THE UNIVERSITY OF CHICAGO

COMBINATORIAL METHODS IN BOOLEAN FUNCTION COMPLEXITY

A DISSERTATION SUBMITTED TO

THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES

IN CANDIDACY FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY

ANNA GÁL

CHICAGO, ILLINOIS

AUGUST 1995

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, János Simon. His continued support and attention has been very beneficial for my work and I learned a lot from him. I would like to thank László Babai for his guidance and for all his generous help that greatly influenced this work and my education.

The regular meetings for the last few years with Gyuri Turán, Farrokh Vatan and Satya V. Lokam have been very motivating and helped to build my understanding of complexity theory.

I am grateful to Avi Wigderson for his kind invitation to Hebrew University. I have been very fortunate to have the opportunity to work with him and enjoy the hospitality and excellent research environment of the Computer Science Department at Hebrew University. The problems considered in Section 3.1 and in Chapter 4 have been suggested by him, and he introduced me into these areas.

I have learned a lot from working with Mario Szegedy. The results in Section 2.2 are joint work with him.

I thank Mike Paterson and Amos Beimel for their enthusiastic collaboration which led to the results in Section 3.1.

Finally, I would like to thank the Department of Computer Science of the University of Chicago for a friendly and productive environment.

TABLE OF CONTENTS

ABSTRACT

In this dissertation we explore a central theme of the theory of computing: the study of the inherent complexity of computational tasks via combinatorial models such as Boolean circuits, span programs, branching programs.

In the *first part* of the thesis we study fault tolerance of Boolean circuits.

First we study the model of independent random faults, introduced by von Neumann. In this model the gates of the circuit may fail with a probability bounded by some small constant, and the failures occur independently. A circuit is reliable, if it produces the correct result with high probability on any input. We give a general lower bound for the size needed for the reliable computation of Boolean functions in this model. In some cases this matches the known upper bounds. We prove that the reliable computation of any Boolean function with sensitivity s requires $\Omega(s \log s)$ gates if the gates of the circuit fail independently with a fixed positive probability. This theorem was stated by Dobrushin and Ortyukov in 1977, but their proof was not complete as pointed out by Pippenger, Stamoulis and Tsitsiklis in 1991. We use the general approach of Dobrushin and Ortyukov together with a new probabilistic argument. The $\Omega(s \log s)$ bound holds even if s is the block sensitivity instead of the sensitivity of the Boolean function.

Next we introduce a model for adversarial faults. We consider synchronized circuits and we allow an adversary to choose a small constant fraction of the gates at each level of the circuit to be faulty. We require that even in the presence of such faults the circuit compute a “loose version” of the given function. We present an efficient construction for computing arbitrary symmetric functions in this model. We also show a perhaps unexpected relation between this model and probabilistically checkable proofs.

The *second part* of the thesis gives two lower bounds for computing Boolean functions. The first result provides lower bounds for monotone span programs, a model introduced in 1993 by Karchmer and Wigderson. The second result exhibits an AC^0-computable function that requires exponential size read-once branching programs. Both lower bounds are based on combinatorial properties of the families of minterms of the functions computed.

In the *third part* of the thesis we study the relation of Boolean and arithmetic circuits. We prove that polynomial size semi-unbounded fan-in Boolean circuits of depth d with n inputs can be simulated by polynomial size semi-unbounded fan-in arithmetic circuits of depth $O(d + \log n)$, where the arithmetic operations $+$, $-$, $\times$ are performed in an arbitrary finite field. The proof is based on a randomized reduction using the Isolation Lemma of Mulmuley, Vazirani and Vazirani.

CHAPTER 1

INTRODUCTION

1.1. Complexity of Boolean functions

A central problem of the theory of computing is to understand the inherent complexity of computational tasks in terms of various resources required.

We will study combinatorial models of computation for computing Boolean functions.

1.1.1. The basic model: Boolean circuits

Suppose we want to compute the value of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, using a certain set of operations, and we want to find out what is the minimum number of operations we have to perform for computing the function. The set of available operations is called the *basis* of the computation. A basis Φ is called *complete* if any Boolean function can be computed using only operations from Φ.

A *Boolean circuit* is a directed acyclic graph composed of input nodes of indegree 0, labeled by variables from the set $\{x_1, \ldots, x_n\}$, and nodes of indegree ≥ 1 labeled by Boolean operations from a given basis. The nodes of the circuit are called *gates*. The indegree of a node corresponds to the number of variables of the associated operation and it is called the *fan-in* of the gate.

The *size* of a circuit is the number of its gates. The *depth* of a circuit is the length of the longest directed path from an input to the output of the circuit.

A Boolean circuit *computes* the Boolean function $f : \{0,1\}^n \to \{0,1\}$ if the value computed by the output gate of the circuit equals to the value of the function $f(x_1, \ldots, x_n)$ on each input.

The *circuit complexity* over the basis Φ of a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is the minimum number of gates in a Boolean circuit over the basis Φ computing f.

We note that changing from one finite and complete basis to another may influence the size and depth of optimal circuits for a given Boolean function by at most constant factors [62].

We will consider for example the standard Boolean basis $\{\wedge, \vee, \neg\}$, with fan-in two $\wedge$, $\vee$ gates. This is a complete basis, i.e. any Boolean function can be computed using only these operations. If the basis is not mentioned, we will have the standard Boolean basis in mind.

Lupanov [59] proved that any Boolean function of n variables can be computed by circuits over $\{\wedge, \vee, \neg\}$ of size $2^n/n + o(2^n/n)$. Shannon's counting argument [85] shows that size $\Omega(2^n/n)$ is necessary for computing almost all Boolean functions of n variables. Thus the circuit complexity over $\{\wedge, \vee, \neg\}$ of almost all Boolean functions of n variables is $2^n/n + o(2^n/n)$. However the largest known lower bounds for the circuit complexity of explicit Boolean functions of n variables are linear in n [20, 67, 90, 104], and we do not know an explicit function that requires superlogarithmic circuit depth.

The circuit complexity of a Boolean function is related to the complexity of computing the function in many other important models of computation. For example, if a function can be computed by Turing machines in time $T(n)$ then it has Boolean circuits of size $O(T(n) \log T(n))$ [72].

1.1.2. The complexity classes NC and AC

A *family of functions* $\mathcal{F}$ is a sequence $f_1, f_2, \ldots,$, where f_n is a Boolean function of n variables.

The class NC^i is defined as the class of Boolean function families that can be computed by uniform sequences of constant fan-in circuits in polynomial size and depth $O((\log n)^i)$ [68, 25]. We have $NC = \cup_{i=0}^{\infty} NC^i$. The class NC can be described as the class of problems that are efficiently computable in parallel.

Similarly, the class AC^i is defined as the class of Boolean function families that can be computed by uniform sequences of unbounded fan-in Boolean circuits in polynomial size and depth $O((\log n)^i)$. We have $AC = \cup_{i=0}^{\infty} AC^i$. It is well known that $AC^k \subseteq NC^{k+1} \subseteq AC^{k+1}$ for every k, and $NC = AC$.

By uniform sequences we mean that each circuit in the sequence can be constructed by the same procedure (as a function of the length of the input) and this procedure is efficient in some sense, for example performed by a logspace bounded Turing machine [84].

Sometimes the above classes are defined without the uniformity condition.

We note that these classes are often referred to as language classes. Computing a Boolean function f corresponds to recognizing whether or not a given input x belongs to the language $L = \{x | f(x) = 1\}$.

1.1.3. Boolean formulas

A *Boolean formula* is a Boolean circuit where each gate has outdegree at most 1.

The logarithm of the size of an optimal formula computing a Boolean function f is within a constant factor of the circuit depth required for computing f [89, 98].

The class of Boolean function families computable by polynomial size formulas is the same as the class NC^1.

1.2. Fault tolerance of Boolean circuits

It is important to have constructions that reliably perform a given computation task even in the presence of errors, without increasing the size of the computation by too much.

We consider Boolean circuits with gates that may malfunction, and we study if it is still possible to reliably compute a function.

1.2.1. Independent random faults

Much of the work in this area uses the model of independent random faults, originally introduced by von Neumann [64]. The assumption is that the gates of the circuit may fail with probability bounded by some small constant, and the failures occur independently. The circuit should produce the correct result with high probability on any input. In this model it is known [64, 29, 70] that any function can be reliably computed by $L \log L$ size circuits, where L is the size needed to compute the function without faults. Pippenger [70] proved that almost all functions can be computed with only constant redundancy. By *redundancy* we mean the fraction of the size needed for computation with faults and the size needed to compute the function without faults. These results hold if the probability of a gate being faulty is bounded by some constant $< 1/2$.

We prove that the logarithmic increase in size is necessary for the reliable computation of certain functions. This theorem was stated by Dobrushin and Ortyukov in 1977, but the proof they gave in [28] was found to include questionable arguments by Pippenger, Stamoulis and Tsitsiklis [73]. We use the general approach of Dobrushin and Ortyukov, together with some new probabilistic lemmas. This result appears in [38, 37]. An independent proof of this result was given by Gács [36, 37] and by Reischuk and Schmeltz [81].

1.2.2. Adversarial faults

The problem of building reliable circuits with small redundancy becomes more difficult if the faults are not random. If an adversary chooses to destroy critical parts of the circuit, for example the output gate, then the circuit cannot give the correct result. Yet in many cases we would like to deal with worst case behavior.

Previous attempts to build reliable circuits in the case when the faults are not random allowed using a certain number of absolutely reliable gates. The practical justification for introducing absolutely reliable elements is that circuits can be built using more expensive hardware for certain gates, that will be more reliable than the rest of the circuit. Reliable circuits with small redundancy were obtained but they could tolerate only a negligible (exponentially small) fraction of the gates being faulty [52, 53, 94]. Moreover these constructions have exponential size, thus they give small redundancy only for functions that require exponential size circuits even without faults.

In a recent paper [54] Kleitman, Leighton and Ma investigate the *short-circuit model of gate failure* for Boolean circuits in which a faulty gate is restricted to output one of its input values. They show that in this model it is possible to tolerate worst case faults without introducing absolutely reliable gates.

To our knowledge, there are no previous results achieving small redundancy tolerating a constant fraction of gates being faulty in the case when the faults are not random. For a survey of related results see [71].

We propose a new model of fault tolerance for Boolean circuits. We consider synchronized (leveled) circuits and let the adversary choose a certain fraction of the gates at each level to be faulty. Our model could be thought of as using a constant number of absolutely reliable gates for the last few levels of the circuit. Thus, for practical purposes it can be justified similarly to the previous approaches to handle non-random faults.

Instead of trying to correctly compute the function on every input we define the *loose computation* of a function. For the loose computation of a function f we require the output to be 1 whenever $f(x) = 1$, but the output has to be 0 only on inputs that have a large enough neighborhood where f is identically 0.

We prove that every symmetric function has a synchronized circuit of size $O(n)$ and depth $O(\log n)$ that performs the loose computation of the function even if a constant fraction of the gates at each level is chosen to be faulty by an adversary. This bound is within a constant factor of the complexity of arbitrary symmetric functions (that depend on all n inputs) in the fault free case [98].

There is a connection between our model and recently discovered constructions for proof encodings [10, 11] (also see [13] for a survey). We show that from certain constructions of fault tolerant circuits the theorem of Arora and Safra [10], $NP = PCP(\log n, \log n)$ follows. Our results verify the existence of such fault tolerant circuits. However, combining these results we do not get a simpler proof to the characterization of NP in [10]: our construction uses an even harder result stating that $NP = PCP(\log n, 1)$ [11]. The above results appear in [41].

1.3. Lower bounds for Boolean complexity

While proving superlinear lower bounds for general Boolean circuits remains a very hard open problem, there has been much success in proving lower bounds for restricted versions of the Boolean circuit model.

A *monotone circuit* is a Boolean circuit using only monotone gates, i.e. gates from the basis $\{\wedge, \vee\}$.

Razborov [76] introduced a method of proving superpolynomial lower bounds for computing explicit functions by monotone circuits. He proved $n^{\Omega(\log n)}$ lower bounds for the monotone circuit complexity of the clique and perfect matching functions on n-node graphs. Based on Razborov's method exponential lower bounds were obtained for the monotone circuit complexity of several functions from NP [8, 7].

The above results can be used to derive lower bounds for the depth of monotone circuits, but the depth lower bounds obtained this way will be logarithmic in the size bound.

Karchmer and Wigderson [49] introduced a technique for proving lower bounds on the depth of monotone circuits that are super-logarithmic in the size of optimal circuits for the function considered. They proved $\Omega((\log n)^2)$ lower bounds for the depth of monotone circuits computing the *st-connectivity* function, i.e. deciding whether an undirected graph on n nodes contains a path between the nodes s and t. Based on the method of [49] $\Omega(n^\epsilon)$ and $\Omega(n)$ lower bounds were obtained by [43] and [80] for the depth of monotone circuits computing explicit functions on n-node graphs.

Another area where proving lower bounds has been successful is considering bounded depth circuits. Improving the lower bounds of [1, 35, 102] Hastad [44] proved exponential lower bounds for computing the parity function by bounded depth circuits with unbounded fan-in gates over the basis $\{\wedge, \vee, \neg\}$. Razborov [77] proved that computing the majority function by bounded depth circuits requires exponential size even allowing the use of parity gates. Smolensky [88] extended the above results, proving exponential lower bounds for computing the MOD_r function by bounded depth circuits over $\{\wedge, \vee, \neg, MOD_p\}$, if p is a prime and r is not a power of p.

A very exciting area of research is trying to find new techniques that might lead to superlinear lower bounds for general Boolean circuits [78, 48, 51, 100].

1.3.1. Span programs

Karchmer and Wigderson [50] introduced span programs as a linear algebraic model of computation. A span program for a Boolean function is presented as a matrix over some field with rows labeled by literals of the variables, and the size of the program is the number of rows. The span program accepts an assignment if and only if the all-ones row is a linear combination of the rows whose labels are consistent with the assignment.

Lower bounds for the size of span programs imply lower bounds in several other models, for example for formula size.

Monotone span programs have only positive literals (non-negated variables) as labels of the rows. They compute only monotone functions, even though the computation uses non-monotone linear algebraic operations. It is known that every function with a polynomial size span program is in NC (this follows from [19, 24, 50, 60]), but no monotone analog of this result is known.

In this model, it is not known how to prove large lower bounds for explicit functions even in the monotone case. The $\Omega(m^2/\log m)$ lower bound implied by [26] for monotone span program size is the strongest previously known lower bound for an explicit function on m variables. [17] introduced a method that yields quadratic lower bounds for explicit functions, improving on the bound by [26]. The methods presented in [17] and [26] cannot give lower bounds larger than $\Omega(m^2)$.

We present a new technique for proving lower bounds for monotone span programs, which is a generalization of the method in [17]. The new method could possibly yield even exponential lower bounds. So far, our largest lower bound for an explicit function on m variables is $\Omega(m^{2.5})$. We obtain this bound for the function that is defined to have the value 1 if and only if the input graph contains a 6-clique. These results appear in [18]

Another motivation for studying monotone span programs is their connection to secret-sharing schemes. A *secret-sharing scheme* is a cryptographic tool in which a dealer shares a secret, taken from a finite set of possible secrets, among a set of parties such that only some pre-defined authorized sets of parties can reconstruct the secret. To achieve this goal the dealer distributes private shares to the parties such that any authorized subset of parties can reconstruct the secret from its shares and any non-authorized subset cannot gain even partial information about the secret. The authorized sets correspond to a Boolean function $f : \{0,1\}^m \rightarrow \{0,1\}$, where m is the number of parties, such that the authorized sets are the sets with their characteristic vectors in $f^{-1}(1)$.

A secret-sharing scheme can only exist for authorized sets specified by monotone functions: if a subset B can reconstruct the secret then every superset of B can also reconstruct the secret.

A secret-sharing scheme is considered efficient, if the length of the shares is not too large (say polynomial) relative to the number of parties.

The question of whether there exist Boolean functions with no efficient scheme is open. The best lower bound was proved by Csirmaz [26]. His proof gives, for every m, a Boolean function with m variables for which the sum of the lengths of the shares in every secret-sharing scheme is $\Omega(m^2/\log m)$ times the length of the secret (for every finite set of possible secrets).

Karchmer and Wigderson [50] proved that if there is a monotone span program of size s for some function then there exists a scheme for the corresponding secret-sharing problem in which the sum of the lengths of the shares of all the parties is s. Therefore, every lower bound on the total size of shares in a secret-sharing scheme is also a lower bound on the size of monotone span programs for the same function. On the other hand, lower bounds for monotone span programs imply lower bounds for *linear* secret-sharing schemes [15, 16, 27].

1.3.2. Branching programs

A *branching program* is a directed acyclic graph with a source node called START, and two sinks called ACCEPT and REJECT. Every vertex that is not a sink has outdegree 2, and the two edges leaving a given vertex are labeled by complementary literals $x_i, \bar{x}_i$ for some variable x_i $(1 \leq i \leq n)$. For every input string $(x_1, \ldots, x_n)$, $x_i \in \{0,1\}$ the label of each edge evaluates to 0 or 1 depending on the value of the corresponding variable. A given input string is accepted by the program if and only if there is a directed path from START to ACCEPT along which all edge-labels take value 1 under this input. The branching program is said to compute the Boolean function $f : \{0,1\}^n \to \{0,1\}$ which takes the value 1 precisely on the strings accepted by the branching program. The *size* of a branching program is the number of the nodes. A branching program is *read-once* if every variable occurs at most once along each source-sink path.

A number of papers have presented lower bounds for read-once branching programs. Exponential lower bounds are given for explicit functions in [2, 14, 30, 46, 56, 57, 74, 75, 86, 99, 103]. Results for the more general case of read-k-times branching programs appear in [23, 47, 79].

