Electronic Colloquium on Computational Complexity, Report No. 26 (2015)

Negation-Limited Formulas

Siyao Guo* Ilan Komargodski†

Abstract

Understanding the power of negation gates is crucial to bridge the exponential gap between monotone and non-monotone computation. We focus on the model of formulas over the De Morgan basis and consider it in a negation-limited setting.

We prove that every formula that contains t negation gates can be shrunk using a random restriction to a formula of size $O(t)$ with the shrinkage exponent of monotone formulas. As a result, the shrinkage exponent of formulas that contain a constant number of negation gates is equal to the shrinkage exponent of monotone formulas. Moreover, we show that average-case lower bounds for monotone formulas can be extended to get average-case lower bounds for formulas with few negations. Using the average-case lower bound for polynomial-size monotone formulas of Rossman (CCC '15), we obtain an average-case lower bound for polynomial-size formulas with $n^{1/2-o(1)}$ negations, where n is the input size.

Recently, circuits with few negations have drawn much attention in various areas of theoretical computer science. Specifically, Blais et al. (ECCC '14) studied the uniform-distribution learnability of circuits with few negations, and Guo et al. (TCC '15) proved lower bounds on the number of negations required to represent many cryptographic primitives as circuits. Following Guo et al., we study how many negations are required to implement cryptographic primitives using *formulas*, and provide lower bounds for pseudorandom functions, one-way permutations, hardcore predicates and extractors. In particular, we show that every formula that computes a one-way permutation on n inputs must have $\omega(\log n)$ negations, and that any formula that computes a pseudorandom function on n inputs must contain $\Omega(n)$ negations (which is optimal up to a constant factor). Following Blais et al., we show that formulas with t negations can be learned as fast as circuits with $\log t$ negations.

*Department of Computer Science and Engineering, Chinese University of Hong Kong. Email: syguo@cse.cuhk.edu.hk. Supported by RGC GRF grant CUHK 410111. Part of this work done while visiting IDC Herzliya, supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 307952.

†Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel. Email: ilan.komargodski@weizmann.ac.il. Supported in part by a grant from the I-CORE Program of the Planning and Budgeting Committee, the Israel Science Foundation, BSF and the Israeli Ministry of Science and Technology.

ISSN 1433-8092

Contents

1 Introduction

Understanding the complexity of classical computational models for Boolean functions is the holy grail of theoretical computer science. We focus on one of the simplest and most well studied models known as *Boolean formulas* over the De Morgan basis. Such a formula is a Boolean formula over the basis that includes AND, OR and NOT gates, where the former two are of fan in 2. The size of a formula is defined as the number of leaves it contains. A formula is said to be *monotone* if it does not contain any negation gate.

One of the things that makes it so difficult to prove lower bounds on the size of formulas is the presence of negation gates. The best such lower bound known for formulas is almost cubic [And87, Hås98, Tal14], whereas in the setting of *monotone* formulas, exponential lower bounds are known [Raz85, And85, AB87, Tar88, BU99, HR00].[1] Bridging this gap is a major challenge since even a super-polynomial lower bound on the size of formulas (for a function that is constructible deterministically in polynomial-time) would separate P from NC^1.

In 1962 Nechiporuk [Nec62] considered the model of formulas with a limited number of negation gates and proved the following classical result: $\lceil n/2 \rceil$ negation gates are sufficient to compute any Boolean function on n variables by a formula, and moreover, such a transformation can be made efficiently (see [Nec62, Mor09] and [Juk12]). In this paper, we continue this line of research of negation-limited formulas in three different contexts: complexity, cryptography and learning.

1.1 Our Contributions

We start by considering negation-limited formulas as a complexity-theoretic object and prove two structural results. Then, following on Guo et al. [GMOR15], we study the question of how many negation gates are required to implement fundamental cryptographic primitives as formulas. Lastly, following on Blais et al. [BCO+14], we study how efficiently negation-limited formulas can be learned from uniformly distributed examples.

Complexity of Negation-Limited Formulas

Shrinkage under random restrictions. One of the most successful methods for proving lower bounds in several computational models is the method of shrinkage under random restrictions.[2] This method was invented and first used by Subbotovskaya [Sub61] that proved a lower bound of $\Omega(n^{\Gamma})$ on the size of formulas that compute the parity function on n variables, where $\Gamma \geq 1.5$ is referred to as the *shrinkage exponent* of (De Morgan) formulas under random restrictions. Subsequent improvements on the constant Γ led to improved lower bounds on the formula size. Impagliazzo and Nisan [IN93] and Paterson and Zwick [PZ93] proved that $\Gamma \geq 1.55$ and $\Gamma \geq 1.63$, respectively, Håstad [Hås98] proved that $\Gamma \geq 2 - o(1)$ and very recently Tal [Tal14] closed the gap and proved that $\Gamma = 2$. Apart from being useful for proving lower bounds, shrinkage results have a broad scope

[1] More precisely, there exists an explicit Boolean function on n inputs such that every formula that computes it must be of size $n^{3-o(1)}$ (see [Hås98, Tal14]). Moreover, there exists an explicit Boolean function on n inputs such that every monotone formula that computes it must be of size $2^{\Omega((n/\log n)^{1/3})}$ (see [HR00]).

[2] A *random restriction* with parameter $p \in [0,1]$ is a vector $\rho \in \{0,1,\star\}^n$ such that with probability p each entry gets the value $\star$ and with probability $1-p$ each entry is assigned, with equal probabilities, to 0 or 1. Given a function $f : \{0,1\}^n \to \{0,1\}$ and a random restriction ρ as above, the restricted function $f|_\rho$ is defined in the following way: if $\rho_i \in \{0,1\}$ then the i^{th} input variable of f is fixed to 0 or 1, respectively, and otherwise it is still an unfixed variable. We say that formulas have *shrinkage exponent* Γ if for every function f the expected formula size of $f|_\rho$ is at most $O(p^{\Gamma} \cdot L(f) + 1)$, where $L(f)$ is the formula size of f and the expectation is over the choice of ρ.

of applications in other areas including pseudorandomness [IMZ12], Fourier concentration [IK14] and #SAT algorithms [CKK+14, CKS14].

A major open problem (mentioned e.g., in [PZ93, Hås98, Tal14]) is to understand what is the shrinkage exponent of *monotone* formulas.[3] We study the related question of understanding the shrinkage exponent of negation-limited formulas and provide a trade-off between the number of negations and the shrinkage exponent. More precisely, we prove that every formula that contains t negation gates can be shrunk using a random restriction to size $O(t)$ with the shrinkage exponent of *monotone* formulas. As a simple instantiation of our result, we get that the shrinkage exponent of formulas that contain a constant number of negation gates is exactly the same as the shrinkage exponent of monotone formulas. We refer to Section 4.1 for more background, precise definitions and the exact statement of our result.

Average-case lower bounds extension. An average-case computation (a.k.a. approximate computation) of a function $f : \{0,1\}^n \to \{0,1\}$ is a computation that is required to agree with f only on a $1/2 + \delta$ fraction of the inputs. Besides being interesting in their own right, average-case lower bounds (a.k.a. correlation bounds) have proved useful in many fields of complexity theory, such as derandomization (e.g., [Nis91, NW94]).

Recently, Rossman [Ros15] proved the first average-case lower bound for mNC^1, the class of polynomial-size logarithmic-depth monotone circuits, or equivalently, polynomial-size monotone formulas. More precisely, for every $\varepsilon > 0$, Rossman gives an explicit monotone function on n variables which is $(1/2 + n^{-1/2+\varepsilon})$-hard to approximate in mNC^1 under the uniform distribution. Moreover, he showed that any average-case lower bound for monotone circuits, can be extended to get an average-case lower bound for circuits with few negations. That is, he showed that if a monotone function can be approximated by a circuit of a given size and depth and t negations with probability $1/2 + \delta$ (over the uniform distribution), then it can be approximated by a *monotone* circuit of the same size and depth with probability $1/2 + \delta/2^t$. Thus, his bound for mNC^1 extends to circuits in NC^1 with at most $(1/2 - \varepsilon) \log n$ negations.

In the setting of formulas we get exponential improvement in the approximation factor, and hence, in the number of negations needed. Namely, we show that if a monotone function can be approximated by a formula of a given size and depth and t negations with probability $1/2 + \delta$, then it can be approximated by a *monotone* formula of the same size and depth with probability roughly $1/2 + \delta/t$. Combining this with the average-case lower bound for polynomial-size monotone formulas of Rossman [Ros15], we get for any $\varepsilon > 0$, an average-case lower bound for polynomial-size formulas with roughly $n^{1/2-\varepsilon}$ negations. We refer to Section 4.2 for more details.

Cryptography and Learning of Negation-Limited Formulas

One of the goals of cryptography is to study how simple cryptographic primitives can be, where simplicity can be measures by e.g., the required assumptions, the circuit depth and more.

Goldreich and Izsak [GI12] were the first to study whether basic primitives can be monotone. They showed that one-way functions and pseudorandom generators admit an inherent gap. On the one hand, if one-way functions can be computed by a polynomial-size circuit, then there are one-way functions that can be computed by *monotone* polynomial-size circuits. On the other hand, it is *impossible* for a pseudorandom generator to be monotone.

Recently, Guo et al. [GMOR15] extended the results of [GI12] for other cryptographic primitives. In particular, they showed that one-way permutations cannot be monotone (unlike one-way func-

[3] It is conjectured that the shrinkage exponent of monotone formulas is equal to 3.27, the shrinkage exponent of read-once formulas (see Conjecture 3 in [PZ93]).

tions). Moreover, they initiated the study of the *negation complexity* of basic primitives, where they defined the negation complexity of a function f to be the minimal number of negation gates needed to compute the function by a Boolean *circuit*. They proved several interesting results including (to mention a few) that every pseudorandom function requires $\log n - O(1)$ negations and that every hardcore predicate requires $(1/2 - o(1)) \log n$ negations.

The simplicity of a cryptographic primitive can also be measured by the simplicity of the model in which it can be implemented (see e.g., [AIK06]). Since formulas seem to be simpler than general circuits and many concrete cryptographic constructions instantiations are actually described as formulas (e.g., [Hås87, NR04]), we study the complexity and negation complexity of basic cryptographic primitives represented as formulas, ask the analogous questions to the ones asked by [GI12, GMOR15] and prove the following results.

Pseudorandom functions. We show that any formula that computes a pseudorandom function on n bits must contain at least $\Omega(n)$ negation gates (which is tight up to a constant factor [Nec62, Mor09]). This bound is exponentially higher than the bound obtained by Guo et al. [GMOR15] in the setting of circuits. Moreover, we prove a lower bound on the formula size of any pseudorandom function, that is, we show that any pseudorandom function on n bits requires a formula of size at least $\Omega(n^2)$.

