micali, *How to generate chriptografically strong sequences of pseudo random bits,* 23rd FOCS (1982), pp. 112–117

[20] B. Bollobas, *Random graphs,* Academic Press (1985)

[21] R. Boppana, M. Sipser, *The complexity of finite functions,* (preprint), To appear in Handbook of Theoretical Computer Science, North-Holland 1989

[22] A. Borodin, S.A. Cook, P.W. Dymond, W.L. Ruzzo, M.Tompa, *Two applications of inductive counting for complementation problems,* (manuscript), (1988)

[23] A.K. Chandra, M.L. Furst, R.J. Lipton, *Multi-party protocols,* Proc. 15th ACM STOC, 1983, pp. 94-99.

[24] Ashok K. Chandra, Larry Stockmeyer and Uzi Vishkin. *Constant depth reducibility,* Siam J. Computing. Vol 13, No 2, May 1984, pp. 423-439.

[25] B. Chor, O. Goldreich, *Unbiased bits from weak sources of weak randomness and probabilistic communication complexity,* FOCS '85, pp. 429–442

[26] R.L. Dobrushin, S.J. Ortyukov, *Upper bound for te redundancy of of self correcting arrangements of unreliable functional elements,* Prob. Info. Transm. 13 (1977), pp. 59–65

[27] P. Duris, Z. Galil, *A time-space tradeoff for language recognition,* Math. Systems theory 17 (1984), pp. 3–12

[28] P. Erdős, J. Spencer, *Probabilistic methods in combinatorics,* Akadémiai Kiadó (1974)

[29] R. Fagin, M.M. Klawe, N. Pippenger, L.J. Stockmeyer, *Bounded depth, polynomial size circuits for symmetric functions,* Theor. Comp. Sci., 36 (1985), 239–250

[30] M. Furst, J. B. Saxe and M. Sipser. *Parity, circuits and the polynomial-time hierarchy,* Math. Systems Theory 17 (1984), pp. 13-27.

[31] Y. Gurevich, H. R. Lewis, *A logic for constant depth circuits,* Inf and Control, 61 (1984), pp. 65–74

[32] Y. Gurevich, S. Shelah, *Nondeterministic linear-time tasks may require substantially nonlinear nondeterministic time in the case of sublinear workspace,* 20th STOC pp. 281-289

[33] A. Hajnal, W. Maass, P Pudlak, M. Szegedy and G. Turan. *Threshold circuits of bounded depth,* FOCS '87, pp. 99-110

[34] A. Hajnal, W. Maass and Gy. Turán, *On the communication complexity of graph properties,* Proc. 20th ACM STOC (1988), pp. 186–191

[35] J. Hastad, *Improved lower bounds for small depth circuits,* Proc. 18th ACM STOC (1986), pp. 6–20

[36] J. Hastad. *Computational Limitations of small depth circuits,* Ph. D. dissertation. MIT Press. (1988), 82 pages.

[37] N. Immerman, *Languages that capture complexity classes,* 15th STOC (1983), 347–354

[38] N. Immerman, *Expressibility as a complexity measure: results and directions,* Structures Conf. (1987), pp. 194-202

[39] Immerman $NLOG = coNLOG$

[40] N. Immerman, E. Lander, *Telling Graphs appart: a first order approach to graph isomorphism,* (preprint), (1984)

[41] R. Impagliazzo, L. Levin, L. Lubi, *Pseudorandom generators from one-way function,* STOS '89

[42] S. Istrail *Polynomial universal traversing sequences for cycles are constructible,* STOC '88

[43] J.D. Kahn, N. Linial, N. Nisan, M.E. Saks, *on the cover time of random walks in graphs,* Journal of theoretical probability, vol. 2, No. 1 (1989)

[44] M. Karchmer, *Two time-space tradeoffs for element distinctness,* Theoretical Computer Science 47, (1986)

[45] H. Karlof, R. Paturi, J Simon, *Universal sequences of length* $n^{O(\log n)}$ *for cliques,* (manuscript) (1987)

[46] V. King, *Lower bounds on the complexity of graph properties,* Proc. 20th ACM STOC (1988), pp. 468–476

[47] D. Knuth, *The art of computer programming.* Vol II

[48] T. Leighton, E. Leiserson, B. Maggs, S. Plotkin, J. Wein *Theory of Parallel and VLSI Computation,* MIT, March 1988

[49] N. Linial, N Nisan, *Approximate Inclusion-Exclusion,* To appear

[50] N.Linial, Y. Mansour, N. Nisan, *Constant depth circuits, Fourier transform and Learnability,* To appear

[51] Lipton, Sedgewick, *Lower bounds for VLSI,* Proceedings of the 13^{th} Symposium on Theory of computing, (1981), pp. 300–307.

[52] L. Lovász, *Communication complexity: a survey,* Princeton University, CS-TR-204-89 (1989)

[53] O.B. Lupanov, *Implementing the algebra of formulas in the basis* $+, *, -$*,* Dokl. Acad. Nauk SSSR 136 (1961), 5, pp.1041–1042, (Engl. tr.: Sov. Phis. Dokl 6, (1961), pp. 107–108)

[54] W.F. McColl, M.S. Paterson, *The depth of all Boolean functions,* SIAM J. Comp. 6 (1977), pp. 373–380

[55] M. Minsky, S. Papert, *Perceptrons: An introduction to computational geometry,* MIT press (1982)

[56] E.I. Neciporuk, *A Boolean function,* Soviet Mathematics Doklady 7:4 (1966), pp. 999-1000