We list some functions which have previously been shown to require exponential size read-once branching programs. (This is not intended to be a complete list.) [30] considers the *Hamiltonian-Circuit* and the *Perfect-Matching* problems. [86] proves an exponential lower bound for the function taking value 1 if and only if the input graph on m vertices is $m/2$-regular. [74] presents an exponential lower bound for integer multiplication. The *Clique-Only* function is defined as having value 1 if the input represents the edges of a graph on m vertices which is an $m/2$ size clique. The *Clique-Only* function can be computed by polynomial size read-twice branching programs [99] and by NC^1 circuits [98], (polynomial size logarithmic depth constant fan-in circuits), but requires exponential size read-once branching programs [75, 99, 103]. In the above examples, the exponential lower bounds are of the form $2^{\Omega(\sqrt{n})}$, where n is the number of variables. A lower bound of $2^{\Omega(n)}$ for an n-variable function is given in [2, 14] for the *Triangle-Parity* problem, i.e. for the function taking the value 1 if and only if the input graph contains an odd number of triangles.

As a consequence of [1, 35], none of the above families of functions belongs to AC^0 (see [44, 102] for stronger results).

Jukna [46] and Krause et al. [57] exhibit a function which is AC^0 computable and at the same time it requires exponential size read-once branching programs. They consider the *Exact-Perfect-Matching* function taking the value 1 if and only if the input graph consists of a perfect matching. They prove that this function requires $2^{\Omega(\sqrt{n})}$ size read-once branching programs, where n is the number of variables.

We show that there exist families of functions even in depth-2 monotone AC^0 that require exponential size read-once branching programs. This result appears in [40].

1.4. Boolean vs. arithmetic circuits

There are several approaches to compare the power of Boolean and arithmetic computational models. One possibble approach is to compare complete problems for analogous complexity classes defined by the corresponding models. Valiant and Vazirani [97] gave a randomized reduction from the satisfiability problem to unique satisfiability, and proved that $NP/poly \subseteq \oplus P/poly$. Avi Wigderson [101] proved that $NL/poly \subseteq \oplus L/poly$, by giving a randomized reduction from $s-t$ connectivity to unique $s-t$ connectivity for directed graphs.

Our work complements the above results. We study similar questions for complexity classes defined by limited depth circuits. We consider semi-unbounded fan-in Boolean and arithmetic circuits allowing the $\vee$ and $+$ gates to have unbounded fan-in while requiring the fan-in of the $\wedge$ and $\times$ gates to remain bounded. We prove that polynomial size semi-unbounded fan-in Boolean circuits of depth d with n inputs can be simulated by polynomial size semi-unbounded fan-in arithmetic circuits of depth $O(d + \log n)$, where the arithmetic operations $+$, $-$, $\times$ are performed in an arbitrary finite field.

Observe that this problem is only interesting for semi-unbounded fan-in circuits, because a single $\vee$ of m inputs can be expressed as a polynomial of degree m over any field. Thus it is easy to simulate unbounded fan-in polynomial size, depth d Boolean circuits by unbounded fan-in polynomial size, depth $O(d)$ arithmetic circuits simply by replacing each $\wedge$ gate by $\times$ and simulating each $\vee$ gate by such a polynomial. Similarly, bounded fan-in Boolean circuits are easy to simulate by bounded fan-in depth $O(d)$ arithmetic circuits.

For semi-unbounded fan-in circuits, replacing each $\vee$ gate by the corresponding polynomial we obtain $\Omega(d \log n)$ depth arithmetic circuits. Razborov [77] showed that an m-input $\vee$ can be well approximated by degree $\log m$ polynomials over finite fields. By a result of Borodin [21] using these polynomials and amplifying the approximation it is possible to get $O(d \log\log n + \log n)$ depth and polynomial size semi-unbounded arithmetic circuits that simulate semi-unbounded (and even completely unbounded fan-in) depth d Boolean circuits. Note however that we cannot hope for further improvement if we try to simulate each $\vee$ gate separately. Our "global" reduction gives depth $O(d + \log n)$ simulation. The proof is based on a randomized reduction using the Isolation Lemma of Mulmuley, Vazirani and Vazirani [61]. These results appear in [39, 42].

CHAPTER 2

FAULT TOLERANCE OF BOOLEAN CIRCUITS

2.1. A lower bound in the model of independent random faults

2.1.1. Definitions and previous work

We prove lower bounds on the number of gates needed to compute Boolean functions by circuits with noisy gates. We say that a gate fails if its output is incorrect. A *noisy gate* fails with a probability bounded by some constant $\varepsilon \in (0, 1/2)$, and the gates in the circuit fail independently. A computation is *reliable* if the value computed by the circuit on any given input is correct with high probability.

For reliable computations the size of circuits with noisy gates has to be larger than the size needed for computations using only correct gates. By the *noisy complexity* of a function we mean the minimum number of gates needed for the reliable computation of the function. It depends of course on the error probabilities of the gates, and also on how reliable the circuit has to be. Note that in this model the circuit cannot be more reliable than its last gate. For a given function, the ratio of its noisy and noiseless complexities is called the *redundancy* of the noisy computation of the function.

The following upper bounds are known for the noisy computation of Boolean functions. The results of von Neumann [64], Dobrushin and Ortyukov [29] and Pippenger [70] prove that if a function can be computed by a noiseless circuit of size L, then $O(L \log L)$ noisy gates are sufficient for the reliable computation of the function. Pippenger [70] proved, that any function depending on n variables can be computed by $O(2^n/n)$ noisy gates. Since the noiseless computation of almost all Boolean functions requires $\Omega(2^n/n)$ gates (Shannon [85], Muller [62]), this means that for almost all functions the redundancy of their noisy computation is just a constant. Pippenger

[70] also exhibited specific functions with constant redundancy. For the noisy computation of any function of n variables over a complete basis Φ, Uhlig [93] proved upper bounds arbitrarily close to $\rho(\Phi)2^n/n$ as $\varepsilon \to 0$, where $\rho(\Phi)$ is a constant depending on Φ, and $\rho(\Phi)2^n/n$ is the asymptotic bound for the noiseless complexity of almost all Boolean functions of n variables (Lupanov [59]).

These are rather surprising results. It is natural to ask whether there exist functions with nonconstant redundancy or whether the $O(L \log L)$ upper bound of [64, 29, 70] is tight for some functions, and if so, exhibit such functions.

Dobrushin and Ortyukov in their 1977 paper [28] stated the following theorem providing answers to this important problem: The computation of any function with sensitivity s requires $\Omega(s \log s)$ gates if the gates of the circuit fail independently with a fixed probability $\varepsilon \in (0, 1/2)$, but the value computed by the circuit on any input is incorrect with probability not greater than $p \in (0, 1/3)$. Thus, in particular, the reliable computation of the parity or the "or" functions of n variables requires $\Omega(n \log n)$ noisy gates.

Unfortunately, as noticed by Pippenger, Stamoulis and Tsitsiklis [73], the proof in [28] is incorrect. Pippenger, Stamoulis and Tsitsiklis [73] pointed out the two questionable arguments in the proof, and suggested that part of the strategy seemed hopelessly flawed. They gave in [73] an $\Omega(n \log n)$ lower bound for the parity function, keeping part of the approach of Dobrushin and Ortyukov, but replacing a significant part of their proof with entirely new arguments using specific properties of the parity function. The more general statement about any function with given sensitivity remained unproven.

We prove that functions with sensitivity s do indeed require $\Omega(s \log s)$ noisy gates for their reliable computation. We can prove the stronger $\Omega(b \log b)$ lower bound, where b is block sensitivity rather than sensitivity. The results hold for circuits with arbitrary constant fan-in gates. Thus, they also hold for circuits over an incomplete basis, for example monotone circuits. The proof uses the original Dobrushin-Ortyukov strategy, proving the correct probabilistic lemmas to carry it out.

We note that these are the only known lower bounds proving nonconstant redundancy for functions other than the parity function, and they allow to prove maximal $\Omega(\log L)$ redundancy of noisy computation over arbitrary constant fan-in basis for a large class of functions, including all symmetric functions.

A different proof of Theorem 2.1.4 by Péter Gács, which works for any $p \in (0, 1/2)$ is presented in [37]. The paper of Reischuk and Schmeltz [81] gives an independent proof of the $\Omega(s \log s)$ lower bound. The dependence on the failure probabilities of the gates was later improved by Evans and Schulman [31, 33]. The effect of errors on the depth of the computation has been studied in [32, 31, 33].

2.1.2. The lower bound

Let f be a Boolean function of n variables. Let $x = (x_1, \ldots, x_n)$ be any input string. Denote by x^ℓ the input string which differs from x only in the ℓ-th bit, i.e. $x^\ell_i = x_i$ for each $i \neq \ell$ and $x^\ell_\ell = \neg x_\ell$.

Definition 2.1.1 The function f is *sensitive* to the ℓ-th bit on x if $f(x) \neq f(x^\ell)$. The *sensitivity of f on x* is the number of bits to which f is sensitive on x. The *sensitivity* of f is the maximum over all x of the sensitivity of f on x.

We consider Boolean circuits with gates having constant fan-in and computing functions from a finite set Φ. A complete basis is a set of functions such that any Boolean function can be represented by their composition. Φ may or may not be a complete basis. We assume only that any circuit C computing a particular function f uses constant fan-in gates computing functions from a finite set Φ_C, such that f can be represented by the composition of functions from Φ_C.

Let $n(\Phi_C)$ be the maximum fan-in of the gates computing functions from the set Φ_C. Let n_g denote the fan-in of gate g.

Definition 2.1.2 Let $z \in \{0, 1\}^{n_g}$. Denote by $g(z)$ the value of the function that the gate g has to compute, on input z. We say that the gate g *fails*, if receiving input z it outputs a value different from $g(z)$.

Theorem 2.1.4 gives the lower bound for the case that the gates fail independently with a fixed probability $\varepsilon \in (0, 1/2)$. It has been argued (Pippenger [70]) that for proving lower bounds this is the best model to consider, as opposed to proving upper bounds, where the assumption that the gates fail independently with probability at most $\varepsilon \in (0, 1/2)$ is more appropriate.

Definition 2.1.3 Denote by $C(x)$ the value computed by the circuit C on input x. We say that a circuit *computes f with error probability at most p* if the probability that $C(x) \neq f(x)$ is at most p for any input x.

The main lower bound theorem is stated below:

Theorem 2.1.4 *Let ε and p be any constants so that $\varepsilon \in (0, 1/2)$, $p \in (0, 1/4)$. Let f be any Boolean function with sensitivity s. Suppose a circuit whose gates fail independently with fixed probability ε computes f with error probability at most p . Then the number of gates of the circuit is at least $\Omega(s \log s)$.*

Corollary 2.1.5 *The redundancy of the noisy computation by Boolean circuits of any function of n variables with $O(n)$ noiseless complexity and $\Omega(n)$ sensitivity is $\Omega(\log n)$.*

Corollary 2.1.5 applies to a large class of functions. In particular the following statement holds:

Corollary 2.1.6 *The redundancy of the noisy computation by Boolean circuits of any nonconstant symmetric function of n variables is $\Omega(\log n)$.*

We note that there is a difference between the redundancy of noisy computation by circuits and by decision trees. A similar model of noisy computation is considered by Feige et al. [34] for Boolean decision trees. The nodes of the tree are allowed to be independently faulty with some probability, and the result of the computation has to be correct with at least a fixed probability for every input. Feige et al. [34] give bounds for the depth of noisy decision trees computing symmetric functions. These bounds show that some nonconstant symmetric functions have constant redundancy of noisy computation by decision trees.

Corollary 2.1.7 *There exist Boolean functions of n variables with constant redundancy of noisy computation by decision trees and $\Omega(\log n)$ redundancy of noisy computation by circuits.*

2.1.3. Noisy wires

Following Dobrushin and Ortyukov, for the proof of Theorem 2.1.4 we consider an equivalent problem.

Let C be a circuit satisfying the condition that if its gates fail independently with probability ε then the circuit computes f with error probability at most p.

As suggested in [28], consider the case when not only the gates but the wires of C may fail as well. We say that a wire fails when it transmits an incorrect value.

Let $\delta \in [0, \varepsilon/n(\Phi_C)]$ and suppose that the wires of C fail independently, each with probability δ. This means that the input $y \in \{0,1\}^{n_g}$ received by gate g may be different from the input $t \in \{0,1\}^{n_g}$ that the gate should have received.

The following statement is proved as Lemma 3.1 in [28]: Let $\varepsilon \in (0, 1/2)$, $\delta \in [0, \varepsilon/n(\Phi_C)]$. Then for any gate g of the circuit C there exist unique values $\eta_g(y,\delta) \in [0,1]$, so that if the wires of C fail independently with probability δ and the gate g fails with probability $\eta_g(y,\delta)$ when receiving input y, then the probability that the output of g is different from $g(t)$ (where t is the input entering the input wires of the gate) is equal to ε.

Consider now the behavior of circuit C in two different failure modes. In the first mode the wires of the circuit are correct and the gates fail independently with probability $\varepsilon \in (0, 1/2)$. In the second mode each wire fails independently with fixed probability $\delta \in [0, \varepsilon/n(\Phi_C)]$ and each gate fails independently with probability $\eta_g(y,\delta)$ when receiving y. Lemma 3.2 of [28] shows that these two failure modes are equivalent in the sense that the circuit C computes f with the same error probability:

for any input x and any gate g the output of g differs from the output computed by the same gate in an error free computation of C on input x with the same probability in both modes. Thus to prove Theorem 2.1.4 it suffices to prove a lower bound for the size of C computing f with error probability at most p with errors occurring at both the wires and the gates. More precisely, we shall prove the following

Theorem 2.1.8 *Let δ and p be any constants so that $\delta \in (0, 1/2)$, $p \in (0, 1/4)$. Let f be any function with sensitivity s. Let C be a circuit such that its wires fail independently with fixed probability δ and each gate g fails independently with probability $\eta_g(y, \delta)$ when receiving y. Suppose C computes f with error probability at most p. Then the number of gates of C is at least $\Omega(s \log s)$.*

2.1.4. Probabilistic lemmas

In this section we prove a few statements which we will need for the proof of Theorem 2.1.4.

Lemma 2.1.9 *Let $\alpha_1, \ldots, \alpha_n$ be independent events, $\gamma \in (0, 1)$ and $Pr[\bigcup_{i=1}^{n} \alpha_i] \leq \gamma$. Then*

$$Pr[\bigcup_{i=1}^{n} \alpha_i] \geq (1 - \gamma) \sum_{i=1}^{n} Pr[\alpha_i] \, .$$

Proof:

$$\begin{aligned} Pr[\bigcup_{i=1}^{n} \alpha_i] &\geq \sum_{i=1}^{n} Pr[\alpha_i \cap (\neg \bigcup_{j \neq i}{}_{j=1}^{n} \alpha_j)] \\ &\geq \sum_{i=1}^{n} Pr[\alpha_i] \, (1 - Pr[\bigcup_{j=1}^{n} \alpha_j]) \\ &\geq (1 - \gamma) \sum_{i=1}^{n} Pr[\alpha_i] \, . \end{aligned}$$

Lemma 2.1.10 *Let E be an event, p and c constants from $(0, 1)$. Let α and λ be independent events such that $Pr[\lambda] \geq c$ and $Pr[E \mid \alpha] \geq 1 - p$. Then*

$$Pr[E \mid \alpha \cap \lambda] \geq 1 - \frac{p}{c} \, .$$

Proof:

$$Pr[E \mid \alpha \cap \lambda] = 1 - Pr[\neg E \mid \alpha \cap \lambda]\,.$$

$$\begin{aligned} Pr[\neg E \mid \alpha \cap \lambda] &= \frac{Pr[\neg E \cap \alpha \cap \lambda]}{Pr[\alpha \cap \lambda]} \le \frac{Pr[\neg E \cap \alpha]}{Pr[\alpha \cap \lambda]} \\ &= \frac{Pr[\neg E \mid \alpha] Pr[\alpha]}{Pr[\alpha] Pr[\lambda]} \le \frac{p}{c}\,. \end{aligned}$$

Lemma 2.1.11 *Let E be an event and $p \in (0,1)$ a constant. Let $\alpha_1, \ldots, \alpha_n$ be independent events such that $Pr[E \mid \alpha_i] \ge 1 - p$ for $\forall i$. Then*

$$Pr[E \mid \bigcup_{i=1}^{n} \alpha_i] \ge (1 - \sqrt{p})^2\,.$$

Proof:

We prove that if the conditions of the lemma hold then for any $c \in (0,1)$

$$Pr[E \mid \bigcup_{i=1}^{n} \alpha_i] \ge (1 - \frac{p}{c})(1 - c)\,. \qquad (2.1)$$

Taking $c = \sqrt{p}$ we get the statement of the lemma.

Let us use the notation $\lambda_i = \neg(\alpha_1 \cup \ldots \cup \alpha_i)$. Then the events α_1, $\alpha_2 \cap \lambda_1$, $\ldots$, $\alpha_{k+1} \cap \lambda_k$ do not intersect and

$$\bigcup_{i=1}^{n} \alpha_i = \alpha_1 \dot\cup (\alpha_2 \cap \lambda_1) \dot\cup (\alpha_3 \cap \lambda_2) \dot\cup \ldots \dot\cup (\alpha_n \cap \lambda_{n-1})\,,$$

$$Pr[\lambda_1] \ge Pr[\lambda_2] \ge \ldots \ge Pr[\lambda_{n-1}]\,. \qquad (2.2)$$

Fix any constant $c \in (0,1)$. Suppose $Pr[\lambda_k] \ge c$ for some k. Then since $\alpha_{\ell+1}$ and λ_ℓ are independent events, by Lemma 2.1.10 and (2.2) the following holds for each $1 \le \ell \le k$:

$$Pr[E \mid \alpha_{\ell+1} \cap \lambda_\ell] \ge 1 - \frac{p}{c}\,.$$

Since $\bigcup_{i=1}^{k+1} \alpha_i = \alpha_1 \dot\cup (\alpha_2 \cap \lambda_1) \dot\cup \ldots \dot\cup (\alpha_{k+1} \cap \lambda_k)$ we get that

$$\text{if } \; Pr[\lambda_k] \ge c \; \text{ then } \; Pr[E \mid \bigcup_{i=1}^{k+1} \alpha_i] \ge 1 - \frac{p}{c}\,. \qquad (2.3)$$

This proves (2.1) if $Pr[\lambda_{n-1}] \geq c$.