One-way functions and permutations. We start with a simple observation that if there are one-way functions in NC^1 (which is a relatively mild assumption, implied by most number-theoretic or algebraic intractability assumptions commonly used in cryptography), then there are one-way functions computable by *monotone* formulas of logarithmic-depth. Then, we show that every permutation on n bits that can be computed by a formula of size s that contains t negations can be inverted (on every image) in time $2^{2t} \cdot s$. This shows, in particular, that every implementation of a one-way permutation using a polynomial-size formula must have at least $\omega(\log n)$ negations. As a comparison, Guo et al. [GMOR15] left open the question whether one-way permutations are computable by circuits that contain just one negation gate.

Hardcore predicates and extractors. We also prove that other basic primitives are highly non-monotone when implemented as formulas. A Boolean function $h : \{0,1\}^n \to \{0,1\}$ is a *hardcore predicate* for a function $f : \{0,1\}^n \to \{0,1\}$ if, given $f(x)$, it is hard to compute $h(x)$. A *strong extractor* is a function that produces almost uniform bits from weak sources of randomness, even when the truly random seed used for extraction is revealed (see Section 2 for the exact definitions). We prove that any hardcore predicate on n bits requires roughly $\Omega(\sqrt{n})$ negation gates. In addition, we show that any extractor of 100 bits on n-bit sources of min-entropy $n^{1/2-\varepsilon}$ requires formulas with at least $\Omega(n^{\varepsilon})$ negation gates. As before, these bounds are exponentially higher than the bounds obtained by Guo et al. [GMOR15] in the setting of circuits.

We refer to Sections 4.3 to 4.5 for the exact details and precise statements.

Uniform-distribution learnability. Monotone functions are known to be somewhat efficiently learnable with high accuracy given uniformly distributed examples. Namely, Bshouty and Tamon [BT96] showed that any monotone Boolean function on n variables can be learned from uniformly distributed examples to error ε in time $O(n^{\sqrt{n}/\varepsilon})$. Recently, Blais et al. [BCO+14] studied the question of learning negation-limited *circuits*. They showed that any function on n variables that can be computed by a circuit with t negations can be learned from uniformly distributed examples to error ε in time $n^{O(2^t \cdot \sqrt{n}/\varepsilon)}$.

We study the question of learning formulas from uniformly distributed examples and show that formulas with t negations can be learned, roughly, as fast as circuits with $\log t$ negations. More precisely, we show that any function f on n variables that can be computed by a formula with t

negations can be learned given labeled examples $(x, f(x))$ (where the x's are chosen uniformly at random) to error ε in time $n^{O(t\cdot\sqrt{n}/\varepsilon)}$. We refer to Theorem 4.18 for the precise statement of our result.

1.2 Related Work

Extensive research has been devoted to the study of negation gates and their power. We start by mentioning the previous efforts that are directly relevant to the results and techniques used in this paper. First, it is the work of Guo et al. [GMOR15] that studied the complexity of some basic cryptographic primitives in the general model of Boolean circuits with a limited number of negations. Second, it is the work of Blais et al. [BCO$^+$14] that studied the efficiency of learning general circuits with a limited number of negation gates. Our results in the context of cryptography and learning theory are influenced by the above papers but require different tools and new ideas (see Sections 4.3 to 4.5).

In addition, we mention the recent result of Rossman [Ros15] that proved an average-case lower bound under product distributions for mNC^1. His average-case lower bound also extends to the negation-limited regime to get an average-case lower bound under the uniform distribution for NC^1 circuits with $(1/2-\varepsilon)\log n$ negation gates (improving upon [AM05]). We use some of his ideas to get an average-case lower bounds for formulas with $n^{1/2-\varepsilon}$ negations in Section 4.2.

Besides the above, the question about the power of negation gates in different models has been studied in many works some of which we mention here. Markov [Mar58] showed that $\lceil \log_2(n+1) \rceil$ negation gates are enough to compute every function by a circuit and Fischer [Fis75] proved that any circuit can be efficiently transformed into a circuit with $\lceil \log_2(n+1) \rceil$ negation gates. Santha and Wilson [SW91] studied negation-limited constant-depth circuits and showed that $\lceil \log_2(n+1) \rceil$ negation gates are not enough to compute some Boolean functions. Raz and Wigderson [RW89] showed that there exists an explicit monotone function that can be computed by polynomial-size, depth $O(\log^2 n)$ circuits, but cannot be computed by any polynomial-size, depth $k \cdot \log n$ circuit that only uses $o(n/2^k)$ negated variables (and is not allowed to have any additional negation gate). Tanaka, Nishino and Beals [TNB96] studied the negation complexity of symmetric function and their techniques were extended [BNT98] to improved the overhead of Fischer's transformation [Fis75]. Sung and Tanaka [ST04] proved a generalization of Markov's theorem for bounded-depth circuits. Iwama, Morizumi and Tarui [IMT09] proved lower bounds on the size of negation-limited inverters and circuits computing the parity function. For more information on negations we refer to Jukna's book [Juk12, §10] and references therein.

1.3 Overview of Our Techniques

In this section we present a high-level overview of the techniques used to obtain some of our results. We begin by explaining the starting point of this work: how to improve upon some of the results of Blais et al. [BCO$^+$14] and Guo et al. [GMOR15] when we restrict the model to formulas (rather than circuits).

Blais et al. and Guo et al. used as their main tool a decomposition theorem for circuits that states that any function f that can be computed using a circuit with t negations can be decomposed using $T+1$ functions $h, g_1, \ldots, g_T$ such that $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(2^t)$, h is either the parity function or its negation and each g_i is a monotone function. Morally, this theorem can be thought of as (an inefficient) transformation that pushes negation gates to the root of the circuit. Using this decomposition, Guo et al. and Blais et al. were able to prove a non-trivial bound on the *total influence* (denoted by $\mathsf{Inf}(\cdot)$) of functions that can be computed with few negations. Namely,

by a simple application of the union bound they showed that if a function $f : \{0,1\}^n \to \{0,1\}$ can be computed using a circuit with t negations, then $\mathsf{Inf}(f) \leq O(2^t \cdot \sqrt{n})$. This bound along with some additional ideas enabled [GMOR15] and [BCO$^+$14] to prove lower bounds for extractors and hardcore predicates, and upper bounds on learning, respectively.

Our starting point is the observation that when the underlying model is formulas, the above decomposition can be made almost the same except for one crucial improvement: any function f that can be computed using a *formula* that contains t negation gates can be decomposed into $T+1$ functions $h, g_1, \ldots, g_T$ as before except that $T = O(t)$. Roughly speaking, the reason for the exponential improvement in T stems from a theorem of Nechiporuk [Nec62] that is tight for formulas (whereas [GMOR15, BCO$^+$14] used a theorem of Markov [Mar58] that is tight for circuits). Using this decomposition theorem one can (as before) get a bound on the total influence of functions $f : \{0,1\}^n \to \{0,1\}$ that can be computed by formulas with t negations of the form $\mathsf{Inf}(f) \leq O(t \cdot \sqrt{n})$. By carefully adjusting the proofs of [GMOR15, BCO$^+$14] one can get exponential improvements in the setting of formulas.[4]

Interestingly, we observe that the above connection between total influence and negation complexity is deeper, more general and does not have to go through the above decompositions. The literature on negation complexity defines a measure, $a(\cdot)$, called "alternation complexity" which denotes the maximal number of times a function $f : \{0,1\}^n \to \{0,1\}$ changes its value along a chain (i.e., a monotone sequence of strings) starting at 0^n and ending at 1^n. We give a direct probabilistic argument for the fact that for any function f it holds that $\mathsf{Inf}(f) \leq O(a(f) \cdot \sqrt{n})$. The bounds used in [GMOR15, BCO$^+$14] follow by using a theorem of Markov [Mar58] that states that $a(f) = O(2^t)$, where t is the number of negations required to compute f using a circuit. The bound for formulas follows by using a theorem of Nechiporuk [Nec62] (see also [Mor09]) that states that $a(f) = O(t)$, where t is the number of negations required to compute f using a formula. Using the latter bound and ideas from [GMOR15, BCO$^+$14], we are able to get lower bounds for extractors and hardcore predicates and an upper bound on learning in the setting of negation-limited formulas. We refer to Theorem 3.2 and Section 4.5 for additional information.

Unfortunately, none of the above techniques seems to be enough to prove our shrinkage theorem and the average-case lower bounds extension theorem for formulas with few negations. First, it is unclear how to use the influence bound to get such structural results. Second, the resulting decomposition is inefficient in the sense that there is no non-trivial bound on the sizes of the resulting g_i's, and implementing h as a formula has a large overhead.[5]

To overcome these issues we prove an *efficient* version of the above decomposition theorem. Specifically, we prove that any function f that can be computed using a formula of size s that contains t negation gates can be decomposed using $T+1$ functions $h, g_1, \ldots, g_T$ such that $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(t)$, h is a read-once formula, each g_i is a monotone function and the total (monotone) formula size of all the g_i's is at most $2s$. In other words, we are able to push all the negation gates to the root of the formula while increasing its size only by a small constant factor (i.e., 2). This decomposition is more involved than the inefficient version and is influenced by ideas and techniques used in recent papers on De Morgan formulas [IMZ12, KRT13, Tal14] (see Section 3

[4] A different approach to get most of the lower bounds of [GMOR15] and the upper bound of [BCO$^+$14] in the formula setting is (roughly speaking) to use the above decomposition theorem for formulas, view h as a circuit on t inputs, apply Fischer's transformation [Fis75] to implement h with $\lceil \log_2(t+1) \rceil$ negations and reduce to the corresponding result on circuits. We choose to give direct proofs since they give more intuition to why the results are correct and sometimes provide stronger statements. For example, in our lower bound on the number of negations needed to implement a pseudorandom function, as in [GMOR15], our proof gives an *efficient statistical test* that can be executed given black-box access to any proposed candidate described as a formula.

[5] The efficiency of the decomposition was not an issue in [GMOR15, BCO$^+$14]. Moreover, shrinkage is a combinatorial property that, unlike e.g., formulas, general circuits do not have.

for further details). Intuitively, to get our shrinkage result we first apply the above decomposition and then apply a random restriction. The analysis relies on two properties of the decomposition: it does not introduce much overhead in the formula size, and the g_i's are monotone and, thus, shrink as well as monotone formulas (see Section 4.1 for further details). To get the average-case lower bounds extension theorem, we plug-in our efficient decomposition theorem into a framework of Rossman [Ros15].

We remark that some of our proofs do not rely on any of the above mentioned techniques but require other ideas. Specifically, in Section 4.3, the proof of the lower bound on the size of any formula that computes a pseudorandom function uses a general bound on the total influence of small formulas [Shi00, Lee09, GKR12]. Moreover, in Section 4.4, the proof of the lower bound on the number of negations required to compute one-way permutations relies on Talagrand's inequality [Tal96] and the structure of formulas. We note again that Guo et al. [GMOR15] left open the question whether one-way permutations are computable by circuits that contain just one negation gate.