[57] J. Von Neuman, *Probabilistic logics and the synthesis of reliable organisms from unreliable components, in J. E. Shannon, J. McCarthy (eds.),* Automata Studies (Princeton University Press) (1956), pp. 43–98

[58] N. Nisan, *Probabilistic vs. deterministic decision trees and CREW PRAM complexity,* (preprint)

[59] N. Nisan, *1-way vs. 2-way acces to randomness in Logspace,* (manuscript), (1989)

[60] N. Nisan, A. Wigderson, *Hardness vs. Randomness,* FOCS '88

[61] I. Parberry, G. Schnitger, *Parallel computation with threshold functions,* Structure in Complexity Theory (ed. A.L. Selman), LNCS 223, Springer (1986), pp. 272–290

[62] N. Pippenger, *On networks of noisy gates,* 26th FOCS (1985), pp. 30–38

[63] W. Paul, N Pippinger, E. Szemeredi, W. T. Trotter, *On determinism versus nondeterminism and related problems,* 24th FOCS pp.429-438

[64] P. Pudlák, *A lower bound on complexity of branching programs,* Proc. Conf. on the Mathematical Foundations of Computer Science, Springer Lecture Notes in Computer Science, vol. 176 (1984), pp. 480–489

[65] A. A. Razborov, *Lower bounds for the monotone complexity of some Boolean functions,* (in Russian), Dokl. Akad. Nauk SSSR, vol. 281 (1985), pp. 798–801

[66] A. A. Razborov, *A lower bound for the monotone network complexity of the logical permanent,* (in Russian) Matematicheskie Zametki 37:6 (1985); English translation: Math. Notes of the Acad. of Sci. of the USSR, vol. 37, pp. 485–493

[67] A. A. Razborov, *Lower bounds on the size of bounded depth networks over a complete basis with logical addition,* (in Russian), Matematicheskie Zametki **41:4** (1987), 598–607; English translation: Math. Notes of the Acad. of Sci. of the USSR, vol. 41:4 (1987), pp. 333–338

[68] A. A. Razborov. *On The Method of Approximations,* STOC '89, pp. 167-176.

[69] J.H. Reif, *On threshold circuits and polynomial computation,* Proc. 2nd. Structure in Complexity Theory Conf. (1987), pp. 118–125

[70] J.H. Reif, J.D. Tygar, *Towards a theory of parallel randomized computation,* TR-07-84, Aiken computation lab., Harvard University (1984)

[71] D.E. Rumelhardt, J.L. McClelland and the PDP Research Group, *Parallel Distributed Processing: Explorations in the mikrostructures of Cognition,* vol. 1 (MIT Press) (1986)

[72] M. Saks and A. Wigderson, *Probabilistic boolean decision trees and the complexity of evaluating game trees,* Proc. 26th IEEE FOCS (1986), pp. 29–38

[73] W.M. Schmidt, *Equations over Finite Fields, An Elementary Approach,* Springer Lecture Notes in Mathematics Vol. 536, Springer 1976.

[74] H.U. Simon, *A tight* $\Omega(n \log n)$ *bound on the paralel RAM's to compute nondegenerate Boolean functions,* FCT '83, Lecture Notes in Comp. Sci., 158 (1984), pp. 151–153

[75] M. Sipser. *Borel sets and circuit complexity,* STOC '83, pp. 61-69.

[76] S. Skyum, *A measure in which Boolean negation is exponentially powerful,* Inf Proc. Lett. 17 (1983), pp. 125–128

[77] S. Skyum, L.G. Valiant, *A complexity theory based on Boolean algebra,* J. ACM 32 (1985), pp. 484–502

[78] R. Smolensky, *Algebraic methods in the theory of lower bound for Boolean circuit complexity,* Proc. 19th ACM STOC, (1987), pp. 77–82

[79] M. Tompa, H. Venkateswaran, *A new pebble game that characterizes parallel complexity classes,* 27th FOCS pp. 348-360

[80] Gy. Turán, *The critical complexity of graph properties,* Inf. Proc. Lett. 18 (1984), pp. 151–153

[81] Gy. Turán, *Lower bounds for synchronous circuits and planar circuits,* Inf. Proc. Lett., 30 (1989), pp. 37–40

[82] Gy. Turán, *On restricted Boolean circuits,* The University of Illinois at Chicago (preprint) (1989)

[83] D. Uhlig, *On the relationship between the circuit complexity of a Boolean function and the number of its subfunctions,* Problems of Cybernetics, 26 (973), pp. 183–201 (in Russian)

[84] L. Valiant, *Graph theoretic argument in low level complexity,* Mathematical Foundation of Computer Science, Springer Verlag - North Holland, 1977 pp. 162-175

[85] U.V. Vazirani, *Towards a strong communication complexity theory or generating quasy-random sequences from two communicating random sources,* STOC '85

[86] I. Wegener, *The complexity of Boolean functions,* Wiley-Teubner Series in Comp. Sci. (1987)

[87] A.C. Yao, *Some complexity questions related to distributed computing,* 11th STOC (1979), pp. 209–213

[88] A.C. Yao, *Theory and applications of trapdoor functions,* Proc. 23the IEEE FOCS (1982), pp. 80–91

[89] A.C. Yao, *Lower bounds by probabilistic arguments,* Proc. 24th IEEE FOCS (1983), pp. 420–428

[90] A.C. Yao, *Separating the polynomial-time hierarchy by oracles,* Proc. 26th IEEE FOCS (1985), pp. 1–10

[91] A.C. Yao, *Lower bounds to randomized algorithms for graph properties,* Proc. 28th IEEE FOCS (1987), pp. 393-400

[92] A. C. Yao. *Circuits and local computation,* STOC '89. pp. 186-196.