If $Pr[\lambda_{n-1}] < c$ and $Pr[\lambda_1] \geq c$ then consider the largest index k such that $Pr[\lambda_k] \geq c$. Thus $1 \leq k < n-1$ and

$$Pr[\lambda_k] \geq c \quad \text{but} \quad Pr[\lambda_{k+1}] < c\,.$$

By (2.3) $Pr[E \mid \bigcup_{i=1}^{k+1} \alpha_i] \geq 1 - (p/c)$, and

$$Pr[\bigcup_{i=1}^{k+1} \alpha_i] \geq 1 - c \ \text{ since } \bigcup_{i=1}^{k+1} \alpha_i = \neg\lambda_{k+1}\,.$$

We get

$$\begin{aligned} Pr[E \mid \bigcup_{i=1}^{n} \alpha_i] &\geq Pr[E \mid \bigcup_{i=1}^{k+1} \alpha_i] Pr[\bigcup_{i=1}^{k+1} \alpha_i] \\ &\geq (1 - \frac{p}{c})(1-c)\,. \end{aligned}$$

If $Pr[\lambda_1] < c$ then $Pr[\alpha_1] \geq 1 - c$ and

$$\begin{aligned} Pr[E \mid \bigcup_{i=1}^{n} \alpha_i] &\geq Pr[E \mid \alpha_1] Pr[\alpha_1] \\ &\geq (1-p)(1-c) \\ &> (1 - \frac{p}{c})(1-c) \end{aligned}$$

which concludes the proof of Lemma 2.1.11.

2.1.5. Proof of the lower bound

We prove the "noisy wires" version (Theorem 2.1.8).

Let z be an input such that f has maximum sensitivity on z. Let $S \subset \{1, \ldots, n\}$ be the set of indices so that $\ell \in S$ if and only if f is sensitive to the ℓ-th bit on input z. Then $|S| = s$, where s is the sensitivity of f.

For each $\ell \in S$ denote by B_ℓ the set of all wires originating from the ℓ-th input of the circuit. Let $m_\ell = |B_\ell|$.

For any set $\beta \subset B_\ell$, let $\omega(\beta)$ be the event that the wires belonging to β fail and the other wires of B_ℓ are correct.

Denote by β_ℓ the subset of B_ℓ where

$$\max_{\beta \subset B_\ell} Pr[C(z^\ell) = f(z^\ell) \mid \omega(\beta)]$$

is obtained. Note that β_ℓ may or may not be the empty set.

By the conditions of the theorem, C computes f with error probability at most p, which means that $Pr[C(z^\ell) = f(z^\ell)] \geq 1 - p$. Thus,

$$Pr[C(z^\ell) = f(z^\ell) \mid \omega(\beta_\ell)] \geq 1 - p\,. \tag{2.4}$$

Denote by α_ℓ the event that the wires of B_ℓ not belonging to β_ℓ fail and the wires of β_ℓ are correct. In other words: $\alpha_\ell = \omega(B_\ell \setminus \beta_\ell)$.

Since f is sensitive to the ℓ-th bit on z

$$Pr[C(z) \neq f(z) \mid \alpha_\ell] = Pr[C(z^\ell) = f(z^\ell) \mid \omega(\beta_\ell)]\,.$$

By (2.4) this means that for each $\ell \in S$

$$Pr[C(z) \neq f(z) \mid \alpha_\ell] \geq 1 - p\,.$$

α_ℓ are independent events since the wires fail independently. Thus we can apply Lemma 2.1.11 and get

$$Pr[C(z) \neq f(z) \mid \bigcup_{\ell \in S} \alpha_\ell] \geq (1 - \sqrt{p})^2.$$

Using this inequality, from

$$\begin{aligned} p &\geq Pr[C(z) \neq f(z)] \\ &\geq Pr[C(z) \neq f(z) \mid \bigcup_{\ell \in S} \alpha_\ell] Pr[\bigcup_{\ell \in S} \alpha_\ell] \end{aligned}$$

we conclude that

$$Pr[\bigcup_{\ell \in S} \alpha_\ell] \leq \frac{p}{(1 - \sqrt{p})^2}\,. \tag{2.5}$$

$p/(1 - \sqrt{p})^2 \in (0, 1)$ since $p \in (0, 1/4)$. Applying Lemma 2.1.9 we get

$$Pr[\bigcup_{\ell \in S} \alpha_\ell] \geq (1 - \frac{p}{(1 - \sqrt{p})^2}) \sum_{\ell \in S} Pr[\alpha_\ell]\,. \tag{2.6}$$

Using

$$Pr[\alpha_\ell] = (1-\delta)^{|\beta_\ell|}\delta^{m_\ell - |\beta_\ell|} \geq \delta^{m_\ell}$$

where δ is the failure probability of the wires, from (2.5) and (2.6) follows

$$\frac{p}{1-2\sqrt{p}} \geq \sum_{\ell \in S} \delta^{m_\ell} .$$

Then

$$\frac{p}{1-2\sqrt{p}} \geq s(\prod_{\ell \in S} \delta^{m_\ell})^{1/s}$$

by the inequality between the arithmetic and geometric means. Taking the logarithm we conclude

$$\sum_{\ell \in S} m_\ell \geq \frac{s}{\log(1/\delta)} \log(s\, \frac{1-2\sqrt{p}}{p}) . \tag{2.7}$$

Since the maximum fan-in of the gates of the circuit $n(\Phi_C)$ is constant, (2.7) means that the number of gates in the circuit is $\Omega(s \log s)$, and this completes the proof of the theorem.

2.1.6. Block sensitivity

Let f be a Boolean function of n variables, $x = (x_1, \ldots, x_n)$ any input and S any subset of indices, $S \subset \{1, \ldots, n\}$. Denote by x^S the input obtained from x by complementing all bits with indices from S and keeping the other bits of x unchanged.

Definition 2.1.12 The function f is *sensitive to* S on input x if $f(x) \neq f(x^S)$. The *block sensitivity of* f *on* x is the largest number b such that there exist b disjoint sets $S_1, \ldots, S_b$ such that for all $1 \leq i \leq b$, f is sensitive to S_i on x. The *block sensitivity* of f is the maximum over all x of the block sensitivity of f on x.

This measure of complexity was introduced by Nisan in [65]. Clearly for any function

$$\text{block sensitivity} \geq \text{sensitivity} .$$

It is shown in [65] that for all monotone functions the sensitivity equals the block sensitivity, but for non-monotone functions the inequality may be strict. A function with quadratic gap between sensitivity and block sensitivity is exhibited by Rubinstein [82].

Theorem 2.1.13 *Let ε and p be any constants so that $\varepsilon \in (0, 1/2)$, $p \in (0, 1/4)$. Let f be any Boolean function with block sensitivity b. If a circuit whose gates fail independently with fixed probability ε computes f with error probability at most p, then the number of gates of the circuit is at least $\Omega(b \log b)$.*

Proof:

Let the block sensitivity of f be maximum on input z, and let $S_1, \ldots, S_b$ be disjoint sets so that for all $1 \leq i \leq b$, f is sensitive to S_i on z. We can apply the proof of Theorem 2.1.4 by defining B_i for $1 \leq i \leq b$ as the set of all wires originating from the inputs with indices from S_i.

Corollary 2.1.14 *The redundancy of the noisy computation by Boolean circuits of any function of n variables with $O(n)$ noiseless complexity and $\Omega(n)$ block sensitivity is $\Omega(\log n)$.*

2.1.7. Open problems

Note that the $O(L \log L)$ upper bound construction [64, 29, 70] works for monotone circuits as well, since it can be realized using only gates computing the majority function in addition to the gates of the original noiseless circuit. Let $L^m(f)$ be the noiseless complexity of computing the monotone function f by monotone circuits. Theorem 2.1.4 shows that for some functions f, $\Omega(L^m(f) \log L^m(f))$ noisy gates are necessary for the reliable computation of f by monotone circuits. Is it still true that the redundancy of the noisy computation of almost all monotone functions by monotone circuits with noisy gates is constant? Andreev [9] showed that this is true for a different failure model, where the gates of the circuit do not fail independently, but the number of faulty gates is at most $2^{o(n)}$.

Considering arbitrary circuits, prove lower bounds stronger than $\Omega(n \log n)$ for the size of reliable circuits with noisy gates computing explicit functions of n

variables. This might be a very difficult problem: if such a lower bound holds for the computation of a function f by unrestricted circuits with gates from a finite complete basis, then the noiseless complexity of that function must be superlinear (in n). Thus exhibiting such a function would solve another fundamental open problem.

But this question is open even for restricted models: prove nonconstant redundancy of noisy computation for an explicit function known to have $\Omega(n \log n)$ noiseless complexity of computation by circuits with gates computing functions from some (incomplete) finite set Φ, for example by monotone circuits.

2.2. A model for tolerating adversarial faults

2.2.1. Description of the model

All the circuits considered here will be *synchronized*. This means that the gates are classified according to their levels numbered from 0 to $D(C)$, where $D(C)$ denotes the depth of the circuit. Wires go only between consecutive levels (a gate on the i^{th} level is fed from gates on the $(i-1)^{th}$ level for $i = 1, \ldots, D(C)$). We would like to build synchronized fault tolerant circuits that give the correct output even if at each level a small, but maliciously chosen constant fraction of the gates are malfunctioning.

Instead of trying to correctly compute the function on every input we define the *loose computation*

Definition 2.2.1 *For any computational device M we say that M δ-loosely computes f if*

1. *whenever $f(x) = 1$ then $M(x) = 1$.*
2. *If $f(z) = 0$ for every z with $d(x, z) \leq \delta n$, then $M(x) = 0$.*

Here $d(x, z)$ denotes the Hamming distance between words x and z. We remark that M can output an arbitrary value or no value at all if input x does not belong to the above two categories.

At the first sight it may appear that computational devices that loosely compute a function f can be far weaker than those that are able to get the value of f everywhere. For instance it is well known [3] that there is an AC_0 circuit that loosely computes the majority function, whereas majority itself cannot be computed in AC_0. One may speculate similarly that every function in NP might be loosely computed in P. Proposition 2.2.3 together with Remark 2.2.4 shows that this is not the case (unless $P = NP$).

Definition 2.2.2 *For an error correcting code E_n with codewords of length q_n and for a function $f : \{0,1\}^n \to \{0,1\}$ we define the function $f \diamond E_n : \{0,1\}^{q_n} \to \{0,1\}$ as follows.*

- *$(f \diamond E_n)(z) = 0$ for all z where z is not a codeword of the code E_n.*
- *If $z = E_n(x)$ then $(f \diamond E_n)(z) = f(x)$.*

Proposition 2.2.3 *If the Hamming distance of any two codewords in E_n is at least δq_n, and M is a computational device that computes $f \diamond E_n$ in a δ-loose manner, then $M(E_n(x)) = f(x)$ on any input x.*

Remark 2.2.4 *It is known from coding theory that there exist linear binary codes E_n with the following properties.*

1. *The matrix of E_n can be polynomially computed in n. (This also means that the length of the codeword, q_n, is polynomial in n.)*
2. *The Hamming distance of any two codewords in E_n is at least δq_n, for some small constant δ.*

There are two ways to interpret the assumption of our model about allowing at most a small constant fraction of the gates to be faulty at each level of the circuit. We may or may not allow the adversary to bias gates at the input level. The following proposition shows that we cannot expect to find a simple transformation that turns an arbitrary circuit into a fault tolerant circuit in our model in the harder case, when faults are allowed at the input level. We note that such a transformation exists in the model of independent random faults [70].

Proposition 2.2.5 *Assume that there is a polynomial time transformation that would turn any circuit C computing a function f into a circuit C' of size polynomial in the size of C such that C' loosely computes f, even if an adversary biases a small constant fraction of the gates at every level of C' including the input level. Then P=NP.*

Proof: We show that the hypothesis implies that the satisfiability of a 3CNF formula could be decided in polynomial time. Let ϕ be a 3CNF formula of n variables. Let us introduce n^2 new variables $y_1, \ldots, y_{n^2}$. Let C be the circuit that computes $f = y_1 \wedge \ldots \wedge y_{n^2} \wedge \phi$. Let C' be the fault tolerant circuit that computes f in a δ-loose manner. By our assumption we can build C' from C in polynomial time. Let z be the input $1 \ldots 1$. We claim that $C'(z)$=1 (without faults) if and only if ϕ is satisfiable. Indeed, if ϕ is not satisfiable then f is identically 0 and hence $C'(z) = 0$. On the other hand if there is an assignment a such that $\phi(a) = 1$ then by changing only at most $n \leq \delta n^2$ digits of z we can turn z into a z' such that $f(z') = 1$. Observe that loose computation preserves the 1 values. But then by the fault tolerant property of C', $C'(z) = 1$ holds. Thus building C' as above would allow us to differentiate between satisfiable and non satisfiable instances.

An observation of Gábor Tardos [92] shows that a similar statement also holds if we do not allow faults at the input level.

2.2.2. Halvers

Our construction for symmetric functions is based on the ϵ-halvers of Ajtai, Komlós and Szemerédi [4]. An ϵ-halver is a bounded depth comparator network with the property that for any set of the l smallest (largest) inputs, where $l \leq n/2$, at most ϵl elements will be among the last (first) $n/2$ outputs. In other words, an ϵ-halver is a halver (a network that separates the $n/2$ largest and the $n/2$ smallest inputs into two disjoint sets) that can misplace at most an ϵ fraction of the elements.

Constant depth linear size ϵ-halvers were constructed in [4]. More efficient ϵ-halvers (with smaller constants) based on units called k-comparators were proven to exist in [5], but no explicit construction has been given. A k-comparator is a unit that sorts its k inputs.

All our constructions work using either ϵ-halver. If the more efficient ϵ-halvers are used, we assume that each k-comparator is realized by depth k 2-comparator networks. We note that the depth of the realization of such ϵ-halvers by 2-comparator networks will be $2k$, since the ϵ-halvers of [5] are depth 2 k-comparator networks. In what follows by the depth of an ϵ-halver we always mean the depth of the realization by 2-comparator networks.

For $0-1$ inputs ϵ-halvers can be realized by monotone Boolean circuits. Each comparator will be realized by an AND - OR gate pair. The AND gate computes the minimum, the OR gate computes the maximum of the two common input bits. The resulting circuit will have the same depth as the original ϵ-halver. In what follows by ϵ-halvers we usually mean the Boolean circuits of the above mentioned form realizing the original ϵ-halvers.

For our purposes we have to consider ϵ-halvers with faulty gates.

Definition 2.2.6 *Denote by $g(y_1, y_2)$ the function that the gate g is supposed to compute on input y_1, y_2. A gate g is* faulty *if its output is different from $g(y_1, y_2)$.*

Definition 2.2.7 *Let C be a circuit with no faulty gates, and let $\tilde{C}$ be a copy of the same circuit with possibly faulty gates. The output of a gate of $\tilde{C}$ is* incorrect *if it is different from the value computed by the same gate in C. Note that the output of a gate may be incorrect because the gate is faulty, or because the inputs of the gate are incorrect.*

The following lemma says that in ϵ-halvers the incorrect outputs cannot cause too much damage on subsequent levels.

Lemma 2.2.8 *At level d of an ϵ-halver the number of incorrect outputs is at most the number of incorrect outputs at level $d-1$ plus the number of faulty gates at level d.*

Proof: Recall that a comparator in our ϵ-halvers is an AND - OR gate pair with common inputs. Level d of the ϵ-halver consists of disjoint comparators. Thus to prove the lemma it is enough to prove that the number of incorrect output bits of a comparator is at most the number of its incorrect input bits plus the number of its faulty gates. (Thus we only have to deal with a single comparator with two inputs and two outputs at a time.) Let us consider the case when there are no faulty gates and exactly one of the two input bits is incorrect. Notice that since the comparator only permutes the two input bits we can have only one incorrect bit in the output in this case. In all other cases the statement for a single comparator is trivially true.

Definition 2.2.9 *We say that a circuit is γ-*faulty *if at most γ fraction of the gates on each level is faulty.*

When talking about γ-faulty circuits we do not consider the inputs to be gates, but just values fed into the circuit. The reason for this is that our constructions will involve γ-faulty circuits connected to one another such that the outputs of one of them provide the inputs to others. Once we receive a set of values as output of a previous circuit it will be fed to the next circuit without further damage. (Another way of making this clear is noting that the wires of our circuits work correctly no matter what the adversary does.) However we will have to take into account that the adversary may destroy some of the real input gates of the whole construction.

For an ϵ-halver denote by L_1 (L_0) the number of $1's$ ($0's$) in the lower part of the output, and by U_1 (U_0) the number of $1's$ ($0's$) in the upper part of the output.

We shall need the observations below in the next section.