1.4 Paper Organization.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of the notation, definitions, and tools underlying our proofs. In Section 3 we present our decomposition theorem for negation-limited formulas as well as the relation between total influence and negation complexity that will be useful to get our main results. Section 4.1 is devoted to the proof of the shrinkage result, Section 4.2 is devoted to the average-case lower bounds extension theorem, Sections 4.3 and 4.4 contain our results for pseudorandom functions, and one-way functions and permutations, respectively, Section 4.5 contains our bounds for hardcore predicates, extractors, and learning. In Section 5 we conclude with some open problems.

2 Preliminaries

In this section we present the notation and basic definitions that are used in this work. For an integer $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, \ldots, n\}$. For a distribution X we denote by $x \leftarrow X$ the process of sampling a value x from the distribution X. Similarly, for a set $\mathcal{X}$ we denote by $x \leftarrow \mathcal{X}$ the process of sampling a value x from the uniform distribution over $\mathcal{X}$. Unless explicitly stated, we assume that the underlying probability distribution in our equations is the uniform distribution over the appropriate set. Further, we let $\mathcal{U}_\ell$ denote the uniform distribution over $\{0,1\}^\ell$. We use $\log x$ to denote a logarithm in base 2. We denote by $\mathsf{wt}(x)$ the Hamming weight of a string $x \in \{0,1\}^n$ (i.e., the number of 1's in the string).

Proposition 2.1 (Chernoff bound)**.** *Let $X = \sum_{i=1}^{n} X_i$ be a sum of identically distributed independent random variables $X_1, \ldots, X_n \in \{0,1\}$. Let $\mu = \mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[X_i]$. It holds that for $\delta \in (0,1)$,*

$$\Pr[X < (1-\delta)\mu] \leq \exp\left(-\delta^2 \mu/2\right)$$

and

$$\Pr[X > (1+\delta)\mu] \leq \exp\left(-\delta^2 \mu/3\right).$$

Boolean Circuits and Formulas

We recall some standard definitions and notation regarding formulas. We refer to [Juk12] for a thorough introduction. We consider formulas over the De Morgan basis $B_{\mathsf{DM}} = \{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$, where the AND and OR gates are of fan-in two. Whenever we refer to formulas we actually refer to De Morgan formulas.

A Boolean formula is a Boolean circuit whose fan-out is at most one. A De Morgan formula is represented by a tree such that every leaf is labeled by an input variable and every internal node is labeled by an operation from B_2. A formula is said to compute a function $f : \{0,1\}^n \to \{0,1\}$ if on input $x \in \{0,1\}^n$ it outputs $f(x)$. The computation is done in the natural way from the leaves to the root. The *size* of a formula F, denoted by $L(F)$, is defined as the number of leaves it contains. For a function f, we denote by $L(f)$ the size of the smallest formula that computes the function f.

A formula is called *read-once* if every input variable labels at most one leaf. A formula F that does not contain negation gates is called a *monotone formula*. We say that a formula F is *anti-monotone* if F is the negation of a monotone formula.

Consider a formula F. Let q be a node in F (q can be either an internal node or a leaf). We refer to the tree rooted at q as a *subformula* of F or a subtree of F.

Let $f : \{0,1\}^n \to \{0,1\}^m$ be a Boolean multi-bit output function. Such a function can be computed by m formulas $F_1, \ldots, F_m$ such that F_i computes the i^{th} output bit of f. The *size* of the formula that computes f is the sum of the sizes of $F_1, \ldots, F_m$. Moreover, the number of negation gates in f is the sum of the number of negation gates in $F_1, \ldots, F_m$.

Decrease, Alternating and Inversion Complexity

For two strings $x, y \in \{0,1\}^n$, we write $x \preceq y$ if $x_i \leq y_i$ for every $i \in [n]$. If $x \preceq y$ and $x \neq y$, then we write $x \prec y$. A *chain* $\mathcal{X} = (x^1, \ldots, x^t)$ is a monotone sequence of strings over $\{0,1\}^n$, i.e., $x^i \preceq x^{i+1}$ for every $i \in [t]$. We say i is a *jump-down position* of f along a chain $\mathcal{X} = (x^1, x^2, \ldots, x^t)$ if $f(x^i) = 1$ and $f(x^{i+1}) = 0$. We let $d(f, \mathcal{X})$ be the number of all jump-down positions of f on chain $\mathcal{X}$ We say a chain $\mathcal{X} = (x^1, x^2, \ldots, x^t)$ is *k-alternating* with respect to a function f if there exist indexes $i_0 < i_1 < \ldots < i_k$ such that $f(x^{i_j}) \neq f(x^{i_j+1})$, for every $j \in [0, k-1]$. We let $a(f, \mathcal{X})$ be the size of the largest set of indexes satisfying this condition. The *decrease* complexity of a Boolean function f is given by $d(f) \stackrel{\text{def}}{=} \max_{\mathcal{X}} d(f, \mathcal{X})$ and the alternating complexity of a Boolean function f is given by $a(f) \stackrel{\text{def}}{=} \max_{\mathcal{X}} a(f, \mathcal{X})$, where $\mathcal{X}$ is a chain over $\{0,1\}^n$. Note that $a(f) \leq 2d(f) + 1$.

For a Boolean function f, we define the *inversion complexity* of f, denoted by $I(f)$, as the minimum number of NOT gates in any formula that computes f. The relation between the inversion complexity and decrease complexity is stated in the following theorem.

Theorem 2.2 ([Nec62, Mor09])**.** *For every Boolean function f it holds that*

$$I(f) = d(f),$$

where $I(f)$ is the inversion complexity of f and $d(f)$ is the decrease of f.

Fourier Basis, Influence and Sensitivity

For each $S \subseteq [n]$, define $\chi_S : \{0,1\}^n \to \{-1,1\}$ as $\chi_S(x) = \prod_{i \in S} (-1)^{x_i}$. It is well known that the set $\{\chi_S\}_{S \subseteq [n]}$ is an orthonormal basis (called the Fourier basis) for the space of all functions $f : \{0,1\}^n \to \mathbb{R}$. It follows that every function $f : \{0,1\}^n \to \mathbb{R}$ can be represented as

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \chi_S(x)$$

where $\widehat{f} : \{0,1\}^n \to \mathbb{R}$, and $\widehat{f}(S) \stackrel{\text{def}}{=} \mathbb{E}_x[(-1)^{\sum_{i\in S} x_i + f(x)}]$ is called the Fourier coefficient of f at $S \subseteq [n]$.

We use $\mathsf{Inf}_i(f)$ to denote the *influence* of the i-th input variable on f, i.e.,

$$\mathsf{Inf}_i(f) \stackrel{\text{def}}{=} \Pr_x[f(x) \neq f(x^{\oplus i})],$$

where $x^{\oplus i}$ denotes the string obtained from x by flipping its i-th coordinate. The *influence* of f (also known as *average-sensitivity*) is defined as $\mathsf{Inf}(f) \stackrel{\text{def}}{=} \sum_{i\in[n]} \mathsf{Inf}_i(f)$.

We let $\mathsf{NS}_p(f)$ denote the *noise sensitivity* of f under noise rate $p \in [0, 1/2]$, which is defined as $\Pr[f(X \oplus Y) \neq f(X)]$, where X is distributed uniformly over $\{0,1\}^n$, and Y is the p-biased binomial distribution over $\{0,1\}^n$, i.e., each coordinate of Y is set to 1 independently with probability p. We refer to O'Donnell's book [O'D14] for an introduction to Fourier analysis.

Some of our proofs rely on the following inequality for monotone Boolean functions.

Proposition 2.3 (Talagrand [Tal96])**.** *For any pair of monotone Boolean functions $f, g\colon \{0,1\}^n \to \{0,1\}$, it holds that*

$$\Pr_x[f(x) = 1 \wedge g(x) = 1] \geq \Pr_x[f(x) = 1] \cdot \Pr_x[g(x) = 1] + \psi\Big(\sum_{i\in[n]} \mathsf{Inf}_i(f) \cdot \mathsf{Inf}_i(g)\Big),$$

where $\psi(x) \stackrel{def}{=} c \cdot x/\log(e/x)$, e is the base of the natural logarithm and $c > 0$ is a fixed constant independent of n.

One-Way Functions, Hardcore Bits and Pseudorandom Functions

We say that a function $f\colon \{0,1\}^n \to \{0,1\}^m$ is an (s,ε)-secure *one-way function* (OWF) if for every circuit C of size at most s,

$$\Pr_{x\leftarrow\{0,1\}^n,\, y=f(x)}[C(y) \in f^{-1}(y)] \leq \varepsilon.$$

If $m = n$, we say that f is *length-preserving*. If f is an (s,ε)-secure one-way function that is lengh-preserving and one-to-one, we say that f is an (s,ε)-secure *one-way permutation* (OWP).

We say that a function $h\colon \{0,1\}^n \to \{0,1\}$ is an (s,ε)-secure hardcore bit for a function $f\colon \{0,1\}^n \to \{0,1\}^m$ if, for every circuit C of size s,

$$\left| \Pr_{x\leftarrow\{0,1\}^n}[C(f(x)) = h(x)] - \frac{1}{2} \right| \leq \varepsilon.$$

Let Π_n be the set of all Boolean functions on n variables, and $f\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$. We say that f is an (s,ε)-secure *pseudorandom function* (PRF) if, for every (non-uniform) algorithm $\mathcal{A}$ that can be implemented by a circuit of size at most s,

$$\left| \Pr_{K\leftarrow\{0,1\}^\lambda}[D^{f(K,\cdot)}(1^\lambda) = 1] - \Pr_{f\leftarrow\Pi_n}[D^{f(\cdot)}(1^\lambda) = 1] \right| \leq \varepsilon.$$

where $\mathcal{A}^h$ denotes the execution of $\mathcal{A}$ with oracle access to a Boolean function $h\colon \{0,1\}^n \to \{0,1\}$ (circuits with access to oracle gates are defined in the natural way).

Extractors

The *min-entropy* of a random variable X, denoted by $H_\infty(X)$, is the largest real number k such that $\Pr[X = x] \leq 2^{-k}$ for every x in the range of X. A distribution X over $\{0,1\}^n$ with $H_\infty(X) \geq k$ is said to be an (n,k)-source. Given random variables X and Y with range $\{0,1\}^m$, we let

$$\delta(X,Y) \stackrel{\text{def}}{=} \max_{S \subseteq \{0,1\}^m} \left| \Pr[X \in S] - \Pr[Y \in S] \right|$$

denote their *statistical distance*. We say that X and Y are ε-close if $\delta(X,Y) \leq \varepsilon$.

We say that a function $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ is a (strong) (k,ε)-*extractor* if, for any (n,k)-source X, the distributions $\mathcal{U}_{s+m}$ and $(\mathcal{U}_s, \mathsf{Ext}(X,\mathcal{U}_s))$ are ε-close.[6]

3 Tools for Negation-Limited Formulas

In this section we present two basic tools for analyzing negation-limited formulas. First, in Theorem 3.1 we prove an efficient decomposition theorem for formulas which, intuitively, pushes all negation gates to the root of the formula. Second, in Theorem 3.2 we prove a relation between the total influence of a function and its alternation complexity.

Theorem 3.1. *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function computed by a formula F of size s containing $t > 0$ negation gates. Then, there exist $T \leq 15(t+1)$ functions $g_1, \ldots, g_T\colon \{0,1\}^n \to \{0,1\}$ and a function $h : \{0,1\}^T \to \{0,1\}$ such that $f(x) = h(g_1(x), \ldots, g_T(x))$, h is computable by a read-once* formula *and $g_1, \ldots g_T$ are computable by monotone formulas of total size at most $2s$.*

Theorem 3.2. *For any function $f : \{0,1\}^n \to \{0,1\}$, it holds that*

$$\mathsf{Inf}(f) \leq O\left(a(f) \cdot \sqrt{n}\right),$$

where Inf is the total influence of f and $a(f)$ is the alternating complexity of f.

Using Nechiporuk's theorem (see Theorem 2.2) we get the following corollary.

Corollary 3.3. *For any function $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a formula with t negations, it holds that*

$$\mathsf{Inf}(f) \leq O\left(t \cdot \sqrt{n}\right).$$

We begin with the proof of Theorem 3.1. We first need the following claim that states that any formula that has t negation gates can be decomposed into $2(t+1)$ subformulas such that each of them is monotone or anti-monotone (i.e., either it has zero negations or it has one negation in the root). Moreover, each such subformula has at most two "special" children which are subformulas by themselves. We note that the proof of Theorem 3.1 draws ideas from a proof of a different decomposition theorem used by Tal [Tal14] which, in turn, is partially built on ideas that were used before in [IMZ12] and then in [KRT13]. However, since the properties of our decomposition are very different, we cannot use the other theorems as a black box.

Claim 3.4. *Any formula F of size s that contains $t > 0$ negations can be decomposed into at most $2(t+1)$ subformulas of total size s, such that each has at most one negation gate in its root. Moreover, each such subformula has at most two "special" children which are other subformulas.*

[6] Two occurrences of the same random variable in an expression refer to the same instance of the variable.

Proof. Execute the following step t times: let $g_1, \ldots, g_s$ be the nodes of the formula F sorted by their distance from the root g_s. For any $i = 1, \ldots s$ if $g_i = \mathsf{NOT}$ we set F_i to be the subformula rooted at g_i and set $F = F \setminus F_i$. This process results with $T = t + 1$ subformulas $F_1, \ldots, F_T$ whose total size is s and each is either monotone or rooted by a NOT gate.

For each subformula F_i with more than two subtree children, find a subtree F_i' of F_i with exactly two subtree children, and divide F_i into F_i' and $F_i \setminus F_i'$. Note that $F_i \setminus F_i'$ now has one fewer subtree children. Continue doing this until all subtrees have at most two subtree children. This process results with the desired number of subtrees, $2(t+1)$, since the above process can continue at most the original number of subtrees. ∎

Proof of Theorem 3.1. Let F be as in the lemma. Apply the decomposition from Claim 3.4 on F to get the subtrees $F_1, \ldots, F_{T'}$, where $T' = 2(t+1)$. We show by induction on T' that one can construct a read-once formula H of size $T \leq 7T'$ and T monotone formulas $G_1, \ldots, G_T$ of size s such that $F(x) = H(G_1(x), \ldots, G_T(x))$. For $t = 0$ (and $T' = 1$) the statement holds trivially.

Assume that the root of the formula F is a node in the subtree F_1, and that the subtree F_1 has two subformula children F_2 and F_3. (The case in which F_1 has only one subformula child is handled similarly). Denote by $k_2^{(1)}, k_3^{(1)} \in F_1$, $k_1^{(2)} \in F_2$ and $k_1^{(3)} \in F_3$ the nodes such that $k_2^{(1)}$ is the father of $k_1^{(2)}$ and $k_3^{(1)}$ is the father of $k_1^{(3)}$. Disconnect F_2 and F_3 from F_1 and add two new leaves labeled by z_2 and z_3 to F_1 as a child of $k_2^{(1)}$ and $k_3^{(1)}$, respectively.

Call the formula F_1 with the two new leaves F'. Notice that by Claim 3.4, F' is either monotone or anti-monotone, namely a negation of a monotone function. We prove the case when F' is anti-monotone and the argument for monotone case is similar. Let F_1' be the minimal subformula of F' that contain both z_2 and z_3 and let F_2' and F_3' be the corresponding subtrees such that $F_1' = F_2' \ \mathsf{gate} \ F_3'$, where $\mathsf{gate} \in \{\mathsf{AND}, \mathsf{OR}\}$, and F_2' contains z_2 (but not z_3) and F_3' contains z_3 (but not z_2). We will construct a formula which is equivalent to F_1'.

We observe that $F_2' = F_2'|_{z_2=0} \ \mathsf{OR} \ (F_2'|_{z_2=1} \ \mathsf{AND} \ z_2)$. This is true since F_2' is monotone (i.e., does not contain any negation gates). Similarly, $F_3' = F_3'|_{z_3=0} \ \mathsf{OR} \ (F_3'|_{z_3=1} \ \mathsf{AND} \ z_3)$. Thus,

$$F_1' = (F_2'|_{z_2=0} \ \mathsf{OR} \ (F_2'|_{z_2=1} \ \mathsf{AND} \ z_2)) \ \mathsf{gate} \ (F_3'|_{z_3=0} \ \mathsf{OR} \ (F_3'|_{z_3=1} \ \mathsf{AND} \ z_3)).$$

Replacing F_1' with a new leaf z (where z is a new special variable) we have (by a similar argument) that $F_1 = F_1|_{z=1} \ \mathsf{OR} \ (F_1|_{z=0} \ \mathsf{AND} \ z)$ (this follows by the anti-monotonicity of F_1). By expanding according to the definition of z we get that

$$\begin{aligned} F_1 = F_1|_{z=1} \ \mathsf{OR} \ (F_1|_{z=0} \ \mathsf{AND} \ ((F_2'|_{z_2=0} \ \mathsf{OR} \ (F_2'|_{z_2=1} \ \mathsf{AND} \ z_2)) \ \mathsf{gate} \\ (F_3'|_{z_3=0} \ \mathsf{OR} \ (F_3'|_{z_3=1} \ \mathsf{AND} \ z_3)))). \end{aligned}$$

Now, we observe that the right hand side can be rewritten as $F''(G_1, ;G_6, z_2, z_3)$, where F'' is read-once and $G_1, ;G_6$ are formulas of size at most s (defined over the same set of variables as the initial F).

Let t_2 and t_3 be the number of subformulas which are descendants of F_2 and F_3 in the formula decomposition, respectively. By induction the subformula of F rooted at $k_1^{(2)}$ is equivalent to $F_2'(G_1^{(2)}(x), \ldots, G_{6t_2}^{(2)}(x))$, where F_2' is read-once and $G_i^{(2)}$ is of size at most s. Similarly, the subformula of F rooted at $k_1^{(3)}$ is equivalent to $F_3'(G_1^{(3)}(x), \ldots, G_{6t_3}^{(3)}(x))$, where F_3' is read-once and $G_i^{(3)}$ is of size at most s. Thus,

$$F(x) = F''(G_1(x), \ldots, G_6(x), F_2'(G_1^{(2)}(x), \ldots, G_{6t_2}^{(2)}(x)), G_1^{(3)}(x), \ldots, G_{6t_3}^{(3)}(x))).$$

By rearranging the right hand size we get a read-once formula of size $T \leq 6 + 6t_2 + 6t_3 \leq 7T'$ and T monotone subformulas each of size at most s such that their composition is equivalent to F. To see that the total size of the subformulas is bounded by $2s$ notice that every subformula was duplicated at most once. ∎

We proceed with the proof of Theorem 3.2.

Proof of Theorem 3.2. Denote by D the set of all pairs of points in $\{0,1\}^n$ that differ at one coordinate. Namely, $(x, y) \in D$ if and only if there exists an $i \in [n]$ such that $x^{\oplus i} = y$.

We define two ways to sample edges from D and show that they define the same distribution. The first way to sample an edge from D is by first sampling a point $x \in \{0,1\}^n$ and then sampling a random direction $i \in [n]$. This gives rise to the edge $(x, x^{\oplus i})$. Notice that for any edge $e \in D$ it holds that

$$\Pr_{x \leftarrow \{0,1\}^n, i \leftarrow [n]}[(x, x^{\oplus i}) = e] = \frac{1}{n \cdot 2^{n-1}}.$$

Moreover, observe that by the definition of total influence, we have that

$$\frac{\mathsf{Inf}(f)}{n} = \Pr_{x \leftarrow \{0,1\}^n, i \leftarrow [n]}[f(x) \neq f(x^{\oplus i})]. \tag{3.1}$$

The second way is defined as follows. Denote by $\mathcal{C}$ the set of all valid chains starting from 0^n and ending at 1^n. First, we sample a random chain $\mathcal{X} = (x_0 = 0^n, x_1, \ldots, x_{n-1}, x_n = 1^n)$ from $\mathcal{C}$. Notice that $\mathsf{wt}(x_i) = i$ for all $i \in [n] \cup \{0\}$. Then, we pick the edge $e^{(i)} = (x_i, x_{i+1})$ for $i \in [n-1] \cup \{0\}$ on the chain with probability $\binom{n-1}{i}/2^{n-1}$. Notice that this is a probability distribution since we have that $\sum_{i=0}^{n-1} \binom{n-1}{i}/2^{n-1} = 1$. Also, observe that a random chain $\mathcal{X}$ from $\mathcal{C}$ contains an arbitrary edge $(x, x') \in D$ with probability $1/(\binom{n-1}{\mathsf{wt}(x)} \cdot n)$. In total, using the above process, the probability to pick an edge $e \in D$ is

$$\begin{aligned}\Pr_{\mathcal{X} \leftarrow \mathcal{C}, (x_i, x_{i+1}) \leftarrow \mathcal{X}}[(x_i, x_{i+1}) = e] &= \Pr_{(x_i, x_{i+1}) \leftarrow \mathcal{X}}[(x_i, x_{i+1}) = e \mid e \in \mathcal{X}] \cdot \Pr_{\mathcal{X} \leftarrow \mathcal{C}}[e \in \mathcal{X}] \\ &= \frac{\binom{n-1}{\mathsf{wt}(x)}}{2^{n-1}} \cdot \frac{1}{\binom{n-1}{\mathsf{wt}(x)} \cdot n} = \frac{1}{n \cdot 2^{n-1}}.\end{aligned}$$