Lemma 2.2.10 *Consider a γ-faulty ϵ-halver of depth c with m inputs. Let the number of 0's in the input be a. If $a \geq m/2$ then*

$$L_1 \leq \epsilon(m-a) + c\gamma m$$

and

$$(a - m/2) - c\gamma m \leq U_0 \leq (a - m/2) + \epsilon m/2 + c\gamma m \ .$$

If $a \leq m/2$ *then*

$$U_0 \leq \epsilon a + c\gamma m$$

and

$$(m/2 - a) - c\gamma m \leq L_1 \leq (m/2 - a) + \epsilon m/2 + c\gamma m \ .$$

Proof: We first estimate what would be the value of L_1 and U_0 in a correct ϵ-halver. Lemma 2.2.8 guarantees that the number of incorrect outputs in the last level of a γ-faulty ϵ-halver of depth c is at most $c\gamma m$.

We summerize what we have learned in the following lemma:

Lemma 2.2.11 *Consider a 0-1 string of length* m*, such that there are* z *0's followed by* $m - z$ *1's. Denote by* z_L *(*z_U*) the number of 0's in the lower (upper) part of this ordered string.*

Consider a γ*-faulty* ϵ*-halver of depth* c *with* m *inputs. Let the number of 0's in the input be* a*, such that* $\mid a - z \mid \leq r$*. Then*

$$\mid U_0 - z_U \mid \leq r + (\epsilon + 2c\gamma)m/2$$

and

$$\mid L_0 - z_L \mid \leq r + (\epsilon + 2c\gamma)m/2 \ .$$

Note that a similar statement holds for the number of 1's in the output.

2.2.3. Symmetric functions

In this section we construct fault tolerant circuits for any symmetric function.

First we show that γ-faulty circuits can compute overwhelming majority functions correctly on all inputs where they are defined.

The overwhelming majority function Maj_k^n has value 1 or 0, respectively, if the number of 1's or the number of 0's in the input is at least k, and it is undefined otherwise.

Lemma 2.2.12 *Let $k > 3/4n$. Then for some $\gamma > 0$, there is a γ-faulty circuit of size $O(n)$ and depth $O(\log n)$ computing Maj_k^n correctly on every input belonging to its domain.*

Proof: The building blocks of our construction are triplets of ϵ-halvers with a majority preserving property. A similar component with this property was introduced by Assaf and Upfal in [12] for building fault tolerant sorting networks in the case of random faults. They call a comparator network with m inputs and $m/2$ outputs a *majority preserver* if it guarantees that if at least a given ($> 1/2$) constant fraction of the m inputs have the same value, then this value appears in at least the same given constant fraction of the $m/2$ outputs. We construct a majority preserver that tolerates a small constant fraction of worst case errors at each of its levels as follows.

First we apply an m-input ϵ-halver to the m inputs. Let us call it M-halver. Then we apply two $m/2$-input ϵ-halvers, one to the lower part the other to the upper part of the output of the M-halver. We refer to these two ϵ-halvers as L-halver and U-halver respectively. The output of the majority preserver will consist of the upper part of the output of the L-halver and of the lower part of the output of the U-halver.

We use a family of ϵ-halvers of depth c with the same parameters ϵ and c for all input lengths. For appropriate constants ϵ and c such family exists.

Let a be the number of 0's in the input of the majority preserver. Suppose

$$a \geq (3/4 + c\gamma)m \,. \tag{2.8}$$

Denote by $L_0(M)$ ($U_0(M)$) the number of 0's in the lower (upper) part of the output of the M-halver. We use similar notation for the L-halver and U-halver. By Lemma 2.2.10 and the inequality (2.8)

$$L_0(M) \geq m/2 - (\epsilon + 2c\gamma)m/2$$

and

$$U_0(M) \geq m/4 \, .$$

Applying Lemma 2.2.10 to both the L-halver and the U-halver, we get

$$U_0(L) + L_0(U) \geq (1 - 2(\epsilon + 2c\gamma))m/2 \, . \tag{2.9}$$

We have proved that if the number of 0's in the input of a majority preserver is at least $(3/4 + c\gamma)$ fraction of the input then at least a $(1 - 2(\epsilon + 2c\gamma))$ fraction of its output is also 0. A similar statement holds about the number of 1's.

We choose γ small enough to have $k > 3/4 + c\gamma$ and such that the following inequality holds:

$$1 - 2(\epsilon + 2c\gamma) \geq 3/4 + c\gamma \, . \tag{2.10}$$

Combining j majority preservers by feeding the outputs of one as inputs to the next we compute a set of size $n/2^j$ with the property that its overwhelming majority is the same as the overwhelming majority of the input. For $n/2^j < 1/\gamma$ we can finish the computation with a small circuit that has fewer than $1/\gamma$ gates at each of its levels, thus the adversary cannot destroy any of its gates. This circuit has to compute only the usual majority of its inputs, and that gives us the correct result.

Next we give a construction for any threshold function. The threshold function Th_k^n has value 1 if and only if at least k of the n input bits have value 1.

Theorem 2.2.13 *For any $\delta > 0$ there is a $\gamma > 0$ such that for any threshold function Th_k^n there is a synchronized circuit such that*

1. *If an adversary destroys a γ fraction of the gates on every level (including the input level), the circuit still computes Th_k^n in a δ-loose manner.*

2. The size of the circuit is $O(n)$.

3. The depth of the circuit is $O(\log n)$.

Proof: According to the definition of loose computation we have to build a circuit that outputs

- 1 if the number of 0's is $\leq n-k$
- 0 if the number of 0's is $\geq n-k+\delta n$
- an arbitrary value otherwise.

Let us consider the correctly ordered input and consider the set of elements at positions $n-k+1, \cdots, n-k+\delta n$. Notice that whenever our circuit must output 1 all these elements are 1, and whenever the circuit must output 0 all these elements are 0.

Choose t and l such that $ln/2^t \geq n-k$ and $(l+1)n/2^t \leq n-k+\delta n$. (We note that the value of t depends only on n and δ, but not on k. This will be important for the proof of Theorem 2.2.14.) Denote by T the set of elements at positions $ln/2^t+1, \cdots, (l+1)n/2^t$ of the correctly ordered input. Denote by z_T the number of 0's in the set T.

Our construction will consist of two parts. The first part denoted by C_1 will have n inputs and $n/2^t$ outputs. For any permutation of the input the number v of 0's in the output will satisfy

$$\mid v - z_T \mid \leq (\epsilon + 2c\gamma + \gamma)n\ .$$

ϵ and c are parameters of the ϵ-halvers we use. The second part will be the construction from Lemma 2.2.12 computing the overwhelming majority Maj_s^m of the outputs of C_1 where $m = n/2^t$ and $s = (1-2^t(\epsilon+2c\gamma+\gamma))m$.

For applying Lemma 2.2.12 we need

$$1 - 2^t(\epsilon + 2c\gamma + \gamma) \geq 3/4 + c\gamma$$

to hold, which is satisfied if $\delta/16 > \epsilon + 3c\gamma$. This inequality determines our possible choices of the parameters ϵ and γ. c will be determined by ϵ and the kind of ϵ-halvers we use. Note that the increase in size and depth relative to the simplest nonfaulty circuits will be decided by c.

The circuit C_1 will be a comparator network consisting of t γ-faulty ϵ-halvers each of depth c. The i-th ϵ-halver will have $n/2^{i-1}$ inputs and $n/2^i$ outputs. Let $l_1 \cdots l_t$ be the binary representation of l. If $l_i = 0$ then the input to the $i+1$-st ϵ-halver will be the lower part of the output of the i-th ϵ-halver. If $l_i = 1$ we take the upper part as input to the next ϵ-halver.

Let a_0 be the number of 0's in the correct input. Since the input of C_1 is provided by the input gates of the whole circuit, in our model the adversary is free to destroy a γ-fraction of the inputs of C_1 as well. Let a denote the number of 0-valued input gates. We have $a_0 - \gamma n \leq a \leq a_0 + \gamma n$. We can apply Lemma 2.2.11 to the first ϵ-halver with $r = \gamma n$. Repeatedly applying Lemma 2.2.11 we get

$$\mid v - z_T \mid \leq \gamma n + \sum_{i=1}^{t} (\epsilon + 2c\gamma) n/2^i \leq (\epsilon + 2c\gamma + \gamma) n$$

which concludes the proof of the theorem.

Theorem 2.2.14 *For any $\delta > 0$ there is a $\gamma > 0$ such that for any symmetric function f there is a synchronized circuit with the following properties.*

1. *If an adversary destroys a γ fraction of the gates on every level (including the input level), the circuit still computes f in a δ-loose manner.*

2. *The size of the circuit is $O(n)$.*

3. *The depth of the circuit is $O(\log n)$.*

Proof: Any symmetric function can be represented by a 0-1 string of length $n+1$, where the i-th element of the string is 1 if and only if $f = 1$ on inputs containing exactly i 1's.

If every set of consecutive 0's in this string has length $< \delta n$, then the δ-loose computation of f becomes trivial. (Giving the output 1 for every input is appropriate.)

Suppose the string representing f contains $h \geq 1$ sets of consecutive 0's each of length $\geq \delta n$. Let the i-th set be (l_i, u_i). For the δ-loose computation of f it is enough to δ-loosely compute each $\neg Th^n_{l_i}$ and each $Th^n_{u_i}$. We compute these $2h$ functions in parallel. We use the construction from Theorem 2.2.13 for δ-loosely computing threshold functions. The negation of a threshold function can be δ-loosely computed in an analogous way. We choose $\gamma = \tilde{\gamma}/2h$, and we choose $\tilde{\gamma}$ as required for the δ-loose computation of threshold functions in Theorem 2.2.13. We note that we choose the parameters δ, ϵ, c and $\tilde{\gamma}$ to have the same value for each of the $2h$ circuits. This guarantees that the number of gates at corresponding levels of the $2h$ circuits is exactly the same. Thus by destroying $\leq \gamma$ fraction of the gates at a given level, it is not possible to destroy more than a $\tilde{\gamma}$ fraction of the gates of a level in any one of the $2h$ circuits. Since $h < 1/\delta < 1/2\gamma$ after computing the $2h$ values in parallel we can combine them without any more faulty gates.

2.2.4. Fault tolerant circuits and probabilistically checkable proofs

Recently, the language class NP received a new characterization in terms of probabilistically checkable proofs. This characterization made it possible to show for the first time that to approximate the largest clique size of any graph within a constant factor is NP hard. In [10] it is shown that $NP = PCP(\log n, \log n)$. The class $PCP(f(n), g(n))$ is defined as the set of those languages L for which there is a randomized, polynomial time oracle machine $M^y(x, r)$ with oracle y, input x of length n and a random string r, such that:

1. $|r| = O(f(n))$.

2. For every fixed x and r, M reads only $O(g(n))$ bits of y.

3. If $x \in L$ then there is an oracle y such that $M^y(x, r) = 1$ with probability one over all r's.

4. If $x \notin L$ then for every oracle y the probability over all r's that $M^y(x, r) = 0$ is at least $3/4$.

The result of Arora and Safra [10] was improved by Arora et al. [11] to show that $NP = PCP(\log n, 1)$. Both proofs rely on relatively difficult techniques from classical algebra, coding theory and combinatorics. In what follows, we give a theorem about fault tolerant circuits in our model that imply the result in [10]. As appealing as it may sound, we do not get a simpler proof to the characterization of NP by Arora and Safra. Our construction uses the even harder result of [11].

Theorem 2.2.15 *Let C be a Boolean circuit and let $f : \{0,1\}^n \rightarrow \{0,1\}$ be the Boolean function computed by C. There exist a code $E = E_C$ and a circuit C' such that C' computes $f \diamond E$ in a δ-loose manner for every $\delta > 0$, even if an adversary destroys $\gamma(\delta) > 0$ fraction of the gates on every level (including the input level). Moreover E and C' have the following properties:*

1. *$|E(x)| \leq q(|x|)$ for some polynomial q independent of C.*

2. *The Hamming distance between any two codewords of E is at least $\delta_0|E|$ for some $0 < \delta_0 < 1$ independent of C. (δ_0 is a function of δ.)*

3. *$D(C') \leq O(\log S(C))$. (This implies that $S(C')$ is polynomial in $S(C)$.)*

4. *C' can be computed from C in polynomial time and $E(x)$ can be computed from C and x in polynomial time.*

Sketch of the proof: The proof is based on using the results of Arora et al. [11] and Theorem 2.2.13.

Let C^1 be the subset of $\{0,1\}^n$ for which C evaluates to 1 and C^0 be the set for which C evaluates to 0. In [11] an algorithm is described for turning a circuit C into a family $\{C_i\}_{i \in I}$ of constant size circuits with input $(G(x), Y)$, where $G(x)$ is an appropriate error correcting encoding of the input x to C and Y is an advise string (both have length polynomial in $S(C)$). This family has the property that

1. For every $x \in C^1$ there is a Y such that for all $i \in I$ we have $C_i(G(x), Y) = 1$.

2. For every $x \in C^0$ and for every Y we have that

$$\mathrm{Prob}_{i\in I}(C_i(G(x), Y) = 0) \geq 3/4 \ .$$

A modification by Lund and Spielmann [58] of the construction in [11] ensures that for every $x \in C^1$ the corresponding Y in condition 1 is unique. More precisely conditions 1 and 2 can be replaced by:

3. For every $x \in C^1$ there is a unique Y_x such that for all $i \in I$ we have $C_i(G(x), Y_x) = 1$. Also, for every $\delta > 0$ there exist $\epsilon > 0$ such that $\mathrm{Prob}_{i\in I}(C_i(H, Y) = 0) \leq \epsilon$ implies that $dist((H, Y), (G(x), Y_x)) \leq \delta|(H, Y)|$ for some $x \in C^1$.

We would like to build circuit C' such that it contains all members of the family $\{C_i\}_{i\in I}$. Because of technical reasons we want the members of the above family to have disjoint inputs. To achieve this we form $G'(x)$ and Y'_x by repeating the bits of $G(x)$ and Y_x as many times as the number of times for which they appear as input for some $\{C_i\}$ ($i \in I$). We note that this way each bit of $G(x)$ (and Y, respectively) will be repeated for the same number of times because of the symmetry of the construction in [11].

The code E is defined as follows. For $x \in C^1$ we define $E(x) = (G'(x), Y'_x)$. For $x \in C^0$ we define $E(x) = (G'(x), 0 \ldots 0)$.

We build C' as follows: We turn the family $\{C_i\}_{i\in I}$ into a family of synchronized circuits such that its members have all disjoint inputs and all outputs are at the same level. We include additional tests for checking that the groups of repeated bits indeed consist of identical bits. This is done by building a bounded degree expander over each group of input bits that must be identical and checking equality for every edge of these expanders. We denote this additional family of constant size circuits by $\{D_j\}_{j\in J}$. We synchronize these circuits with the members of the family $\{C_i\}_{i\in I}$ such that all outputs are at the same level. It will be also important for us that the size of I and J are the same within a constant factor ($|J| = \theta(|I|)$).

Observe that $\wedge_{i \in I} C_i(H, Y) \wedge \wedge_{j \in J} D_j(H, Y) = f \diamond E(H, Y)$. Using the construction described in Theorem 2.2.13 we build a fault tolerant circuit that computes the AND of the output bits of the circuits C_i for $i \in I$ and D_j for $j \in J$.

By condition 3. and by Theorem 2.2.13 this circuit will compute $f \diamond E$ in a δ-loose manner even if γ fraction of the gates is faulty at each level, including the input level, for appropriately chosen $\gamma = \gamma(\delta)$.

Remark 2.2.16 *The additional tests $\{D_j\}_{j \in J}$ are only necessary to tolerate faults that may occur at the input level. If we exclude the possibility of faults occurring at the input level, the additional tests may be left out from the construction.*

Theorem 2.2.17 *From Theorem 2.2.15 the theorem $NP = PCP(\log n, \log n)$ ([10]) follows.*

Proof: It is easy to see that $PCP(\log n, \log n) \subseteq NP$. What we need to show is that $NP \subseteq PCP(\log n, \log n)$ using Theorem 2.2.15. For this it is enough to show that 3SAT is in $PCP(\log n, \log n)$. Let ϕ be a 3SAT instance of n variables and C_ϕ be a circuit that computes ϕ. Clearly C_ϕ has size polynomial in n. We use Theorem 2.2.15 to turn C_ϕ into a fault tolerant circuit C'_ϕ with depth $O(\log n)$ that computes $\phi \diamond E$ for some code E. Observe that C'_ϕ outputs 1 on some input if and only if ϕ is satisfiable. We describe a checking procedure with a "prover" and a "verifier" as follows. (The verifier can be looked at as a polynomial time random oracle machine $M^y(x, r)$. The prover is the one to provide the verifier with advice y.)

We denote the i^{th} gate on the j^{th} level by $g(i, j)$. Let k_j denote the number of the gates on the j^{th} level. The prover's intention is to show that C'_ϕ outputs 1 on some input. He presents the verifier with the evaluation of all gates of C'_ϕ for an input that he claims is a satisfying assignment for C'_ϕ (i.e. C'_ϕ outputs 1 on this input).

The verifier chooses a random number r from 1 to $\max_j k_j$ and checks the value of the gates $\{g(r \, mod \, k_j, j)\}_{j=1..D(C'_\phi)}$ and the value of the gates that are inputs to these gates (two for each gate). The number of checkbits is at most $3D(C'_\phi) = O(\log n)$. The verifier accepts if the output bit has value 1 and if none of the gates in the set $\{g(r\, mod\, k_j, j)\}_{j=1..D(C'_\phi)}$ makes an error in the computation. Clearly, the true prover passes the test of the verifier with probability 1.

What is the probability that the cheating prover is caught? If at each level at most γ percent of the gates is faulty, then by the fault tolerant property of C'_ϕ the output is zero, thus the verifier rejects with probability one. Otherwise there is a level on which γ fraction of the gates is faulty. This will be revealed with probability at least $\gamma/2$ (we may loose a factor of at most 2 because k_l may not divide $\max_j k_j$). With a constant number of independent repetitions this probability can be brought above 3/4.