Therefore, we got that the two ways to sample an edge on the cube have the same distribution. Thus, using Equation (3.1), we get that

$$\frac{\mathsf{Inf}(f)}{n} = \Pr_{\mathcal{X} \leftarrow \mathcal{C}, (x_i, x_{i+1}) \leftarrow \mathcal{X}}[f(x_i) \neq f(x_{i+1})]. \tag{3.2}$$

However, notice that for any $\mathcal{X} \in \mathcal{C}$ it holds that

$$\begin{aligned}\Pr_{(x_i, x_{i+1}) \leftarrow \mathcal{X}}[f(x_i) \neq f(x_{i+1})] &\leq a(f) \cdot \max_{i \in [n-1] \cup \{0\}} \left\{ \frac{\binom{n-1}{i}}{2^{n-1}} \right\} \\ &\leq O\left(a(f)/\sqrt{n}\right),\end{aligned}$$

where the first inequality follows by the definition of $a(f)$ (the maximum number of alternations at any chain) and the second inequality holds since the second term is maximized roughly when $i \approx n/2$ and it is known (by e.g., Stirling's approximation) that $\binom{n}{n/2} = O(2^n/\sqrt{n})$. Plugging this back into Equation (3.2) we get that

$$\mathsf{Inf}(f) \leq O(a(f) \cdot \sqrt{n}).$$

∎

We note that the bound in Theorem 3.2 is tight up to constants. Indeed, for any $n \in \mathbb{N}$, any constant $c \in \mathbb{N}$ (independent of n) and any $t \leq c \cdot \sqrt{n}$ consider the function $f : \{0,1\}^n \to \{0,1\}$ defined as

$$f(x) = \begin{cases} \mathsf{wt}(x) \bmod 2 & \text{if } |\mathsf{wt}(x) - n/2| \leq t/2, \\ 0 & \text{otherwise.} \end{cases}$$

First, it is easy to see that $t - 1 \leq a(f) \leq t + 1$. Moreover, a simple analysis shows that $\mathsf{Inf}(f) \geq \Omega(t \cdot \sqrt{n})$. To see this observe that since $t \leq O(\sqrt{n})$, then $\Pr_{x \leftarrow \{0,1\}^n}[|\mathsf{wt}(x) - n/2| \leq t/2] \geq \Omega(t/\sqrt{n})$, and that if x satisfies that $|\mathsf{wt}(x) - n/2| \leq t/2$, then changing each of its n coordinates will flip the value of the function.

4 The Complexity of Negation-Limited Formulas

4.1 Shrinkage under Random Restrictions

A well known property of formulas is called *shrinkage*. We begin with several definitions.

Definition 4.1 (Restriction)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. A restriction ρ is a vector of length n of elements from $\{0, 1, \star\}$. We denote by $f|_\rho$ the function f restricted according to ρ in the following sense: if $\rho_i = \star$ then the i-th input bit of f is unassigned and otherwise the i-th input bit of f is assigned to be ρ_i.

Definition 4.2 (p-Random restriction)**.** A p-random restriction is a restriction as in Definition 4.1 that is sampled as follows. For every $i \in [n]$, independently with probability p set $\rho_i = \star$ and with probability $\frac{1-p}{2}$ set ρ_i to be 0 and 1, respectively. We denote this distribution of restrictions by $\mathcal{R}_p$.

Definition 4.3 (Shrinkage exponent)**.** Let $\mathcal{F}$ be a class of functions associated with a size function $\mathsf{size} \colon \mathcal{F} \to \mathbb{N}$. The shrinkage exponent of $\mathcal{F}$ is said to be Γ if for any $F \in \mathcal{F}$

$$\mathop{\mathbb{E}}_{\rho \leftarrow \mathcal{R}_p} [\mathsf{size}(F|_\rho)] \leq O\left(p^\Gamma \cdot \mathsf{size}(F) + 1\right).$$

Denote by $\Gamma, \Gamma_0, \Gamma^*$ the shrinkage exponent of (De Morgan) formulas, monotone formulas and read-once formulas, respectively. Denote by Γ_t the shrinkage exponent of formulas that contain at most t negation gates.

Fact 4.4. *The following facts are known:*

1. $\Gamma = 2$ *[Hås98, Tal14].*
2. $\Gamma^* = \log_{\sqrt{5}-1} 2 \approx 3.27$ *[DZ94, HRY95].*
3. *For every* $t \geq 0$ *it holds that* $\Gamma^* \geq \Gamma_t \geq \Gamma = 2$.

Figuring out the value of Γ_0, the shrinkage exponent of *monotone* Boolean formulas, is a major open problem [PZ93, Hås98, Tal14].

Our main theorem of this section is a trade-off between the number of negations in the formula and its shrinkage exponent. In particular, we get that the shrinkage exponent of formulas that contain a constant number of negation gates is equal to Γ_0.

Theorem 4.5. *Let F be a formula that contains $t > 0$ negation gates. It holds that*

$$\mathop{\mathbb{E}}_{\rho \leftarrow \mathcal{R}_p} [L(F|_\rho)] \leq O(p^{\Gamma_0} \cdot L(F) + t).$$

Proof. Given a formula F we decompose it using Theorem 3.1 to get $H, G_1, \ldots, G_T$, where $T \leq 15(t+1)$, $\sum_{i=1}^{T} L(G_i) \leq 2 \cdot L(F)$ and $F(x) = H(G_1(x), \ldots, G_T(x))$. Clearly we have that the formula size of F is at most the sum of the sizes of the G_i's. Namely,

$$L(F) \leq \sum_{i=1}^{T} L(G_i) \leq 2 \cdot L(F), \tag{4.1}$$

where the second inequality is true by the guarantee of the decomposition from Theorem 3.1. Let $\rho \leftarrow \mathcal{R}_p$ be a random restriction. For each $i \in [T]$ since G_i is monotone, we have that $\mathbb{E}_\rho[L(G_i|_\rho)] \leq O(p^{\Gamma_0} \cdot L(G_i) + 1)$. Thus, the expected size of $L(F)$ after applying ρ is upper bounded as follows:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\rho}[L(F|_\rho)] &\leq \sum_{i=1}^{T} \mathop{\mathbb{E}}_{\rho}[L(G_i|_\rho)] && \text{(Linearity of expectation)} \\
&\leq \sum_{i=1}^{T} O(p^{\Gamma_0} \cdot L(G_i) + 1) && \text{(Each } G_i \text{ is monotone)} \\
&\leq O(p^{\Gamma_0} \cdot L(F) + t). && \text{(Equation (4.1))}
\end{aligned}$$

■

Notice that when $t = O(1)$ we get that $\mathbb{E}_\rho[L(F)|_\rho] \leq O(p^{\Gamma_0} \cdot L(F) + 1)$ which means that the shrinkage exponent of such formulas is exactly equal to the shrinkage exponent of monotone formulas. More generally, Theorem 4.5 implies that every formula F that contains $t > 0$ negation gates can be shrunk in two steps of random restrictions such that in the first step the formula F shrinks to size $O(t)$ as monotone formulas shrink (i.e., with Γ_0 as the shrinkage exponent) and in the second step the formula (of size $O(t)$) shrinks as formulas shrink (with Γ as the shrinkage exponent). To be more precise, F can be restricted with a random restriction $\rho_1 \leftarrow \mathcal{R}_{p_1}$, where $p_1 = \sqrt[\Gamma_0]{t/L(F)}$, to get a formula F_1 of size $O(t)$ and then it can be restricted with a random restriction $\rho_2 \leftarrow \mathcal{R}_p$ for any p to get a formula F_2 of size $O(p^\Gamma \cdot t + 1)$.

4.2 Average-Case Lower Bounds Extension

In this section we study average-case lower bounds for negation-limited formulas. We adapt a proof of Rossman [Ros15] to the setting of formulas and prove that if one can approximate a function using a formula with few negations, then it is also possible to approximate it quite well using a monotone formula. Such a statement is useful to extend average-case lower bounds for monotone formulas to average-case lower bounds for formulas with few negation gates.

Theorem 4.6. *Let μ be any product distribution over $\{0,1\}^n$.*[7] *Let $m\colon \{0,1\}^n \to \{0,1\}$ be a monotone function which is balanced under the uniform distribution, namely $m^{-1}(0) = m^{-1}(1) = 2^{n-1}$. Suppose that there exists a function $f\colon \{0,1\}^n \to \{0,1\}$ computable by a formula of size s, depth d and $t > 0$ negations such that*

$$\Pr_{x \leftarrow \mu}[f(x) = m(x)] \geq \frac{1}{2} + \delta.$$

[7] The statement can be generalized to hold for any distribution μ which satisfies that $\mu(x)\mu(y) \leq \mu(x \wedge y)\mu(x \vee y)$ for any $x, y \in \{0,1\}^n$. This condition is known as the FKG condition and is clearly satisfied by product distributions. See [Ros15] for more details.

Then, there exists a function g computably by a monotone formulas of size s and depth d such that

$$\Pr_{x\leftarrow\mu}[g(x)=m(x)]\geq\frac{1}{2}+\Omega\left(\frac{\delta}{t}\right).$$

Proof. For a function $h\colon\{0,1\}^n\to\{0,1\}$, let $\mathsf{mpairs}(h)=\{(x,y)\in h^{-1}(0)\times h^{-1}(1)\mid x\prec y\}$, i.e., the set of all the pairs $x,y\in\{0,1\}^n$ such that $x\prec y$ and $h(x)<h(y)$.

Rossman [Ros15] proved the following claim which states that there exists a probability distribution ν (defined according to m) supported on $\mathsf{mpairs}(m)$ such that the correlation between any (not necessarily monotone) function h and m is related with the probability mass of v over the monotone pairs of h.

Claim 4.7 ([Ros15, §F])**.** *There exists a probability distribution ν (defined according to m) supported on $\mathsf{mpairs}(m)$ such that for any function $h\colon\{0,1\}^n\to\{0,1\}$*

$$\nu(\mathsf{mpairs(h)})\geq 2\cdot\Pr_{x\leftarrow\mu}[h(x)=m(x)]-1,$$

where $\nu(\mathsf{mpairs(h)})\stackrel{\text{def}}{=}\Pr_{(x,y)\leftarrow\nu}[(x,y)\in\mathsf{mpairs(h)}]$. Moreover, equality holds when h is a monotone function.

Using the assumption of the theorem, it follows that $\nu(\mathsf{mpairs(f)})\geq 2\cdot\Pr_{x\leftarrow\mu}[f(x)=m(x)]-1\geq\delta$. Next, we show that there exists a monotone formula g of size at most s and depth d such that $\nu(\mathsf{mpairs(g)})\geq\nu(\mathsf{mpairs(f)})/T$, where $T=O(t)$. This implies (using Claim 4.7 and the fact that g is monotone) the desired conclusion, namely,

$$\Pr_{x\leftarrow\mu}[g(x)=m(x)]=\frac{1}{2}\cdot(1+\nu(\mathsf{mpairs(g)}))\geq\frac{1}{2}+\Omega\left(\frac{\delta}{t}\right).$$

To prove the existence of g we use the following claim.

Claim 4.8. *There exist $T=O(t)$ monotone formulas $g_1,\ldots,g_T$ of size at most s and depth at most d such that $\mathsf{mpairs}(h)\subseteq\bigcup_{i\in[T]}\mathsf{mpairs}(g_i)$.*

Proof. By Theorem 3.1, we can write $f(x)=h'(g_1(x),\ldots,g_T(x))$, where $T\leq 15(t+1)=O(t)$, g_i are monotone formulas each of size at most s and depth d.[8] For any $(x,y)\in\mathsf{mpairs}(h)$, since $f(x)\neq f(y)$, there must exist an $i\in[T]$ such that $g_i(x)\neq g_i(y)$. Then, since g_i is monotone, it must be that $g_i(x)=0$ and $g_i(y)=1$, and hence, $(x,y)\in\mathsf{mpairs}(g_i)$. Therefore, $\mathsf{mpairs}(h)\subseteq\bigcup_{i\in[T]}\mathsf{mpairs}(g_i)$. ■

Claim 4.8 implies that $\nu(\mathsf{mpairs}(h))\leq\sum_{i=1}^{T}\nu(\mathsf{mpairs}(g_i))$. Therefore, by averaging, there exists an $i\in[T]$ such that $\nu(\mathsf{mpairs}(g_i))\geq\nu(\mathsf{mpairs}(h))/T$, which completes the proof. ■

We recall the average-case lower bound of Rossman [Ros15].

Theorem 4.9 ([Ros15, Corollary 1.2])**.** *For every $\varepsilon>0$, there is an explicit monotone function $f:\{0,1\}^n\to\{0,1\}$ such that for every polynomial-size monotone formula F^+ it holds that $\Pr_{x\leftarrow\{0,1\}^n}[F^+(x)=f(x)]\leq 1/2+n^{-1/2+\varepsilon}$.*

Using Theorem 4.6 and Theorem 4.9 we obtain the following corollary.

Corollary 4.10. *For every $\varepsilon>\omega(\log^{-1}n)$, there is an explicit function $f:\{0,1\}^n\to\{0,1\}$ such that for every polynomial-size formula F with $n^{1/2-\varepsilon}$ negations it holds that $\Pr_{x\leftarrow\{0,1\}^n}[F(x)=f(x)]\leq 1/2+o(1)$.*

[8] The original statement of Theorem 3.1 only bounds the total size of $g_1,\ldots,g_T$, however it is easy to see that the same decomposition results with formulas $g_1,\ldots,g_T$ each of size at most s and depth at most d.

4.3 Pseudorandom Functions

In this section we study the negation-limited complexity of pseudorandom functions. In Theorem 4.11 we prove that any formula that computes a pseudorandom function on n input bits (where the seed is not counted as part of the input) must contain at least $\Omega(n)$ negation gates. The proof of the theorem follows the proof of [GMOR15] that was proved in the setting of general Boolean circuits but using the theorem of Nechiporuk (see Theorem 2.2) which gives better results for formulas.

Then, in Theorem 4.13 we prove that any formula that computes a pseudorandom function must be of size at least $\Omega(n^2)$. We prove this using the fact that small formulas have low total influence, and giving an adversary that distinguishes any function f with somewhat low influence from a random function.

Theorem 4.11. *If $f\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$ is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function, then any Boolean formula computing f contains at least $\Omega(n)$ negation gates.*

Proof. Let Π_n be the set of all Boolean functions on n input bits. By the definition of a secure pseudorandom function we have that for any probabilistic polynomial-time algorithm D it holds that

$$\left| \Pr_{K \leftarrow \{0,1\}^\lambda}[D^{f(K,\cdot)}(1^\lambda) = 1] - \Pr_{f \leftarrow \Pi_n}[D^{f(\cdot)}(1^\lambda) = 1] \right| \leq \frac{1}{3}. \tag{4.2}$$

Let $e_i \stackrel{\text{def}}{=} 1^i 0^{n-i}$ and let $\mathcal{X} = (e_0, \ldots, e_n)$ be a chain over $\{0,1\}^n$. We define the algorithm D that has oracle access to a function h as follows: D starts by querying f at each of $e_0, \ldots, e_n$, computes $d(f, \mathcal{X})$ and outputs 1 if and only if $d(f, \mathcal{X}) \geq n/8$. It is easy to see that D can be implemented in polynomial time.

If f is a completely random function, we have that $\mathbb{E}_{f \leftarrow \Pi_n}[d(f, \mathcal{X})] = n/4$ and by Chernoff's bound (see Proposition 2.1) we have that

$$\Pr_{f \leftarrow \Pi_n}[|d(f, \mathcal{X}) - n/4| < n/8] \geq 1 - \exp(-n).$$

Thus, using Equation (4.3) we have that

$$\Pr_{K \leftarrow \{0,1\}^m}[D^{f(K,\cdot)}(1^\lambda) = 1] \geq \frac{2}{3} - o(1).$$

Therefore, there must exist a seed $K^* \in \{0,1\}^\lambda$ for which $d(f_K, \mathcal{X}) \geq n/4$, where $f_K \stackrel{\text{def}}{=} f(K, \cdot)$. Using Theorem 2.2 we have that if F is a formula with t negations that computes f_K, then

$$n/4 \leq d(f_K, \mathcal{X}) \leq d(f_K) = t.$$

Finally, it is easy to conclude that any formula for f also requires at least $\Omega(n)$ negation gates. ■

In the following theorem we show that pseudorandom functions must be computed by formula of size at least $\Omega(n^2)$ (even non-monotone ones). Our proof relies on the following theorem which says that the total influence of a function f that can be computed by a formula of size s is at most $O(\sqrt{s})$. This theorem follows from [Shi00, Lee09] who used a quantum approach or from [GKR12] who gave an alternative classical proof.

Theorem 4.12 ([Shi00, Lee09],[GKR12])**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then*

$$\mathsf{Inf}(f) \leq O(\sqrt{L(f)}).$$

Theorem 4.13. *If $f\colon \{0,1\}^\lambda \times \{0,1\}^n \to \{0,1\}$ is a $(\mathsf{poly}(n), 1/3)$-secure pseudorandom function, then any Boolean formula that compute f must be of size at least $\Omega(n^2)$.*

Proof. Let Π_n be the set of all functions that map strings of length n to bits. By the definition of a secure pseudorandom function we have that for any probabilistic polynomial-time algorithm D it holds that

$$\left| \Pr_{K\leftarrow\{0,1\}^\lambda}[D^{f(K,\cdot)}(1^\lambda) = 1] - \Pr_{f\leftarrow\Pi_n}[D^{f(\cdot)}(1^\lambda) = 1] \right| \leq \frac{1}{3}. \tag{4.3}$$

We define the algorithm D that has oracle access to a function h as follows: D picks a random point x and computes $s(f,x)$ which we define as the number of inputs $x^{\oplus i}$ such that $f(x) \neq f(x^{\oplus i})$, and outputs 1 if and only if $s(f,x) \geq n/4$. It is easy to see that D can be implemented in polynomial time.

If f is a completely random function, we have that $\mathbb{E}_{f\leftarrow\Pi_n, x\leftarrow\{0,1\}^n}[s(f,x)] = n/2$ and by Chernoff's bound (see Proposition 2.1) we have that

$$\Pr_{f\leftarrow\Pi_n, x\leftarrow\{0,1\}^n}[|s(f,x) - n/2| < n/4] \geq 1 - \exp(-n).$$

Thus, using Equation (4.3) we have that

$$\Pr_{K\leftarrow\{0,1\}^m}[D^{f(K,\cdot)}(1^\lambda) = 1] \geq \frac{2}{3} - o(1).$$

Therefore, there must exist a seed $K^* \in \{0,1\}^\lambda$ for which $\Pr_{x\leftarrow\{0,1\}^n}[s(f_K,x) \geq n/4] \geq 2/3 - o(1)$, where $f_K \stackrel{\text{def}}{=} f(K,\cdot)$. Then, by the definition of total influence and using Markov's inequality, we have that

$$\mathsf{Inf}(f_K) = \mathop{\mathbb{E}}_{x\leftarrow\{0,1\}^n}[s(f_K,x)] \geq \Pr_{x\leftarrow\{0,1\}^n}\left[s(f_K,x) \geq \frac{n}{4}\right] \cdot \frac{n}{4} = \Omega(n).$$

Using Theorem 4.12 we have that if f can be computed by a formula of size s, then

$$\mathsf{Inf}(f) \leq O(\sqrt{s}).$$

Thus, we conclude that $L(f) \geq \Omega(n^2)$. ∎

We remark that a somewhat different approach to prove Theorem 4.13 is to use Kraphchenko's bound [Khr71]. In addition, we note that one can combine the natural-proofs technique of Razborov and Rudich [RR97] and known lower bounds for formulas [Hås98, Tal14] to rule out the existence of super strong (say $(2^{n^{1-O(1)}}, 2^{-n^{1-O(1)}})$-secure) PRFs computable by formulas of size $O(n^{3-o(1)})$. We note that the constant $1/3$ in both statements (Theorems 4.11 and 4.13) is arbitrary and can be replaced with any other constant in $[0,1)$.

4.4 One-Way Functions and Permutations

In this section we study the negation-limited complexity of one-way functions and one-way permutations in the model of Boolean formulas.

We start with a simple observation (see Observation 4.14) that if one-way functions in NC^1 exist, then there exist one-way functions that can be computed by monotone logarithmic-depth formulas. Then, we consider the negation-limited complexity of permutations. Guo et al. [GMOR15] showed that one-way permutations cannot be monotone but they left open the question whether one-way permutations are computable by circuits that contain just one negation gate. In Theorem 4.15, we show that any one-way permutation is not computable by a formula that has $O(\log n)$ negations.