Remark 2.2.18 *We note that the proof of Theorem 2.2.17 does not require the full strength of Theorem 2.2.15. It would be enough to have a circuit C' that δ-loosely computes $f \diamond E$ for just some $\delta > 0$ in the presence of faults. We could have also relaxed the fault tolerant property of C' on the input level when proving Theorem 2.2.17. However, the fact that C' can be computed from C in polynomial time is crucial.*

2.2.5. Open problems

Theorem 2.2.15 (together with Proposition 2.2.3) shows that if we allow the input to be given in certain encoded form, we can obtain for any Boolean function f circuits with small redundancy computing f exactly and tolerating worst case faults in our model. However the encoding used in Theorem 2.2.15 depends on the function f, thus it does more than just encoding the input, it also encodes information about the value of the function. It would be interesting to achieve an analogous result using a code that does not depend on the function.

It is an interesting question whether a different construction could be used in the proof of Theorem 2.2.15. If there was a simpler encoding, that could be produced by an efficient fault tolerant computation, Theorem 2.2.15 could provide direct fault tolerant constructions for arbitrary functions.

On the other hand according to Theorem 2.2.17, a different proof of Theorem 2.2.15, that does not rely on constructions of probabilistically checkable proofs would provide an alternative technique for the area of probabilistically checkable proofs.

Finally, without giving constructions for arbitrary functions, it would be interesting to find direct fault tolerant constructions for other than symmetric functions.

CHAPTER 3

LOWER BOUNDS FOR BOOLEAN COMPLEXITY

3.1. Lower bounds for monotone span programs

3.1.1. Description of the model

The model of span programs was introduced by Karchmer and Wigderson in [50].

Let $\mathcal{F}$ be a field, and $\{x_1, \ldots, x_m\}$ be a set of variables. A *span program* over $\mathcal{F}$ is a labeled matrix $\hat{M}(M, \rho)$ where M is a matrix over $\mathcal{F}$, and ρ is a labeling of the rows of M by literals from $\{x_1, \ldots, x_m, \bar{x}_1, \ldots, \bar{x}_m\}$ (every row is labeled by one literal). The *size* of $\hat{M}$ is the number of rows in M.

A span program accepts or rejects an input by the following criteria. For every input sequence $\delta \in \{0,1\}^m$ define the submatrix M_δ of M consisting of those rows whose labels are set to 1 by the input δ, i.e., either rows labeled by some x_i such that $\delta_i = 1$ or rows labeled by some $\bar{x}_i$ such that $\delta_i = 0$. The span program $\hat{M}$ *accepts* δ if and only if $\mathbf{1} \in \text{span}(M_\delta)$, i.e., some linear combination of the rows of M_δ gives the vector $\mathbf{1}$. (The row vector $\mathbf{1}$ has the value 1 in each coordinate.) A span program *computes* a Boolean function f if it accepts exactly those inputs δ where $f(\delta) = 1$.

A span program is called *monotone* if the labels of the rows are only the positive literals $\{x_1, \ldots, x_m\}$. We denote by $\text{SP}_{\mathcal{F}}(f)$ (resp. $\text{mSP}_{\mathcal{F}}(f)$) the size of the smallest span program (resp. monotone span program) over $\mathcal{F}$ that computes f.

The number of columns does not effect the size of the span program. However, we observe that it is always possible to use no more columns than the size of the program (since we may restrict the matrix to a set of linearly independent columns without changing the function that is computed).

3.1.2. A linear upper bound

We present a monotone span program of linear size (exactly m) for a function on m variables, that is known to have $\Omega(m^{3/2}/(\log m)^3)$ monotone circuit complexity. We consider the function *Non-Bipartite*$_n$, whose input is an undirected graph on n vertices, represented by $m = \binom{n}{2}$ variables, one for each possible edge. The value of the function is 1 if and only if the graph is not bipartite.

Theorem 3.1.1 *$mSP_{GF(2)}(\text{Non-Bipartite}_n) = m$, where $m = \binom{n}{2}$.*

Proof: We construct a monotone span program accepting exactly the non-bipartite graphs as follows. There will be m rows, each labeled by a variable. There is a column for each possible complete bipartite graph on n vertices. The column for a given complete bipartite graph contains the value 0 in each row that corresponds to an edge of the given graph and contains 1 in every other row.

This program rejects every bipartite graph G. This is because G is contained in some complete bipartite graph, and so there will be a column that contains only 0's in the rows labeled by the edges of G. Therefore the vector $\mathbf{1}$ is not a linear combination of these rows.

Next we show that the program accepts every non-bipartite graph. Since the span program is monotone, it is sufficient to show that it accepts every *minimal* non-bipartite graph, i.e., every odd cycle. Let C be an arbitrary odd cycle. The intersection of any odd cycle with any complete bipartite graph has an even number of edges, so C has an odd number of edges which are *not* in any given complete bipartite graph. Hence the sum of the row vectors corresponding to all the edges in C is odd in each column, i.e., gives the vector $\mathbf{1}$ over GF(2), and so C is accepted by the span program.

We note that the lower bound by Razborov's method (see [76, 7, 48]) for triangles also applies to the function that accepts exactly the non-bipartite graphs, thus the monotone circuit complexity of the function *Non-Bipartite*$_n$ is $\Omega((n/\log n)^3) = \Omega(m^{3/2}/(\log m)^3)$.

3.1.3. The Method for Proving Lower Bounds

A *minterm* of a monotone function is a minimal set of its variables with the property that the value of the function is 1 on any input that assigns 1 to each variable in the set, no matter what the values of the other variables.

The idea of our technique is to show that if the size of a span program (i.e., the number of rows in the matrix) is too small, and the program accepts all the minterms of the function f then it must also accept an input that does not contain a minterm of f, which means that the program does not compute f.

We introduce the definition of a critical family of minterms of a monotone Boolean function. We prove that the cardinality of a critical family for a function f is a lower bound on the size of monotone span programs computing f.

Definition 3.1.2 *Let f be a monotone Boolean function and $\mathcal{M}_f$ be the family of all of its minterms. Let $\mathcal{H} \subseteq \mathcal{M}_f$ be a subfamily of the minterms of f. We say that a subfamily $\mathcal{H} \subseteq \mathcal{M}_f$ is a* critical family for f, *if every $H \in \mathcal{H}$ contains a set $T_H \subseteq H$, $|T_H| \geq 2$, such that the following two conditions are satisfied.*

C1. The set T_H uniquely determines H in the family. That is, no other set in the family $\mathcal{H}$ contains T_H.

C2. For any subset $Y \subseteq T_H$, the set $S_Y = \bigcup_{G \in \mathcal{H}, G \cap Y \neq \emptyset} G \setminus Y$ does not contain any member of $\mathcal{M}_f$.

Note that Condition C2 requires that S_Y does not contain any minterm of f and not just a minterm from $\mathcal{H}$.

Observation 3.1.3 *If $\mathcal{H}$ is a critical family and $|T_H| = t$ for each $H \in \mathcal{H}$, then $|\mathcal{H}| \leq \binom{m}{t}$.*

Theorem 3.1.4 *Let f be a monotone Boolean function, and let $\mathcal{H}$ be a critical subfamily of minterms for f. Then for every field $\mathcal{F}$,*

$$mSP_{\mathcal{F}}(f) \geq |\mathcal{H}| \ .$$

Proof. Let M be the matrix of a monotone span program computing f, and let r be the number of rows of M. Any minterm of $\mathcal{H}$ is accepted by the program. By definition, this means that, for every $H \in \mathcal{H}$, there is some vector $\mathbf{c}_H \in \mathcal{F}^r$ such that $\mathbf{c}_H \cdot M = \mathbf{1}$, and where $\mathbf{c}_H$ has nonzero coordinates only at rows labeled by variables from H. For any given H there may be several such vectors, we pick one of them and denote it by $\mathbf{c}_H$.

Since $\mathbf{c}_H$ is taken from $\mathcal{F}^r$, the number of linearly independent vectors among the vectors $\mathbf{c}_H$ for $H \in \mathcal{H}$ is a lower bound for r, i.e., for the size of the span program computing f. We show that all the vectors $\mathbf{c}_H$ for $H \in \mathcal{H}$ must be linearly independent.

Suppose, that this is not the case, i.e., for some $H \in \mathcal{H}$

$$\mathbf{c}_H = \sum_{A \in \mathcal{A}} \alpha_A \mathbf{c}_A \ , \tag{3.1}$$

where $\alpha_A \in \mathcal{F}$ and $\mathcal{A} = \mathcal{H} \setminus \{H\}$.

Let us consider the set $T_H \subseteq H$ from Definition 3.1.2.

Lemma 3.1.1 *If (3.1) holds, then for any nonempty subset $Y \subseteq T_H$ the following must hold.*

$$\sum_{A \in \mathcal{A}, A \cap Y \neq \emptyset} \alpha_A \ = 1 \ .$$

Proof: Suppose that for some $Y \subseteq T_H$, $\sum_{A \in \mathcal{A}, A \cap Y \neq \emptyset} \alpha_A \ = \gamma \neq 1$.

Let us consider the vector

$$\mathbf{c} = \sum_{A \in \mathcal{A}, A \cap Y \neq \emptyset} \alpha_A \mathbf{c}_A \ - \mathbf{c}_H \ .$$

We have $\mathbf{c} \cdot M = (\gamma - 1)\mathbf{1}$, thus $1/(\gamma - 1)\mathbf{c} \cdot M = \mathbf{1}$, and the program accepts the set of variables that label the rows corresponding to nonzero coordinates of $\mathbf{c}$.

Recall that each $\mathbf{c}_A$ has nonzero coordinates only at rows labeled by variables from A. Thus for $A \cap Y = \emptyset$ the coordinates of $\mathbf{c}_A$ are zero at rows labeled by variables from Y. By (3.1),

$$\mathbf{c} = \mathbf{0} - \sum_{A \in \mathcal{A}, A \cap Y = \emptyset} \alpha_A \mathbf{c}_A \ .$$

Therefore, the vector $\mathbf{c}$ has zero coordinates at all rows labeled by variables from Y.

On the other hand, all the nonzero coordinates of $\mathbf{c}$ are at rows labeled by variables that appear in some sets G such that $G \cap Y \neq \emptyset$. Therefore, the program accepts $S_Y = \bigcup_{G \in \mathcal{H}, G \cap Y \neq \emptyset} G \setminus Y$, that (by Definition 3.1.2) does not contain any minterm of f. This proves the lemma.

From Lemma 3.1.1, we get a system of linear equations in the unknowns α_A. We prove that this system of equations has no solution, contradicting (3.1). Suppose that $|T_H| = t$. Let us consider the following $(2^t - 1) \times (2^t - 1)$ 0–1 matrix Q. The rows and columns of Q are indexed by the nonempty subsets of T_H. For $\emptyset \neq Y, Z \subseteq T_H$, $Q(Y, Z) = 1$ if and only if $Y \cap Z \neq \emptyset$.

Observation 3.1.5 *The matrix Q has full rank over any field $\mathcal{F}$.*

(This can be shown by a simple transformation of Q to a triangular matrix.)

We will show, that if (3.1) holds, then taking $\beta_Z = \sum_{A \in \mathcal{A}, A \cap T_H = Z} \alpha_A$ as a coefficient for the column $Z \neq T_H$, we get the column indexed by T_H as a linear combination of the other columns of Q. Notice that the column of Q indexed by T_H consists of all 1's. We show that for any Y, $\emptyset \neq Y \subseteq T_H$, we have $\sum_{\emptyset \neq Z \subset T_H} \beta_Z Q(Y, Z) = 1$.

By Condition C1 of Definition 3.1.2, for $A \in \mathcal{A}$ we have $A \cap T_H \neq T_H$. If $Y \subseteq T_H$ then $A \cap Y = A \cap T_H \cap Y$. By Lemma 3.1.1, if (3.1) holds we have

$$1 = \sum_{A \in \mathcal{A}, A \cap Y \neq \emptyset} \alpha_A = \sum_{\emptyset \neq Z \subset T_H} \left(\sum_{A \in \mathcal{A}, A \cap T_H = Z, Z \cap Y \neq \emptyset} \alpha_A \right) = \sum_{\emptyset \neq Z \subset T_H} \beta_Z Q(Y, Z),$$

and the column T_H is a linear combination of the other columns of Q. Since Q has full rank this is not possible, and so (3.1) cannot hold, i.e., all the vectors $\mathbf{c}_H$ for $H \in \mathcal{H}$ are linearly independent. This concludes the proof of the theorem.

3.1.4. Lower bounds for clique functions

We consider the function $Clique_{k,n}$, whose input is an undirected graph on n vertices, represented by $m = \binom{n}{2}$ variables, one for each possible edge. The value of the function is 1 if and only if the graph contains a clique of size k.

It is known ([7, 76]) that the monotone circuit complexity of $Clique_{k,n}$ is $2^{\Omega(\sqrt{k})}$ for $k = O((n/\log n)^{2/3})$, and for fixed k it is $\Omega((n/\log n)^k)$. However, the strongest known lower bound for the monotone span program complexity of the $Clique_{k,n}$ function is our $\Omega(n^5) = \Omega(m^{2.5})$ lower bound that holds for $k \geq 6$. For $k \leq 4$, we obtain lower bounds that are tight up to a constant factor.

First we present a few simple but important observations that are helpful in finding critical families for clique functions.

For given k, we partition the set of n vertices into k classes $C_i, i = 1, \ldots, k$, of approximately equal size. Given a fixed partition of the n vertices into k classes we say that a k-clique is *multicolored* if each of its k vertices belong to a different class. Thus a multicolored clique will never contain an edge between two vertices in the same class.

Let $\mathcal{M}$ be an arbitrary family of multicolored k-cliques. Let T_K be some subset of the edges of the clique $K \in \mathcal{M}$. Let us denote the vertices of K by $v_1, \ldots, v_k$, and consider for $Y \subseteq T_K$ the set $S_Y = \cup_{G \in \mathcal{M}, G \cap Y \neq \emptyset} G \setminus Y$. Suppose S_Y contains a k-clique Z with vertices $z_1, \ldots, z_k$.

Claim 3.1.6 *The vertices of Z all belong to different classes, say $z_i \in C_i$, for $i = 1, \ldots, k$.*

Proof: S_Y only contains edges that appear in k-cliques that belong to the family $\mathcal{M}$, and so only edges between vertices from different classes.

Claim 3.1.7 *For each edge $(v_i, v_j) \in Y$ at least one of $z_i \neq v_i$ or $z_j \neq v_j$ must hold.*

Proof: If Z contained both v_i and v_j for $(v_i, v_j) \in Y$ then Z could not be a k-clique contained in S_Y since S_Y does not contain an edge between v_i and v_j.

Lemma 3.1.2 *Given any partition of the n vertices into three classes, the family $\mathcal{M}$ of multicolored 3-cliques is critical for* $Clique_{3,n}$.

Proof: Let H be an arbitrary multicolored 3-clique (triangle), and let T_H be the set of two of its edges, for example (v_1, v_2) and (v_2, v_3). There is only one triangle containing T_H, thus Condition C1 is satisfied. To see that Condition C2 holds, let us consider for $Y \subseteq T_H$ the set $S_Y = \cup_{G \in \mathcal{M}, G \cap Y \neq \emptyset} G \setminus Y$, and suppose that it contains a triangle Z with vertices z_1, z_2, z_3.

If $Y = T_H$, then $z_2 = v_2$ must hold, since there are no edges in S_Y incident to any other vertex from C_2. By Claim 3.1.7 we have $z_1 \neq v_1$ and $z_3 \neq v_3$. Therefore, the edge (z_1, z_3) cannot be present in S_Y, since all the edges of S_Y are contributed by triangles that contain at least one of v_1 or v_3.

If $Y \neq T_H$, then it consists of a single edge, (v_1, v_2) say. Then S_Y does not contain any edge between the classes C_1 and C_2, and so, by Claim 3.1.6, cannot contain a triangle.

Lemma 3.1.3 *Given any partition of the n vertices into four classes, the family of multicolored 4-cliques is critical for* $Clique_{4,n}$.

Proof: Let H be an arbitrary multicolored 4-clique, and let T_H be the set of two of its nonadjacent edges, for example (v_1, v_2) and (v_3, v_4). Condition C1 is satisfied, since two nonadjacent edges uniquely determine a 4-clique. To see that Condition C2 holds, as in the previous lemma, let us consider S_Y for $Y \subseteq T_K$ and suppose that it contains a 4-clique Z with vertices z_1, z_2, z_3, z_4.

If $Y = T_H$ then, by Claim 3.1.7, without loss of generality we have $z_1 \neq v_1$. Any edges incident to z_1 could only be contributed to S_Y by cliques that contain (v_3, v_4). Thus, a clique containing z_1 would also have to contain both v_3 and v_4, which is not possible by Claim 3.1.7.

As in the previous lemma, if $Y \neq T_H$ then it consists of a single edge and S_Y does not contain a 4-clique.

We note that for $k \geq 5$ the family of multicolored k-cliques is not critical for $Clique_{k,n}$. The critical families we use for proving lower bounds for 5- and 6-cliques will be appropriately chosen subfamilies of multicolored cliques.

Theorem 3.1.8 *For every field $\mathcal{F}$,*

$$mSP_{\mathcal{F}}(Clique_{6,n}) = \Omega(m^{2.5}) \; .$$

Proof: We show that the family of minterms of the $Clique_{6,n}$ function contains a large critical subfamily.