Observation 4.14. *Assume that there is a one-way function in* NC^1*. Then, there is a one-way function computable by a logarithmic-depth monotone formula.*

Proof. Recall the transformation of Goldreich and Izsak [GI12] that transformed every one-way function into a monotone one-way function. Let C be a circuit that computes a one-way function and let C' be a circuit obtained from C by pushing all negation gates to the leaves and replacing negated variables by auxiliary variables, namely, $C(x) = C'(x, \bar{x})$, where $\bar{x}_i = \neg x_i$. Let $\mathsf{Th}_k : \{0,1\}^n \to \{0,1\}$ be a function such that $\mathsf{Th}_k(x) = 1$ if and only if the hamming weight of x is at least k. Notice that for any x of hamming weight k it holds that $\neg x_i = \mathsf{Th}_k(x \wedge 1^{i-1}01^{n-i})$. Therefore, $N(x) = (\mathsf{Th}_k(x \wedge 01^{n-1}), \ldots, \mathsf{Th}_k(x \wedge 1^{n-1}0))$ and we get that

$$C''(x) = (\mathsf{Th}_{n/2} \wedge C'(x, N(x))) \vee \mathsf{Th}_{(n/2)+1}(x)$$

is a monotone function which is efficiently computable and weakly one-way. Then, applying the standard hardness amplification process they obtain a one-way function (we refer to [GI12] for the exact detail).

We observe that if we start with a one-way functions in NC^1, then the reduction of [GI12] results with a monotone one-way function which is in NC^1. Then, we use the standard transformation from circuits in NC^1 to formulas. Since this transformation preserves monotonicity and depth, we complete the proof. We note that the above transformation of [GI12] uses threshold functions which are computable by (uniform) formula of logarithmic depth (using sorting networks [AKS83]). ∎

Theorem 4.15. *Let* $f\colon \{0,1\}^n \to \{0,1\}^n$ *be a permutation. If* f *is computable by a formula of size* s *that contains* t *negations, then there exists a deterministic algorithm whose running time is* $2^{2t} \cdot s$ *such that given as input any* $y = f(x)$ *outputs* x*. In particular, if* $s \in \mathsf{poly}(n)$ *and* $t = O(\log n)$*, then the algorithm runs in polynomial-time.*

Proof. Let $f_i\colon \{0,1\}^n \to \{0,1\}$ be the Boolean function corresponding to the i-th output bit of f and $F_i\colon \{0,1\}^n \to \{0,1\}$ be a formula computing f_i. Let $S = \{i \in [n] \mid F_i \text{ is monotone}\}$, i.e., the collection of indices $i \in [n]$ for which F_i contains no negations. Since f has $t < n$ negations, we obtain $|S| \geq n - t$. Let $S_1 = \{i \in S \mid \exists j \in [n], \forall x \in \{0,1\}^n : \ F_i(x) = x_j\}$, i.e., the collection of indices $i \in S$ for which F_i is a dictator function. Let $I_i = \{j \in [n] \mid \mathsf{Inf}_j(f_i) \neq 0\}$, i.e., the set of input variables that f_i depends on.

Consider functions f_ℓ and f_k, where $\ell \neq k \in S$. By Talagrand's inequality (Proposition 2.3),

$$\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] \ \geq \ \Pr_x[f_\ell(x) = 1] \cdot \Pr_x[f_k(x) = 1] + \psi\Big(\sum_{i \in [n]} \mathsf{Inf}_i(f_\ell) \cdot \mathsf{Inf}_i(f_k)\Big).$$

Since f is a permutation, $\Pr_x[f_\ell(x) = 1 \wedge f_k(x) = 1] = 1/4$ and $\Pr x[f_\ell(x) = 1] = \Pr_x[f_k(x) = 1] = 1/2$. Thus, since f_ℓ and f_k are monotone and using the definition of ψ, we get that

$$\sum_{i \in [n]} \mathsf{Inf}_i(f_\ell) \cdot \mathsf{Inf}_i(f_k) = 0.$$

Therefore, $I_\ell \cap I_k = \emptyset$, i.e., f_ℓ and f_k depend on a disjoint set of input variables. Since the above holds for every pair ℓ, k such that $\ell \neq k \in S$, we obtain

$$n \geq \Big|\bigcup_{i \in S} I_i\Big| = \sum_{i \in S} |I_i| = \sum_{i \in S_1} |I_i| + \sum_{i \in S \setminus S_1} |I_i|\,. \tag{4.4}$$

For $i \in S \setminus S_1$, since the function f_i is non-constant we have that $|I_i| \geq 2$. Plugging this into Equation (4.4), we obtain

$$n \geq \sum_{i \in S_1} 1 + \sum_{i \in S \setminus S_1} 2 = 2\,|S| - |S_1| \geq 2(n-t) - |S_1|\,,$$

which implies that $|S_1| \geq n - 2t$.

Given $y = f(x)$, we can invert y and find $x' = x$ using following algorithm:

1. For every $i \in S_1$, we set x'_j to be y_i where j is the only element in the set I_i.
2. Go over all possible assignments on the unassigned variables in x' until $f(x') = y$,
3. Output x'.

After the first step, $|S_1| \geq n-2t$ variables are assigned correctly. The number of unassigned variables is at most $2t$, so that we can try all possible assignments on the remaining unassigned variables in time $2^{2t} \cdot s$, where s is the evaluation time of the permutation. If $s \in \mathsf{poly}(n)$ and $t = O(\log n)$, we get that the above algorithm runs in polynomial-time. ∎

4.5 Hardcore Predicates, Extractors and Learning

In this section we use Corollary 3.3 to generalize the results of [GMOR15, BCO$^+$14] to the setting of formulas. Our proofs rely on Corollary 3.3 and the high-level ideas of [GMOR15, BCO$^+$14].

First, we prove that any formula that computes a hardcore bit must use roughly $\sqrt{n}$ negation gates. Second, we show that any strong $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor with output length m can only be computed by formulas with $\Omega(m \cdot n^{\alpha})$ negation gates.[9] Lastly, we show that formulas with t negations can be learned as fast as circuits with $\log t$ negations. More precisely, we show that any function that can be computed by a formula with t negations can be learned in time $n^{O(t \cdot \sqrt{n}/\varepsilon)}$ to error ε.

Theorem 4.16. *Assume that there exists a family $f = \{f_n\}_{n \in \mathbb{N}}$ of $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions, where each $f_n \colon \{0,1\}^n \to \{0,1\}^n$. Then, for every $\varepsilon > 0$, there exists a family $g^{\varepsilon} = \{g_n\}_{n \in \mathbb{N}}$ of (length-preserving) $(\mathsf{poly}(n), n^{-\omega(1)})$-secure one-way functions for which the following holds. If $h = \{h_n\}_{n \in \mathbb{N}}$ is a $(\mathsf{poly}(n), n^{-\omega(1)})$-secure hardcore predicate for g^{ε}, then for every n sufficiently large, any formula computing h_n contains at least $\Omega(n^{1/2-\varepsilon})$ negations.*

Theorem 4.17. *Let $0 < \alpha < 1/2$ be a constant, and $m = m(n) \geq 100$. Further, suppose that $\mathcal{H} \subseteq \{h \mid h \colon \{0,1\}^n \to \{0,1\}^m\}$ is a family of functions such that each output bit $h_i \colon \{0,1\}^n \to \{0,1\}$ of a function $h \in \mathcal{H}$ is computed by a formula and the total number of negations of a function $h \in \mathcal{H}$ is at most t. Then, if $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor, then $t = \Omega(m \cdot n^{\alpha})$.*

Theorem 4.18. *There is a uniform-distribution learning algorithm that learns any unknown function $f : \{0,1\}^n \to \{0,1\}$ that can be computed by a formula with t negations to error ε in time $n^{O(t \cdot \sqrt{n}/\varepsilon)}$.*

Proof of Theorem 4.16. It follows from a result of Goldmann and Russell [GR00] that under the existence of one-way functions, there exists a one-way function family $g^{\varepsilon} = \{g_n\}_{n \in \mathbb{N}}$ that only admits hardcore predicates with total influence $\Omega(n^{1-\varepsilon})$. Our result follows easily using Corollary 3.3 which states that the total influence of formulas computed with t negations is $O(t \cdot \sqrt{n})$. ∎

[9] We remark that, as [GMOR15], we view the extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ as a family of functions $\mathcal{H}_{\mathsf{Ext}} \stackrel{\text{def}}{=} \{h_w \colon \{0,1\}^n \to \{0,1\}^m \mid h_w = \mathsf{Ext}(\cdot, w),\ \text{where } w \in \{0,1\}^s\}$, i.e., the family of functions obtained from the extractor by fixing its seed. Similarly, every such family can be viewed as a strong extractor in the natural way.

Proof of Theorem 4.17. First, we state a lemma from [GMOR15] (building on techniques of [BG13]) and then we prove the theorem.

Lemma 4.19 ([GMOR15, Lemma 5.7])**.** *Let $0 \leq p \leq 1/2$, $0 \leq \gamma \leq 1/4$, and $\mathcal{H} \subseteq \{h \mid h\colon \{0,1\}^n \to \{0,1\}^m\}$ be a family of functions. Assume that*[10] $\mathbb{E}_{i \leftarrow [m]}[\mathsf{NS}_p(h_i)] \leq \gamma$ *where $h_i\colon \{0,1\}^n \to \{0,1\}$ computes the i-th output bit of some function in $\mathcal{H}$, where $i \in [m]$. Then there exists a distribution D over $\{0,1\}^n$ with min-entropy $H_\infty(D) = n \cdot \log(\frac{1}{1-p})$ such that the statistical distance between $(\mathcal{H}, \mathcal{H}(D))$ and $(\mathcal{H}, \mathcal{U}_m)$ is at least $(1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma})$.*

At this point we are ready to prove the theorem. It is known that for any function $g\colon \{0,1\}^n \to \{0,1\}$ and $p(n) \in (0, 1/2)$, $\mathsf{NS}_p(g) = \mathsf{Inf}(g) \cdot p$ (see e.g., [O'D14]). Thus, it follows from Corollary 3.3 that if $h_i\colon \{0,1\}^n \to \{0,1\}^m$ is a function computed by a formula with t_i negations, then

$$\mathsf{NS}_p(h_i) = O(t_i \cdot \sqrt{n} \cdot p).$$

Therefore, $\mathbb{E}_{i \leftarrow [m]}[\mathsf{NS}_p(h_i)]] = O(\frac{\sum_{i=1}^m t_i}{m} \cdot \sqrt{n} \cdot p) = O(\frac{t}{m} \cdot \sqrt{n} \cdot p)$. This upper bound on the noise sensitivity and our assumption on $\mathcal{H}$ allow us to apply Lemma 4.19 with an appropriate choice of parameters, which we describe next. We choose a $0 \leq p \leq \frac{1}{2}$ such that $n \cdot \log \frac{1}{(1-p)} = n^{\frac{1}{2}-\alpha}$. Observe that we can take $p = O(n^{-\frac{1}{2}-\alpha})$. If $t = o(\frac{t}{m} \cdot n^\alpha)$, then we obtain $\gamma = O(t \cdot \sqrt{n} \cdot p) = o(1)$. By Lemma 4.19, there exists a distribution D of min-entropy $H_\infty(D) = n \log \frac{1}{1-p} = n^{\frac{1}{2}-\alpha}$ for which

$$\begin{aligned}\delta((\mathcal{H}, \mathcal{H}(D)), (\mathcal{H}, \mathcal{U}_m)) &\geq (1 - 2\sqrt{\gamma} - 2^{-0.1m})(1 - 2\sqrt{\gamma}) \\ &= 1 - 2^{-0.1m} - o(1) > 1/2,\end{aligned}$$

which contradicts our assumption that $\mathcal{H}$ is an $(n^{\frac{1}{2}-\alpha}, 1/2)$-extractor. Therefore, $t = \Omega(m \cdot n^\alpha)$. ■