We define the subfamily $\mathcal{K}$ of 6-cliques as follows. We partition the set of n vertices into six approximately equal size classes. Let us assume that $n = 6q$, and each class contains q vertices. $\mathcal{K}$ will be a subfamily of the multicolored 6-cliques, under this partition. Between the classes C_1 and C_3, and similarly between the classes C_4 and C_6, we only allow certain pairs of vertices to be connected by an edge in members of $\mathcal{K}$. These *legal* pairs will be specified by a $q \times q$ Boolean matrix N. Between all other pairs of classes we allow arbitrary edges. The edge (a, b) with $a \in C_1$ and $b \in C_3$ ($a \in C_4$ and $b \in C_6$, respectively), is allowed in a member of $\mathcal{K}$ if and only if $N(a, b) = 1$. We choose N such that it does not contain any complete (all ones) 2×2 submatrices. For example the incidence matrix of a projective plane has this property, and its number of 1's is $\Theta(q^{3/2})$, with $\Theta(q^{1/2})$ 1's in each row and column. The constructions in [55, 63, 69] can also be used. (It is described in [69] how to construct matrices with similar properties for arbitrary q.)

The family $\mathcal{K}$ consists of all the 6-cliques that have one vertex from each class, and satisfy the restriction on the edges between classes C_1 and C_3, and C_4 and C_6. The number of such 6-cliques is $\Theta(q^5)$, thus we have $|\mathcal{K}| = \Theta(q^5) = \Theta(m^{2.5})$.

Next we show that $\mathcal{K}$ is critical for $Clique_{6,n}$. Let us consider any fixed member $K \in \mathcal{K}$ of the family, and denote its vertices by $v_1, \ldots, v_6$, where $v_i \in C_i$, for $i = 1, \ldots, 6$. The set T_K we choose will consist of the four edges (v_1, v_2), (v_2, v_3), (v_4, v_5), (v_5, v_6). Obviously, Condition C1 is satisfied.

We now prove that Condition C2 holds. For $Y \subseteq T_K$, suppose the set $S_Y = \cup_{G \in \mathcal{K}, G \cap Y \neq \emptyset} G \setminus Y$ contains a 6-clique Z with vertices $z_1, \ldots, z_6$.

Case 1. Let $Y = T_K$. Notice that if both $z_2 \neq v_2$ and $z_5 \neq v_5$, then S_Y does not contain an edge between z_2 and z_5, thus we have $z_2 = v_2$ or $z_5 = v_5$.

Suppose that only one of these equalities holds, for example $z_2 = v_2$ but $z_5 \neq v_5$. Then, by Claim 3.1.7, $z_1 \neq v_1$ and $z_3 \neq v_3$ must hold. The edge (z_1, z_5) can only be contributed to S_Y by a clique that contains the edge (v_2, v_3), and similarly the edge (z_3, z_5) can only be contributed to S_{T_K} by a clique that contains the edge (v_2, v_1). This means that the edges $(z_1, v_3), (v_1, z_3)$ as well as the edges (v_1, v_3) and (z_1, z_3) appear in some member of the family $\mathcal{K}$. However, this is not possible by our restriction on the legal edges between C_1 and C_3.

Suppose now that both $z_2 = v_2$ and $z_5 = v_5$ holds. Then by Claim 3.1.7 we have $z_1 \neq v_1$, $z_3 \neq v_3$, $z_4 \neq v_4$ and $z_6 \neq v_6$. The edge (z_1, z_4) can only be contributed by a clique that contains (v_2, v_3) or (v_5, v_6). This means that at least one of the edges (z_4, v_6) or (z_1, v_3) is legal. Similarly, from the presence in Z of the edges (z_1, z_6), (z_3, z_4) and (z_3, z_6), respectively, we know that at least one each of (v_4, z_6) or (z_1, v_3), (z_4, v_6) or (v_1, z_3), and (v_4, z_6) or (v_1, z_3), respectively, are legal edges. This means that either both (z_4, v_6) and (v_4, z_6) or both (z_1, v_3) and (v_1, z_3) are legal, and since (v_i, v_j) and (z_i, z_j) must be all legal, we get a contradiction with our restriction on the possible edges of members from $\mathcal{K}$.

Case 2. Let $Y \neq T_K$. In this case the edges in Y cover t vertices, $2 \leq t \leq 5$. We show that S_Y does not even contain a t-clique on the t classes involved. For $t \leq 4$ this directly follows from Lemma 3.1.2 and Lemma 3.1.3.

We still have to deal with the case when $t = 5$, which can only happen if Y consists of three edges. Suppose (without loss of generality) that the three edges of Y are (v_1, v_2), (v_2, v_3) and (v_4, v_5). If $z_2 \neq v_2$, then all the edges incident to z_2 could only be contributed to S_Y by cliques that contain (v_4, v_5). That would mean that the only vertex in C_4 and C_5, respectively, connected to z_2 in S_Y is v_4 and v_5, respectively. Thus we could not get a 6-clique in S_Y that contains z_2. Therefore, $z_2 = v_2$ must hold. Then we have by Claim 3.1.7 that $z_1 \neq v_1$, $z_3 \neq v_3$ and, without loss of generality, $z_5 \neq v_5$. We get a contradiction with the restriction on the edges between C_1 and C_3 as in Case 1.

We have proved that Condition C2 is also satisfied, and $\mathcal{K}$ is a critical family for f. The lower bound follows from Theorem 3.1.4.

Theorem 3.1.9 *For every field $\mathcal{F}$,*

$$mSP_{\mathcal{F}}(Clique_{5,n}) = \Omega(m^{2.25}) \ ,$$

$$mSP_{\mathcal{F}}(Clique_{4,n}) \geq 3(n/4)^4 - O(n^2) = \Omega(m^2) \ ,$$

$$mSP_{\mathcal{F}}(Clique_{3,n}) \geq 2(n/3)^3 - O(n) = \Omega(m^{\frac{3}{2}}) \ .$$

The proof of this theorem is basically included in the proof of the lower bound for 6-cliques and in Lemmas 3.1.2 and 3.1.3. The bounds for $Clique_{4,n}$ and $Clique_{3,n}$ are slightly stronger (by constant factors) than the bound directly implied by Theorem 3.1.4. We omit the details from this version of the paper. Our lower bounds for $Clique_{3,n}$ and $Clique_{4,n}$ are tight up to constant factors.

Let us define $Clique^*_{4,n}$ to be the monotone Boolean function whose set of minterms is the set of multicolored 4-cliques defined for a fixed partition of the vertices into four approximately equal classes. We observe that the above lower bound applies to this function as well, and is asymptotically tight in this case.

Corollary 3.1.10 *Let $n = 4q$. Then, for every field $\mathcal{F}$,*

$$3q^4 \leq mSP_{\mathcal{F}}(Clique^*_{4,n}) \leq 3q^4 + 3q^3 \ .$$

3.1.5. Dual span programs

We presented our lower bounds using certain combinatorial properties of the family of minterms of the function. We observe that the corresponding properties of the family of maxterms of the function imply the same lower bounds. This will follow from the theorem below.

A *maxterm* of a monotone function is a minimal set of its variables with the property that the value of the function is 0 on any input that assigns 0 to each variable in the set, no matter what the values of the other variables.

We say that a given input contains a set of variables, if each variable belonging to the set is evaluated to 1 by the given input.

Let f be a monotone Boolean function. Let $f^*(x_1, \ldots, x_m) = \neg f(\bar{x_1}, \ldots, \bar{x_m})$. It is not hard to see that the minterms of f^* are exactly the maxterms of f. (For example, if f accepts the graphs that contain a clique of size k, then f^* accepts the graphs whose complement does not contain any clique of size k.)

The following fact is well known. For completeness we give a proof.

Theorem 3.1.11 *For every field $\mathcal{F}$, $mSP_{\mathcal{F}}(f) = mSP_{\mathcal{F}}(f^*)$.*

Proof: Let M be a monotone span program computing f. Any minterm A of f is accepted by M. By definition, this means that $\mathbf{1} \in \text{span}(M_A)$, i.e. $\mathbf{c}_A \cdot M = \mathbf{1}$ for some vector $\mathbf{c}_A$, where $\mathbf{c}_A$ has nonzero coordinates only at rows labeled by variables from A.

We construct M^* computing f^* as follows. We use the same number of rows and the same labeling of the rows as M. The number of columns of M^* will be equal to the number of minterms of f. The columns of M^* will simply be the vectors $\mathbf{c}_A$ for each minterm A of f.

Claim 3.1.12 *M^* rejects every input that does not contain a maxterm of f.*

Proof: M^* rejects the complement of any minterm of f since the columns consists of only 0's on the complements of minterms of f. If a given input does not contain a maxterm of f then its complement contains a minterm of f and it will be rejected by M^*.

Claim 3.1.13 *M^* accepts every input that contains a maxterm of f.*

Proof: Let us suppose that σ is an input that contains a maxterm, and it is rejected by M^*. As observed in [50], by duality an input σ is rejected if and only if there is an affine combination of the columns of M^* that gives 0 on each row that is labeled by a variable that is set to 1 by σ. Let us denote the column vector that we get as a result of this affine combination by $\mathbf{a}$. We have $\mathbf{a} = \sum \alpha_A \mathbf{c}_A$ where $\sum \alpha_A = 1$.

Consider the input σ^* that assigns 1 to a given variable if and only if the above affine combination gives a nonzero value on at least one row labeled by the given variable. Then $f(\sigma^*) = 0$ since all the variables of some maxterm of f are set to 0 in σ^*.

For any given column $\mathbf{c}_A$ of M^* we have $\mathbf{c}_A M = \mathbf{1}$. It follows that $\mathbf{a}M = \mathbf{1}$. This shows that there is a linear combination of the rows of M labeled by variables that are set to 1 by σ^* that gives the vector $\mathbf{1}$. Thus M must accept σ^* and we get a contradiction, proving the claim.

The proof of the theorem follows from the above two claims.

3.1.6. Open problems

The $\Omega(m^{2.5})$ lower bound presented here for the 6-clique function is the largest known lower bound on the size of monotone span programs for an explicit function of m variables. In particular it is not known whether the k-clique function for larger values of k and the perfect matching function can be computed by polynomial size monotone span programs. Recall that every function that can be computed by polynomial size span programs belongs to NC^2 (this follows from [19, 24, 50, 60]). Thus it is unlikely that the k-clique function could be computed by polynomial size monotone span programs, but this would be plausible for the perfect matching function.

3.2. A lower bound for read-once branching programs

We present a Boolean function in n variables that is computable in depth 2 monotone AC^0 but requires $2^{\sqrt{n}}$ size read-once branching programs.

3.2.1. Relating branching program size to the number of subfunctions

Let f be a Boolean function on the set of variables X. Consider a partition of X into two parts Y and $B = X \setminus Y$. For every fixed assignment σ of truth values to the variables in Y we get a subfunction f_σ on the remaining variables. Let $N(f, Y)$ denote the number of different subfunctions we obtain under all possible assignments to Y. Note that both $2^{|Y|}$ and $2^{2^{|B|}}$ are upper bounds for $N(f, Y)$.

Some variant of the following observation has been used in most papers on this subject. It is a special case of general results in [46, 56, 86] and is in fact implicit already in the method of Wegener [99] and Dunne [30].

Lemma 3.2.1 *([99, 30, 46, 56, 86]) Let f be a Boolean function of n variables. Assume that m is an integer, $1 \leq m \leq n$, such that for any m-element subset Y of the variables $N(f, Y) = 2^m$ holds. Then the size of any read-once branching program computing f is at least $2^m - 1$.*

For completeness we include a proof.

Proof: Let Z be an arbitrary set of at most $m - 1$ variables, and let σ be a truth assignment to the variables in Z. Then the subfunction f_σ must depend on each of the remaining $n - |Z|$ variables, since otherwise Z could be extended to a set Y of m variables with $N(f, Y) < 2^m$. This also means that any path leading to a sink (ACCEPT or REJECT) in a branching program computing f must have length at least m.

Let v be an arbitrary node of a read-once branching program computing f. Suppose that there is a path P of length at most $m - 1$ leading from START to v. Let Y_P be the set of variables queried along the path P. We show that for any other path P' leading to v, $Y_{P'} = Y_P$ must hold. First we observe that the subprogram starting at v must depend on all the variables outside Y_P. This means that $Y_{P'} \subseteq Y_P$ since the branching program is read-once. But then we have $|Y_{P'}| \leq m - 1$, and by the same reasoning it follows that $Y_P \subseteq Y_{P'}$. We have shown that if a path leading to a vertex v queries at most $m-1$ variables, then any other path leading to the same vertex must query the exact same set of at most $m-1$ variables.

Let Z be an arbitrary set of at most $m-1$ variables. Suppose that there is a variable $z \in Z$ such that two different paths leading to the same vertex and querying exactly the variables in Z evaluate z differently. Then we could find an assignment σ to the variables in $Z \setminus \{z\}$ such that the subfunction f_σ does not depend on z, which is not possible.

From the above argument, it follows that the first $m-1$ levels of any read-once branching program computing f must form a complete binary tree.

3.2.2. An AC^0-computable function with exponential read-once branching program complexity

We use finite projective planes to exhibit an AC^0-computable function that requires exponential size read-once branching programs.

Let $\Pi = (P, L)$ be a projective plane of order q. (P is the set of points and L is the set of lines, viewed as subsets of P.) Let $n = q^2 + q + 1$ and $m = q + 1$. So $|P| = |L| = n$ and each line has m points.

We assign a variable x_i to each point $i \in P$, and define the following Boolean function.

Definition 3.2.1

$$f_\Pi(x_1, \ldots, x_n) = \bigwedge_{\lambda \in L} \bigvee_{i \in \lambda} x_i .$$

Theorem 3.2.2 *The size of any read-once branching program computing f_Π is at least $2^{\sqrt{n}}$.*

The proof is based on the following combinatorial property of projective planes.

Proposition 3.2.3 *Let $J = \{p_1, \ldots, p_m\}$ be a set of m distinct points of Π. Then there exist distinct lines $\lambda_1, \ldots, \lambda_m$ such that for $1 \le i, j \le m$ we have $p_i \in \lambda_j$ if and only if $i = j$.*

Proof: Recall that there are exactly m lines that contain any given point. Let us consider an arbitrary point $p_i \in J$, and the m lines that contain it. Since any two lines intersect in at most one point, each of the other $m-1$ points of the set J belong to at most one of these lines. Thus at least one of the m lines containing p_i will contain no other point from the set J.

Proof of Theorem 3.2.2: We show that for every q-element subset A of the variables, $N(f_\Pi, A) = 2^q$ holds, i.e. each truth assignment to the variables in A yields a different subfunction on the remaining variables. Since each line λ defines a clause $\bigvee_{i\in\lambda} x_i$ of the function f_Π, it follows from Proposition 3.2.3 that for an arbitrary q-element subset A of the variables there exist q clauses such that each variable from A appears in exactly one of them, and each variable appears in a different clause. Assume w.l.o.g. that $A = \{x_1, \ldots, x_q\}$, and the corresponding clauses are $\lambda_1, \ldots, \lambda_q$.

Let σ_1 and σ_2 be different truth assignements to the variables in A. Suppose they differ in the value of x_i, i.e., $x_i = 0$ in σ_1 and $x_i = 1$ in σ_2. Let us consider the assignment ξ to the variables outside A that sets each variable in the clause λ_i containing x_i to 0 and sets all other variables outside A to 1. Since any other clause λ_j has only one variable in common with the clause λ_i and there are at most $q-1$ variables in A that do not appear in λ_i, λ_j must contain at least one point which is neither in A nor in the clause λ_i thus it is set to 1 by ξ. Then we have $f_{\sigma_1}(\xi) = 0$ and $f_{\sigma_2}(\xi) = 1$. This shows that different truth assignments to the variables in A yield different subfunctions of f_Π.

The bound then follows from Lemma 3.2.1.

3.2.3. Open problems

It remains an open problem to find AC^0 computable function families that require $2^{\Omega(n)}$ read-once branching program size.

CHAPTER 4

BOOLEAN VS. ARITHMETIC CIRCUITS

Valiant and Vazirani [97] give a randomized reduction from SAT to $USAT$ (unique satisfiability). Given any Boolean formula F, their probabilistic construction yields another formula F', such that if F is not satisfiable then F' is not satisfiable, while if F is satisfiable then with reasonable ($\geq n^{-c}$) probability F' has a unique satisfying assignment. From this reduction, it follows that $NP/poly \subseteq \oplus P/poly$.

Avi Wigderson [101] proves that $NL/poly \subseteq \oplus L/poly$. The proof is based on a randomized reduction from $s-t$ connectivity to unique $s-t$ connectivity for directed graphs.

We prove that polynomial size semi-unbounded fan-in Boolean circuits of depth d with n inputs can be simulated by polynomial size semi-unbounded fan-in arithmetic circuits of depth $O(d+\log n)$, where the arithmetic operations $+$, $-$, $\times$ are performed in an arbitrary finite field.

We achieve the depth $O(d+\log n)$ simulation by a randomized reduction to "unique witnesses" using a slight modification of the Isolation Lemma of Mulmuley, Vazirani and Vazirani [61]. We note that the proof of [101] is based on using the Isolation Lemma, and that [61] showed that the Isolation Lemma can be used to prove the Valiant-Vazirani [97] result for the clique function.

4.1. The Isolation Lemma

We state the Isolation Lemma of Mulmuley, Vazirani and Vazirani [61]. Let us consider the set system $(V, \mathcal{F})$ where $V = \{v_1, \ldots, v_n\}$ is a finite set, and $\mathcal{F}$ is a family of distinct subsets of V, i.e. $\mathcal{F} = \{F_1, \ldots, F_k\}$, $F_j \subseteq V$ for $1 \leq j \leq k$. Given weights w_i to each element $v_i \in V$, we define the weight of the set F_j to be $\sum_{v_i \in F_j} w_i$.

Isolation Lemma [61] *If we choose integer weights uniformly and independently from* $[1, 2n]$ *then with probability* $\geq 1/2$ *there is a unique minimum weight set in* $\mathcal{F}$.

In this work we need a version of the Isolation Lemma that holds for multisets as well, i.e. for sets possibly containing some elements with multiplicities. The *support* of a multiset F is the set of elements occurring in F.

For multisets the Isolation Lemma does not hold in its original form. However the proof of the Isolation Lemma in [61] yields the following.

Lemma 4.1.1 *[61] Let* $(V, \mathcal{F})$ *be a multiset-system, where* $\mathcal{F}$ *is a family of multisets of the elements of* V*. Let us assign integer weights to the elements of* V *uniformly and independently from* $[1, 2|V|]$*. Then with probability* $\geq 1/2$*, all minimum weight sets in* $\mathcal{F}$ *have the same support.*

We note that Nisan [66] proved that the Isolation Lemma holds for multisets as well if we allow larger weights depending on the maximum multiplicity.

4.2. Semi-unbounded fan-in circuits

4.2.1. Definitions and the main results

We consider Boolean circuits with gates from the standard Boolean basis $\{\wedge, \vee, \neg\}$, and arithmetic circuits with gates from the basis $\{+, -, \times\}$ over a finite field F. *Semi-unbounded fan-in* circuits have constant fan-in $\wedge$ (resp. $\times$) gates and unbounded fan-in $\vee$ (resp. $+$) gates. For semi-unbounded fan-in Boolean circuits we allow negations only at the input level.

Complexity classes defined by semi-unbounded fan-in Boolean circuits have been characterized in terms of several other models, and they correspond to some well known language classes. SAC^k denotes the class of languages accepted by polynomial size, $O((\log n)^k)$ depth semi-unbounded fan-in Boolean circuits. The uniform version of SAC^1 is the same as the class $LOGCFL$ of languages logspace reducible to context-free languages [91], [95]. Properties and characterizations of $LOGCFL$ are studied in [25], [87]. Semi-unbounded fan-in circuits of larger depths correspond to extensions of context-free languages. For $d = \Omega(\log n)$ the class of languages accepted by polynomial size, depth $O(d)$ semi-unbounded fan-in circuits is identical to the class of languages accepted by nondeterministic auxiliary pushdown automata of $O(\log n)$ space and $2^{O(d)}$ time [25], [95]. There are other characterizations of these language classes in terms of alternating Turing machines and first order formulae [45], [83], [95]. These equivalences hold for both the uniform and the nonuniform versions of the models. For a survey see [22].

An interesting property of classes defined by semi-unbounded fan-in circuits, proved by Borodin et al. [22], is that they are closed under complementation for all depths that are $\Omega(\log n)$.

We consider the arithmetic analogs of the above complexity classes, defined by semi-unbounded fan-in arithmetic circuits. These classes have been studied for example in [6].

$\oplus SAC^k$ denotes the class of polynomials over $GF(2)$ computed by polynomial size, depth $O((\log n)^k)$ semi-unbounded fan-in arithmetic circuits over $GF(2)$.

We denote by $\hat{d}(f)$ (resp. $\hat{d}_F(f)$) the smallest depth of polynomial size semi-unbounded fan-in Boolean (resp. arithmetic) circuits for computing f.

We prove the following results.

Theorem 4.2.1 *For every Boolean function f on n variables and every finite field F, $\hat{d}_F(f) = O(\hat{d}(f) + \log n)$*

Corollary 4.2.2 $SAC^k \subseteq \oplus SAC^k$

Theorem 4.2.3 *Every Boolean function f on n variables can be approximated, i.e. computed on $\geq (1 - 2^{-k})$ fraction of the inputs, by semi-unbounded fan-in arithmetic circuits over any fixed finite field in polynomial size and depth $O(\hat{d}(f) + \log k)$, for arbitrary $k > 0$.*

4.2.2. Certificates

Let us consider an arbitrary Boolean circuit with $\vee$, $\wedge$ gates, provided with the $2n$ input literals $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$. Fixing a 0-1 assignment to the variables $x_1, \ldots, x_n$ determines the values computed by each gate of the circuit. We refer to the wires of the circuit as edges. The edges are oriented from the inputs of a gate g to g. For a fixed input assignment we label each edge of the circuit with the value computed by the gate at the starting node of the edge. For each gate that outputs 1, there is a set of edges all labelled 1 that forces the given gate to output 1. We call the graphs formed by these edges *certificates.* A certificate depends on the gate it belongs to and on the particular assignment to the input variables. There may be several certificates for the same gate on the same input. Gates that output 0 on a given input do not have certificates on that input.

Let us now give a formal definition of certificates. We denote the set of gates that are inputs to a given gate g by $I(g)$.

Definition 4.2.4 *The circuit Z is called a partial circuit of C if it satisfies the following conditions:*

- *the set of gates of Z is a subset of the set of gates of C*
- *the output gate of Z is the output gate of C,*
- *for every $\wedge$ gate g of Z, $I_Z(g) = I(g)$,*
- *for every $\vee$ gate g of Z, $\emptyset \neq I_Z(g) \subseteq I(g)$,*

where $I_Z(g)$ stands for the set of input gates to g in the circuit Z.

Definition 4.2.5 *A partial circuit Z is minimal, if for every $\vee$ gate g of Z, $|I_Z(g)| = 1$.*

Let α be a fixed assignment to the input variables. Let $\epsilon \in \{0, 1\}$. We say that $g(\alpha) = \epsilon$ if the gate g outputs the value ϵ on the input assignment α. We say that $C(\alpha) = \epsilon$ if the circuit C outputs the value ϵ on the input assignment α.

Observation 4.2.6 *If Z is a partial circuit of C then given any input assignment α, $Z(\alpha) \leq C(\alpha)$.*

Definition 4.2.7 *A certificate for $C(\alpha) = 1$ is a partial circuit Z of C that satisfies the condition that all the gates of Z output 1 on the assignment α.*

We note that an equivalent definition is to require that all the literals participating in Z are set to 1 by the assignment α.

Definition 4.2.8 *For a given gate g of a circuit C the subcircuit C_g is defined as follows:*

- *the set of gates of C_g is a subset of the set of gates of C,*
- *the output gate of C_g is the gate g,*
- *for every gate h of C_g, $I_{C_g}(h) = I(h)$.*

Definition 4.2.9 *A certificate for $C_g(\alpha) = 1$ is called a certificate for $g(\alpha) = 1$.*

We note that a certificate for $g(\alpha) = 1$ exists if and only if $g(\alpha) = 1$.

Definition 4.2.10 *A certificate is minimal if the corresponding partial circuit is minimal.*

Observation 4.2.11 *If there is a unique certificate for $g(\alpha) = 1$, it has to be a minimal certificate.*

Let G be a partial circuit of the circuit C, and suppose that the edges of C have been assigned weights. Let E be the set of edges of G. For a given partial circuit G with edges E we define a multiset $\tilde{E}(G)$ with support E as follows. We expand G into a tree $\tilde{G}$ by taking the output of G to be the root and by splitting the nodes of G that have outdegree ≥ 2 into several copies. We define $\tilde{E}(G)$ to be the multiset of edges of G, taking each edge with multiplicity according to $\tilde{G}$. We assign the weight of each edge of G to all of its copies in $\tilde{E}(G)$.

Definition 4.2.12 *We define the weight of a partial circuit G to be the weight of the multiset $\tilde{E}(G)$.*

We define the weight of a certificate to be the weight of the corresponding partial circuit.

Lemma 4.2.13 *Let us assign integer weights to the edges of the circuit C uniformly and independently from $[1, 2m]$, where m is the number of edges of C. Then for every fixed input assignment α such that $C(\alpha) = 1$, with probability $\geq 1/2$, there is a unique minimum weight certificate for $C(\alpha) = 1$.*

Proof The statement of the lemma follows from Lemma 4.1.1, since multisets with the same support belong to the same certificate.

4.2.3. The randomized reduction

Lemma 4.2.14 *Let C be a polynomial size, depth $d = O(\log n)$ circuit with unbounded fan-in $\vee$ gates and bounded fan-in $\wedge$ gates. Let m be the number of edges of C, and let c be the maximum fan-in of the $\wedge$ gates. One can construct a polynomial size, depth $\leq 2d$ circuit C' with unbounded fan-in $\vee$ gates and bounded fan-in $\wedge$ gates, such that*

- *if $C(\alpha) = 0$ then $C'(\alpha) = 0$*
- *if $C(\alpha) = 1$ then with probability $\geq 1/c^d 4m$, there is a unique certificate for $C'(\alpha) = 1$.*

Proof For simplicity, we present the construction for circuits with $\wedge$ gates of fan-in 2. It can be generalized easily to any fan-in c. The size remains polynomial if c is constant.

We assume that both C and C' are provided with the values of the $2n$ input literals $x_1, \ldots, x_n, \bar{x}_1, \ldots, \bar{x}_n$.

Let C be a Boolean circuit with $\vee$ gates of unbounded fan-in and $\wedge$ gates of fan-in 2, that has polynomial size and depth $d = O(\log n)$. Let m be the number of edges in the circuit. We assign a weight to each edge of the circuit by chosing a random integer uniformly and independently from $[1, 2m]$.

Next we choose a random integer L uniformly from $[1, 2^d 2m]$. ($2^d 2m$ is the maximum possible weight of a minimal certificate.)

Let us denote by $\Gamma(C)$ the set of gates of the circuit C.

$\Gamma(C')$ will consist of two disjoint classes of gates: principal gates and auxiliary gates. We denote the set of principal gates by Φ and the set of auxiliary gates by Φ_{aux}. Thus $\Gamma(C') = \Phi \mathbin{\dot{\cup}} \Phi_{aux}$. All principal gates (except the ones for the literals) will be $\vee$ gates and all auxiliary gates will be $\wedge$ gates.

Φ will be a subset of $\Gamma(C) \times \{1, \ldots, L\}$. For a fixed $g \in \Gamma(C)$ the set $H(g) \subseteq \{g\} \times \{1, \ldots, L\}$ will denote the set of all principal gates of the form (g, i), and $\Phi = \dot{\bigcup}_{g \in \Gamma(C)} H(g)$.

If g is an $\wedge$ gate of C then the gate $(g, i) \in H(g)$ will be associated with a set $A(g, i) \subseteq \Phi_{aux}$ of auxiliary gates where $0 \leq |A(g, i)| \leq L^2$.

We refer to the gates in the sets $H(g)$ and $A(g, i)$ as copies of g.

We construct the circuit C' inductively. We say that a gate $g \in \Gamma(C)$ has been *processed* if we have created all its copies in C'.

We start by processing the literals. For each literal x_i^ϵ we create a gate labelled $(x_i^\epsilon, 0)$. This gate will output the value of the corresponding literal x_i^ϵ. Each literal will have only one copy.

Let $g \in \Gamma(C)$ be an $\vee$ gate that has all its input gates processed. Let w_h denote the weight of the edge from $h \in I(g)$ to g. For each gate $h \in I(g)$ we consider the set $H(h)$. For each $(h, i) \in H(h)$ we create an $\vee$ gate $(g, i + w_h)$ if $i + w_h \leq L$ and if $(g, i + w_h)$ has not been created yet. The input gates to the $\vee$ gate (g, j) are the gates (h, i) with $h \in I(g)$ satisfying $i + w_h = j$.

Let $g \in \Gamma(C)$ be an $\wedge$ gate with input gates h_1, h_2 such that h_1 and h_2 have been processed. Let w_1 and w_2 denote the weights of the corresponding edges from h_1 and h_2, resp., to g. For each pair of gates $(h_1, i) \in H(h_1)$, $(h_2, j) \in H(h_2)$ such that $i + j + w_1 + w_2 \leq L$ we create an $\wedge$ gate in $A(g, i + j + w_1 + w_2)$ with input gates (h_1, i), (h_2, j). For each $k \in \{1, \ldots, L\}$ such that $A(g, k) \neq \emptyset$ we create an $\vee$ gate (g, k) with input set $I((g, k)) = A(g, k)$.

Let g_{out} be the output gate of the circuit C. The output gate of the circuit C' will be the gate (g_{out}, L) if it exists. If we did not create such a gate, we make the output of C' to be constant 0.

The circuit C' we constructed has the following properties.

Observation 4.2.15 *Let g be any gate of C. Then $(g, w) \in \Gamma(C')$ if and only if the circuit C_g has a minimal partial circuit with weight w.*

Observation 4.2.16 *Let α be a fixed assignment to the input variables and $(g, w) \in \Gamma(C')$. Then $(g, w)(\alpha) = 1$ if and only if there is a minimal certificate for $g(\alpha) = 1$ with weight w.*

We note that these properties would not hold if we defined the weight of partial circuits and certificates as the weight of their set of edges instead of using Definition 4.2.12, and that these properties are crucial for proving Lemma 4.2.14.

Next we show that the circuit C' constructed this way satisfies the requirements of Lemma 4.2.14.

The depth of C' is $\leq 2d$ and the size of C' is $\leq 2L^3S = n^{O(1)}$, where S is the size of C.

For any gate g of C and all gates (g, i) of C' we have $(g, i)(\alpha) \leq g(\alpha)$ on any assignment α. Thus if $C(\alpha) = 0$ then $C'(\alpha) = 0$.

To prove that the last requirement of Lemma 4.2.14 is satisfied, we need the following lemma.

Let α be a fixed input assignment such that $C(\alpha) = 1$. Let $\mathcal{F}_\alpha \neq \emptyset$ be the family of all certificates for $C(\alpha) = 1$. Let W denote a fixed assignment of weights to the edges of C. Let $\rho(W, \alpha)$ denote the weight of the minimum weight certificate in $\mathcal{F}_\alpha$.

Lemma 4.2.17 *Suppose there is a unique minimum weight certificate in $\mathcal{F}_\alpha$ and suppose that $L = \rho(W, \alpha)$. Then there is a unique certificate for $C'(\alpha) = 1$.*

Proof From the conditions of the lemma it follows that there is a minimal certificate of weight L for $C(\alpha) = 1$. By Observations 4.2.15 and 4.2.16 this means that in the circuit C' there is a gate (g_{out}, L) (where g_{out} is the output gate of C), and that $C'(\alpha) = (g_{out}, L)(\alpha) = 1$.

Thus, there is a certificate for $C'(\alpha) = 1$. If it is not unique, then there is more than one minimal certificate for $C(\alpha) = 1$ with weight L, and we get a contradiction. This proves Lemma 4.2.17.

The probability that there is a unique minimum weight certificate in $\mathcal{F}_\alpha$ and $L = \rho(W, \alpha)$ is at least $1/2 \cdot 1/(2^d 2m)$ (by Lemma 4.2.13 and by choosing L uniformly from $[1, 2^d 2m]$). Thus, we proved that if $C(\alpha) = 1$ then with probability $\geq 1/2^d 4m$ there is a unique certificate for $C'(\alpha) = 1$. This concludes the proof of Lemma 4.2.14.

4.2.4. The simulation

Proof of Theorem 4.2.1 Let us first consider the case of depth $d = O(\log n)$ circuits. By a standard probabilistic argument it follows from Lemma 4.2.14 that there exist $T = c^d 8nm = n^{O(1)}$ circuits $C^1, \ldots, C^T$ such that the following is true for every input assignment α

- if $C(\alpha) = 0$ then $\forall i \in \{1, \ldots, T\}$, $C^i(\alpha) = 0$,
- if $C(\alpha) = 1$ then $\exists j \in \{1, \ldots, T\}$ such that there is a unique certificate for $C^j(\alpha) = 1$.

Now we are ready to construct the simulating arithmetic circuit. In each circuit C^i we replace each $\vee$ gate by a $+$ gate and each $\wedge$ gate by a $\times$ gate. We denote the new circuit by $\diamondsuit C^i$. We let the circuits $\diamondsuit C^1, \ldots, \diamondsuit C^T$ compute in parallel over a common input $x_1, \ldots, x_n, 1 - x_1, \ldots, 1 - x_n$. To their outputs we apply a transformation that turns any value that is different from 1 into 0, and keeps the values that are equal to 1 unchanged. Over a given finite field this takes a constant number of gates for each $\diamondsuit C^i$. To compute the $\vee$ of these values, recall that the $\vee$ of

T variables can be represented by a polynomial of degree T over the given field. We compute this polynomial by a $\log T$ depth semi-unbounded fan-in circuit with $+$ and $\times$ gates. Let us denote the circuit obtained this way by $\diamondsuit C$.

The circuits $\diamondsuit C^1, \ldots, \diamondsuit C^T$ have the following property:

- if $C(\alpha) = 0$ then $\forall i \in \{1, \ldots, T\}$, $\diamondsuit C^i(\alpha) = 0$,
- if $C(\alpha) = 1$ then $\exists j \in \{1, \ldots, T\}$ such that $\diamondsuit C^j(\alpha) = 1$.

This follows from the corresponding properties of the circuits $C^1, \ldots, C^T$ and from the fact that if there is a unique certificate for $C^j(\alpha) = 1$ then $\diamondsuit C^j(\alpha) = 1$.

We conclude that on any input assignment α, $C(\alpha) = \diamondsuit C(\alpha)$. This proves the theorem if $d = O(\log n)$.

For larger d we divide the circuit C into $r = \log n$ depth parts and perform the above simulation on each part. The total depth of the simulating circuit will be $\leq (d/r) \cdot (2r + O(\log n)) = O(d + \log n)$, which concludes the proof of Theorem 4.2.1.

For proving Theorem 4.2.3 we construct the circuits $\diamondsuit C^1, \ldots, \diamondsuit C^T$ and transform their outputs to Boolean values as above. To achieve the $O(d + \log k)$ depth for approximate simulations over a fixed finite field we use polynomials of degree k that approximate the $\vee$ of these values. By [77] (cf. [88], Lemma 1) given any probability distribution on the 2^T inputs there exist polynomials of degree k that compute the $\vee$ of T variables with probability $\geq (1 - 2^{-k})$ over the input distribution. This proves Theorem 4.2.3.