Proof of Theorem 4.18. The proof of the theorem follows by combining Corollary 3.3 with the following fact. By [LMN93], any class of functions $\mathcal{F}$ for which for any $f \in \mathcal{F}$ it holds that $\sum_{S \subset [n], |S| \geq \tau} \widehat{f}(S)^2 \leq \varepsilon$ for $\varepsilon > 0$ and $\tau = \tau(\varepsilon, n)$, then $\mathcal{F}$ can be learned from uniform examples in time $\mathsf{poly}(n^\tau, 1/\varepsilon)$. Using Corollary 3.3 we have that $\mathsf{Inf}(f) \leq O(t) \cdot \sqrt{n}$ for every f as in the statement. This implies that

$$\sum_{S \subset [n], |S| \geq \Omega(t) \cdot \sqrt{n}/\varepsilon} \hat{f}(S)^2 \leq \varepsilon$$

from which the theorem follows by [LMN93]. ■

5 Open Problems

In this paper we study the power of negation gates in the model of Boolean De Morgan formulas. Among other results, we proved that any hardcore predicate on n bits requires roughly $\Omega(\sqrt{n})$ negation gates, that any extractor of 100 bits on n-bit sources of min-entropy $n^{1/2-\varepsilon}$ requires formulas with at least $\Omega(n^\varepsilon)$ negation gates and that any one-way permutation on n bits requires $\omega(\log n)$ negations. One the other hand, the best upper bound we are aware of for these primitives is the universal bound of $O(m \cdot n)$ negations that applies for any function that outputs m bits [Nec62, Mor09]. Improving these lower bounds or providing matching upper bounds would be very interesting.

[10] The original statement in [GMOR15, Lemma 5.7] assumes that $\mathsf{NS}_p(h_i)] \leq \gamma$ for every $i \in [m]$. We observe that a simple extension of their proofs also work with the assumption $\mathbb{E}_{i \leftarrow [m]}[\mathsf{NS}_p(h_i)] \leq \gamma$.

Morizumi [Mor09] showed that any formula F can be transformed into a formula F' that has only $\lceil n/2 \rceil$ negations and such that $L(F') \leq L(F) \cdot O(n^{6.3})$. His transformation uses as a building block the monotone formula that compute the threshold function of Valiant [Val84] which gives a short but non-explicit construction. We leave open the question whether one can come up with an explicit and efficient transformation from any formula to a formula with few negations.

Lastly, we mention the important open problem of determining the shrinkage exponent of monotone formulas.

Acknowledgments

We thank Avishay Tal for helpful discussions and, in particular, for a discussion that led to the proof of Theorem 3.1. We also thank Andrej Bogdanov, Moni Naor, Alon Rosen and Eylon Yogev for inspiring discussions.

References

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, 1987.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in NC^0. *SIAM J. Comput.*, 36(4):845–888, 2006.

[AKS83] Miklós Ajtai, János Komlós, and Endre Szemerédi. An $o(n \log n)$ sorting network. In *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, STOC*, pages 1–9. ACM, 1983.

[AM05] Kazuyuki Amano and Akira Maruoka. A superpolynomial lower bound for a circuit computing the clique function with at most $(1/6) \log \log n$ negation gates. *SIAM J. Comput.*, 35(1):201–216, 2005.

[And85] Alexander E. Andreev. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Doklady Akademii Nauk SSSR*, 288:1033–1037, 1985. In Russian. English translation in *Soy. Math. Dokl.* 31 (1985), 530–534.

[And87] Alexander E. Andreev. On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes. *Moscow Univ. Math. Bull.*, 42:63–66, 1987. In Russian.

[BCO+14] Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. *CoRR*, abs/1410.8420, 2014.

[BG13] Andrej Bogdanov and Siyao Guo. Sparse extractor families for all the entropy. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science, ITCS*, pages 553–560, 2013.

[BNT98] Robert Beals, Tetsuro Nishino, and Keisuke Tanaka. On the complexity of negation-limited boolean networks. *SIAM J. Comput.*, 27(5):1334–1347, 1998.

[BT96] Nader H. Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *J. ACM*, 43(4):747–770, 1996.

[BU99] Christer Berg and Staffan Ulfberg. Symmetric approximation arguments for monotone lower bounds without sunflowers. *Computational Complexity*, 8(1):1–20, 1999.

[CKK+14] Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. In *Proceedings of the 29th Conference on Computational Complexity, CCC*, pages 262–273, 2014.

[CKS14] Ruiwen Chen, Valentine Kabanets, and Nitin Saurabh. An improved deterministic #sat algorithm for small De Morgan formulas. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 165–176, 2014.

[DZ94] Moshe Dubiner and Uri Zwick. How do read-once formulae shrink? *Combinatorics, Probability & Computing*, 3:455–469, 1994.

[Fis75] Michael. J. Fischer. The complexity of negation-limited networks–a brief survey. *Automata Theory and Formal Languages*, 33:71–82, 1975.

[GI12] Oded Goldreich and Rani Izsak. Monotone circuits: One-way functions versus pseudorandom generators. *Theory of Computing*, 8(1):231–238, 2012.

[GKR12] Anat Ganor, Ilan Komargodski, and Ran Raz. On the noise stability of small De Morgan formulas. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:174, 2012.

[GMOR15] Siyao Guo, Tal Malkin, Igor Carboni Oliveira, and Alon Rosen. The power of negations in cryptography. *To appear in Proceedings of the 12th Theory of Cryptography Conference, TCC*, 2015. Available as ECCC Report 2015/22.

[GR00] Mikael Goldmann and Alexander Russell. Spectral bounds on general hard-core predicates. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science, STACS*, pages 614–625, 2000.

[Hås87] Johan Håstad. One-way permutations in NC0. *Inf. Process. Lett.*, 26(3):153–155, 1987.

[Hås98] Johan Håstad. The shrinkage exponent of de Morgan formulas is 2. *SIAM J. Comput.*, 27(1):48–64, 1998.

[HR00] Danny Harnik and Ran Raz. Higher lower bounds on monotone size. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, STOC*, pages 378–387, 2000.

[HRY95] Johan Håstad, Alexander A. Razborov, and Andrew Chi-Chih Yao. On the shrinkage exponent for read-once formulae. *Theor. Comput. Sci.*, 141(1&2):269–282, 1995.

[IK14] Russell Impagliazzo and Valentine Kabanets. Fourier concentration from shrinkage. In *Proceedings of the 29th Conference on Computational Complexity, CCC*, pages 321–332, 2014.

[IMT09] Kazuo Iwama, Hiroki Morizumi, and Jun Tarui. Negation-limited complexity of parity and inverters. *Algorithmica*, 54(2):256–267, 2009.

[IMZ12] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 111–119, 2012.

[IN93] Russell Impagliazzo and Noam Nisan. The effect of random restrictions on formula size. *Random Struct. Algorithms*, 4(2):121–134, 1993.

[Juk12] Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012.

[Khr71] V.M. Khrapchenko. A method of determining lower bounds for the complexity of π schemes. *Matematischi Zametki*, 10"1:83–92, 1971. In Russian.

[KRT13] Ilan Komargodski, Ran Raz, and Avishay Tal. Improved average-case lower bounds for DeMorgan formula size. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 588–597, 2013.

[Lee09] Troy Lee. A note on the sign degree of formulas. *CoRR*, abs/0909.4607, 2009.

[LMN93] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

[Mar58] A. A. Markov. On the inversion complexity of a system of functions. *J. ACM*, 5(4):331–334, 1958.

[Mor09] Hiroki Morizumi. Limiting negations in formulas. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming, ICALP*, pages 701–712, 2009.

[Nec62] E. I. Nechiporuk. On the complexity of schemes in some bases containing nontrivial elements with zero weights. *Problemy Kibernetiki*, 8:123–160, 1962. In Russian.

[Nis91] Noam Nisan. Pseudorandom bits for constant depth circuits. *Combinatorica*, 11(1):63–70, 1991.

[NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 51(2):231–262, 2004.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.

[PZ93] Mike Paterson and Uri Zwick. Shrinkage of de Morgan formulae under restriction. *Random Struct. Algorithms*, 4(2):135–150, 1993.

[Raz85] A. A. Razborov. Lower bounds for the monotone complexity of some boolean functions. *Soviet Math. Dokl.*, 31:354–357, 1985. In Russian.

[Ros15] Benjamin Rossman. Correlation bounds against monotone NC^1. *To appear in Proceedings of the 30th Conference on Computational Complexity, CCC*, 2015. Available at `http://research.nii.ac.jp/~rossman/mNC1.pdf`.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RW89] Ran Raz and Avi Wigderson. Probabilistic communication complexity of boolean relations (extended abstract). In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS*, pages 562–567, 1989.

[Shi00] Yaoyun Shi. Lower bounds of quantum black-box complexity and degree of approximating polynomials by influence of Boolean variables. *Inf. Process. Lett.*, 75(1-2):79–83, 2000.

[ST04] Shao Chin Sung and Keisuke Tanaka. Limiting negations in bounded-depth circuits: An extension of markov's theorem. *Inf. Process. Lett.*, 90(1):15–20, 2004.

[Sub61] B.A Subbotovskaya. Realizations of linear function by formulas using $+, \cdot, -$. *Doklady Akademii Nauk SSSR*, 136:3:553–555, 1961. In Russian.

[SW91] Miklos Santha and Christopher B. Wilson. Polynomial size constant depth circuits with a limited number of negations. In *Proceedings of the 8th Annual Symposium on Theoretical Aspects of Computer Science, STACS*, pages 228–237, 1991.

[Tal96] Michel Talagrand. How much are increasing sets positively correlated? *Combinatorica*, 16(2):243–258, 1996.

[Tal14] Avishay Tal. Shrinkage of De Morgan formulae by spectral techniques. In *Proceedings of the 55th IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 551–560, 2014.

[Tar88] Éva Tardos. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica*, 8(1):141–142, 1988.

[TNB96] Keisuke Tanaka, Tetsuro Nishino, and Robert Beals. Negation-limited circuit complexity of symmetric functions. *Inf. Process. Lett.*, 59(5):273–279, 1996.

[Val84] Leslie G. Valiant. Short monotone formulae for the majority function. *J. Algorithms*, 5(3):363–366, 1984.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de