4.2.5. Open problems

It is an interesting open question whether analogous relations hold for uniform circuit classes. Similarly, it is not known whether one can remove the nonuniformity from the Valiant and Vazirani [97] or the Wigderson [101] results. We note that a result in this direction has appeared in [50], proving that $SL \subseteq \oplus L$, where SL stands for symmetric logspace.

REFERENCES

[1] M. Ajtai, "$\sum_1^1$-formulae on finite structures", *Annals of Pure and Applied logic* 24, 1983, pp. 1-48.

[2] M. Ajtai, L. Babai, P. Hajnal, J. Komlós, P. Pudlak, V. Rödl, E. Szemerédi and G. Turán, "Two lower bounds for branching programs", In *Proceedings of the 18th ACM STOC*, 1986, pp. 30-38.

[3] M. Ajtai, M. Ben-Or, "A theorem on probabilistic constant depth computations," In *Proc. of "16-th ACM Symposium on Theory of Computing"*, 1984, pp. 471-474.

[4] M. Ajtai, J. Komlós, E. Szemerédi, "An $O(n \log n)$ sorting network," In *Proc. of "15-th ACM Symposium on Theory of Computing"*, 1983, pp. 1-9.

[5] M. Ajtai, J. Komlós, E. Szemerédi, "Halvers and expanders," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 686-692.

[6] E. Allender and J. Jiao, "Depth reduction for noncommutative arithmetic circuits," In *Proc. of the 25th STOC*, 1993, pp. 515-522.

[7] N. Alon and R. Boppana. "The monotone circuit complexity of Boolean functions". *Combinatorica*, 7, 1987, pp. 1-22.

[8] A. E. Andreev: "On a method for obtaining lower bounds for the complexity of individual monotone functions", *Sov. Math. Dokl.*, 31, 1985 pp. 530-534.

[9] A. E. Andreev: " Circuit synthesis in complete monotone basis", *Mat. Vopr. Kibern.* 1, 1988, pp. 114-139.

[10] S. Arora and S. Safra, "Probabilistic checking of proofs," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 2-13.

[11] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, "Proof verification and hardness of approximation problems," In *Proc. of "33-rd IEEE Symposium on the Foundations of Computer Science"*, 1992, pp. 14-23.

[12] S. Assaf, E. Upfal, "Fault tolerant sorting network", In *Proc. of "31-st IEEE Symposium on the Foundations of Computer Science"*, 1990, pp. 275-284.

[13] L. Babai, "Transparent proofs and limits to approximation," University of Chicago Technical Report CS93-15, 1993.

[14] L. Babai, P. Hajnal, E. Szemerédi and G. Turán, "A lower bound for read-once branching programs", JCSS v. 35 n.2, Oct. 1987 pp. 153-162.

[15] A. Beimel, "Ideal Secret Sharing Schemes", Master's thesis, Technion - Israel Institute of Technology, Haifa, 1992. (In Hebrew, abstract in English).

[16] A. Beimel and B. Chor, "Universally ideal secret sharing schemes", *IEEE Transactions on Information Theory*, IT-40, 3, 1994, pp. 786-794.

[17] A. Beimel, A. Gál and M. Paterson, "Lower bounds for monotone span programs", Technical Report BRICS-RS-94-46, BRICS, Department of Computer Science, University of Aarhus, December 1994.

[18] A. Beimel, A. Gál and M. Paterson. "Lower bounds for monotone span programs", To appear in *Proceedings of FOCS'95.*

[19] S. J. Berkowitz. "On computing the determinant in small parallel time using a small number of processors", *Inform. Process. Lett.*, 18, 1984, pp. 147–150.

[20] N. Blum, "A Boolean function requiring $3n$ network size", *Theoretical Computer Science*, 28, 1984, pp. 337-345.

[21] A. Borodin, Personal communication.

[22] A. Borodin, S. A. Cook, P. W. Dymond, W. L. Ruzzo, M. Tompa, "Two applications of inductive counting for complementation problems," *SIAM J. Comput.*, Vol. 18, No. 3, 1989, pp. 559-578.

[23] A. Borodin, A. Razborov and R. Smolensky, "On lower bounds for read-k-times branching programs", *Computational Complexity*, 3, 1993, pp. 1-18.

[24] G. Buntrock, C. Damm, H. Hertrampf, and C. Meinel, "Structure and importance of the logspace-mod class", *Math. Systems Theory*, 25, 1992, pp. 223-237.

[25] S. A. Cook, "A taxonomy of problems with fast parallel algorithms", *Information and Control*, 64, 1985, pp. 2-22.

[26] L. Csirmaz. "The dealer's random bits in perfect secret sharing schemes", Preprint, 1994.

[27] M. van Dijk, "A linear construction of perfect secret sharing schemes", In A. De Santis, editor, *Advances in Cryptology – Eurocrypt 94, pre-proceedings*, 1994, pp. 23-36.

[28] R. L. Dobrushin and S. I. Ortyukov, "Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.* 13, 1977, pp. 59-65.

[29] R. L. Dobrushin and S. I. Ortyukov, "Upper bound for the redundancy of self-correcting arrangements of unreliable functional elements," *Prob. Inf. Trans.* 13, 1977, pp. 203-218.

[30] P. E. Dunne, "Lower bounds on the complexity of 1-time only branching programs", in FCT 85, Lecture Notes in CS n. 199, Springer, New York, 1985, pp. 90-99.

[31] W. Evans, "Information theory and noisy computation", Ph. D. Thesis, University of California at Berkeley, ICSI Technical Report TR-94-057, 1994.

[32] W. Evans, L. Schulman, "Signal propagation with application to a lower bound on the depth of noisy formulas", In *Proc. of "34th IEEE Symposium on the Foundations of Computer Science"*, 1993, pp. 594-603.

[33] W. Evans, L. Schulman, "Information theory and noisy computation", abstract for IEEE International Symposium on Information Theory, 1995.

[34] U. Feige, D. Peleg, P. Raghavan and E. Upfal, "Computing with unreliable information", In *Proc. of "22nd ACM Symposium on the Theory of Computing"*, 1990, pp. 128-137.

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

[36] P. Gács: Unpublished letter, 1981.

[37] P. Gács and A. Gál, "Lower Bounds for the Complexity of Reliable Boolean Circuits with Noisy Gates," *IEEE Trans. Inform. Theory*, Vol. 40, No. 2, 1994, pp. 579-583.

[38] A. Gál, "Lower Bounds for the Complexity of Reliable Boolean Circuits with Noisy Gates," In *Proc. of "32-nd IEEE Symposium on the Foundations of Computer Science"*, 1991, pp. 594-601.

[39] A. Gál, "Semi-unbounded fan-in circuits: Boolean vs. arithmetic" In *Proceedings of the 10th Annual Structure in Complexity Theory*, 1995, pp. 82-87.

[40] A. Gál, "A simple function that requires exponential size read-once branching programs", University of Chicago Technical Report, TR-95-09, 1995.

[41] A. Gál and M. Szegedy, "Fault tolerant circuits and probabilistically checkable proofs", In *Proceedings of the 10th Annual Structure in Complexity Theory*, (1995), pp. 65-73.

[42] A. Gál and A. Wigderson, "Boolean complexity classes vs. their arithmetic analogs", Submitted to *Random Structures and Algorithms.*

[43] M. Goldman and J. Hastad, "A simple lower bound for monotone clique using a communication game", *Information Processing Letters* 41, 1992, pp. 221-226.

[44] J. Hastad, "Almost optimal lower bounds for small depth circuits", In, *Advances in Computing Research, 5: Randomness and Computation*, Micali, S., ed. JAI Press, 1989, pp. 143-170.

[45] N. Immerman, "Upper and lower bounds on first order expressibility," *J. Comput. System Sci.*, 25, 1982, pp. 76-98.

[46] S. Jukna, "Entropy of contact circuits and lower bound on their complexity", *Theor. Comput. Sci.*, 47:2, 1988, pp. 113-129.

[47] S. Jukna, "A note on read-k-times branching programs", Technical report 448, Universitat Dortmund, 1992, journal version: RAIRO Theoretical Informatics and Applications, vol. 29, Nr. 1 (1995), pp. 75-83.

[48] M. Karchmer, "On proving lower bounds for circuit size", In *Proceedings of the 8th Annual Structure in Complexity Theory*, 1993, pp. 112-118.

[49] M. Karchmer and A. Wigderson, "Monotone circuits for connectivity require superlogarithmic depth", *SIAM J. on Discr. Math.*, 3:2, 1990, pp. 255-265.

[50] M. Karchmer and A. Wigderson, "On span programs", In *Proceedings of the 8th Annual Structure in Complexity Theory*, 1993, pp. 102-111.

[51] M. Karchmer and A. Wigderson, "Characterizing non-deterministic circuit size", In *Proc. of the 25th STOC*, 1993, pp. 532-540.

[52] G. I. Kirienko, "On self-correcting schemes from functional elements," *Probl. Kibern.* 12, 1964, pp. 29-37.

[53] G. I. Kirienko, "Synthesis of self-correcting schemes from functional elements for the case of growing number of faults in the scheme," *Diskret. Anal.* 16, 1970, pp. 38-43.

[54] D. Kleitman, T. Leighton and Y. Ma, "On the design of reliable Boolean circuits that contain partially unreliable gates," In *Proc. of "35-th IEEE Symposium on the Foundations of Computer Science"*, 1994, pp. 332-346.

[55] T. Kővári, V. T. Sós and P. Turán, "On a problem of K. Zarankiewicz", *Colloq. Math.*, 3, 1954, pp. 50-57.

[56] M. Krause, "Exponential lower bounds on the complexity of real time and local branching programs", *J. Inform. Proc. Cybern.* (EIK), 24:3, 1988, pp. 99-110.

[57] M. Krause, C. Meinel and S. Waack, "Separating the eraser Turing machine classes L_e, NL_e, $co-NL_e$ and P_e", Theor. Comput. Sci., 86 (1991), pp. 267-275.

[58] C. Lund, D. Spielmann, Personal communication.

[59] O. B. Lupanov, "On a method of circuit synthesis", *Izv. VUZ Radiofizika*, 1, 1958, pp. 120-140.

[60] K. Mulmuley, "A fast parallel algorithm to compute the rank of a matrix over an arbitrary field", *Combinatorica*, 7, 1987, pp. 101-104.

[61] K. Mulmuley, U. Vazirani and V. Vazirani, "Matching is as easy as matrix inversion," In *Proc. of the 19th STOC*, 1987, pp. 345-354.

[62] D. E. Muller, "Complexity in electronic switching circuits", *IRE Trans. Electr. Comput.*, 5, 1956, pp. 15-19.

[63] E. I. Nečiporuk, "On a Boolean matrix", *Problemy Kibernet.*, 21, 1969, pp. 237-240. English translation in *Systems Theory Res.*, 21, 1971, pp. 236-239.

[64] J. von Neumann," Probabilistic logics and the synthesis of reliable organisms from unreliable components", In *Automata Studies*, C. E. Shannon and J. McCarthy Eds., Princeton University Press, Princeton, NJ, 1956, pp. 329-378.

[65] N. Nisan, "CREW PRAMs and decision trees", In *Proc. of "21-st ACM Symposium on the Theory of Computing"*, 1989, pp. 327-335.

[66] N. Nisan, Personal communication.

[67] W. Paul, "A 2.5n-lower bound on the combinatorial complexity of Boolean functions, *SIAM J. Comp.*, Vol.6, 1977, No. 3, pp. 427-443.

[68] N. Pippenger, "On simultaneous resource bounds", In *Proc. of "20th IEEE Symposium on the Foundations of Computer Science"*, 1979, pp. 307-311.

[69] N. Pippenger, "On another Boolean matrix", *Theoretical Computer Science*, 11, 1980, pp. 49-56.

[70] N. Pippenger, "On networks of noisy gates", In *Proc. of "26-th IEEE Symposium on the Foundations of Computer Science"*, 1985, pp. 30-36.

[71] N. Pippenger, Developments in "The Synthesis of Reliable Organisms from Unreliable Components", In *Proc. of "Symposia in Pure Mathematics"*, Vol. 50, 1990, pp. 311-324.

[72] N. Pippenger and M. Fischer, "Relationships among complexity measures", IBM Research Report RC-6569 (1977), Yorktown Heights.

[73] N. Pippenger, G. D. Stamoulis and J. N. Tsitsiklis, "On a lower bound for the redundancy of reliable networks with noisy gates", *IEEE Trans. Inform. Theory*, vol. 37, no. 3, 1991, pp. 639-643.

[74] S. Ponzio, "A lower bound for integer multiplication with read-once branching programs", In *Proceedings of the 27th STOC*, 1995, pp. 130-139.

[75] P. Pudlak and S. Zak, "Space complexity of computations", Tech. Rep. Univ. Prague, 1983.

[76] A. A. Razborov, "Lower bounds for the monotone complexity of some Boolean functions", *Sov. Math. Dokl.*, 31, 1985, pp. 354-357.

[77] A. A. Razborov, "Lower bounds for the size of circuits of bounded depth with basis $\{\wedge, \oplus\}$," *Math. notes of the Academy of Sciences of the USSR*, 41(4), 1987, pp. 333-338.

[78] A. A. Razborov, "On the method of approximation", In *Proceedings of the 21st ACM Symposium on Theory of Computing*, 1989, pp. 167-176.

[79] A. A. Razborov, "Lower bounds for deterministic and nondeterministic branching programs", In *Proceedings of the 8th FCT, Lecture Notes in Computer Science*, 529, 1991, pp. 47-60.

[80] R. Raz and A. Wigderson, "Monotone circuits for matching require linear depth", In *Proc. of the 22nd STOC*, 1990, pp. 287-292.

[81] R. Reischuk, B. Schmeltz, "Reliable Computation with Noisy Circuits and Decision Trees – A General n log n Lower Bound," In *Proc. of "32-nd IEEE Symposium on the Foundations of Computer Science"*, 1991, pp. 602-611.

[82] D. Rubinstein, "Sensitivity vs. block sensitivity of Boolean functions," *Combinatorica*, 15 (2), 1995, pp. 297-299.

[83] W. L. Ruzzo, "Tree-size bounded alternation," *J. Comput. System Sci.*, 21, 1980, pp. 218-235.

[84] W. L. Ruzzo, "On uniform circuit complexity", *J. Comput. System Sci.*, 22, 1981, pp. 365-383.

[85] C. E. Shannon, "The synthesis of two-terminal switching circuits", *Bell Syst. Techn. J.*, 28, 1949, pp. 59-98.

[86] J. Simon and M. Szegedy, “A new lower bound theorem for read-only-once branching programs and its applications”, In *Advances in Computational Complexity Theory*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 13, 1993, pp. 183-193.

[87] S. Skyum and L. G. Valiant, “A complexity theory based on Boolean algebra,” In *Proc. of the 22nd FOCS*, 1981, pp. 244-253.

[88] R. Smolensky, “Algebraic methods in the theory of lower bounds for Boolean circuit complexity,” In *Proc. of the 19th STOC*, 1987, pp. 77-82.

[89] P. M. Spira, “On time-hardware complexity tradeoffs for Boolean functions”, In *Proc. of the 4th Hawaii Symp. on Syst. Sc.*, 1971, pp. 525-527.

[90] L. Stockmeyer, “On the combinatorial complexity of certain symmetric Boolean functions”, *Mathematical Systems Theory*, Vol. 10, 1977, pp. 323-336.

[91] I. H. Sudborough, “On the tape complexity of deterministic context-free languages,” *J. Assoc. Comput. Mach.*, 25, 1978, pp. 405-414.

[92] G. Tardos, Personal communication.

[93] D. Uhlig, “Reliable networks from unreliable gates with almost minimal complexity”, In *Proc. of “Fundamentals of Computation Theory”*, Kazan, 1987, LNCS 278, Springer-Verlag, 1987, pp. 462-469.

[94] D. Uhlig, “On the synthesis of self-correcting schemes from functional elements with a small number of reliable elements,” *Math. Notes. Acad. Sci. USSR* 15, 1974, pp. 558-562.

[95] H. Venkateswaran, “Properties that characterize LOGCFL,” In *Proc. of the 19th STOC*, 1987, pp. 141-150.

[96] H. Venkateswaran, Personal communication.

[97] L. G. Valiant and V.V. Vazirani, "NP is as easy as detecting unique solutions," *Theoretical Computer Science*, 47, 1986, pp. 85-93.

[98] I. Wegener, "The Complexity of Boolean Functions", Wiley-Teubner, 1987.

[99] I. Wegener, "On the complexity of branching porgrams and decision trees for clique functions", *Journal of the ACM*, 35, 1988, pp. 461-471.

[100] A. Wigderson, "The fusion method for lower bounds in circuit complexity", In *Bolyai Society Mathematical Studies, Combinatorics, Paul Erdős is Eighty*, (Volume 1), Keszthely (Hungary), 1993, pp. 453-467.

[101] A. Wigderson, "$NL/poly \subseteq \oplus L/poly$," *Proc. of 9th Conf. Structure in Complexity Theory*, 1994, pp. 59-62.

[102] A. Yao, "Separating the polynomial hierarchy by oracles" In *Proc. of "26th IEEE Symposium on the Foundations of Computer Science"*, 1985, pp. 1-10.

[103] S. Zak, "An exponential lower bound for one time only branching porgrams", In *Proc. of MFCS'84*, Springer Lect. Notes in Comp. Sci. 176, 1984, pp. 562-566.

[104] U. Zwick, "A 4n lower bound on the combinatorial complexity over $U_2 = B_2 \setminus \{\oplus, \equiv\}$ of certain symmetric Boolean functions", *SIAM J. on Comput.*, 20, No. 3, 1991, pp. 499-505.