Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 26 (2015)

Negation-Limited Formulas

Siyao Guo * Ilan Komargodski †

Abstract

We give an efficient structural decomposition theorem for formulas that depends on their negation complexity and demonstrate its power with the following applications:

- 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.
- We give an efficient transformation of formulas with t negation gates to circuits with $\log t$ negation gates. This transformation provides a generic way to cast results for negation-limited circuits to the setting of negation-limited formulas. For example, using a result of Rossman (CCC '15), we obtain an average-case lower bound for formulas of polynomial-size on n variables with $n^{1/2-\varepsilon}$ negations.

In addition, we prove a lower bound on the number of negations required to compute one-way permutations by polynomial-size formulas.

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 two. 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 (see [Hås98, Tal14]), whereas in the setting of *monotone* formulas, exponential lower bounds are known (see [HR00, GP14] and references therein).[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

*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.

†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.

[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 monotone function on n inputs such that every monotone formula that computes it must be of size $2^{\Omega(n/\log n)}$ (see [GP14]).

ISSN 1433-8092

any Boolean function on n variables by a formula, and moreover, any formula can be efficiently transformed into a formula that computes the same function but contains at most $\lceil n/2 \rceil$ negation gates (see [Nec62, Mor09] and [Juk12]).

In this paper, we continue this line of research and study negation-limited formulas with two main perspectives. The first perspective, which is motivated by bridging the gap between monotone and non-monotone formulas, is that we view negation-limited formulas as a natural extension of monotone formulas and try to extend various complexity properties of monotone formulas to the negation-limited setting. The second perspective, which is motivated by separating the power of circuits and formulas, is that we view negation-limited formulas as a restricted form of negation-limited circuits and ask natural questions about negation-limited circuits in the setting of formulas.

1.1 Our Contributions

The main tool: efficient decomposition of negation-limited formulas

We prove an *efficient* structural decomposition theorem for negation-limited formulas. 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 (in polynomial-time) into $T+1$ 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 $2s$. That is, roughly speaking, we are able to (efficiently) 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 theorem serves us as the main tool to extend results for *monotone* formulas to *negation-limited* formulas, and to leverage results concerning negation-limited *circuits* to negation-limited *formulas*. We give two applications to demonstrate the usage of our main tool.

Application 1: shrinkage of negation-limited formulas 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] who proved a lower bound of $\Omega(n^\Gamma)$ on 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 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 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

[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 $f\colon \{0,1\}^n \to \{0,1\}$ 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.

[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]).

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.

Application 2: efficient transformation from negation-limited formulas to circuits. The decomposition theorem gives a way to efficiently transform formulas with t negations into circuit with roughly $\log t$ negations. Specifically, we prove that a formula of size s that contains t negations can be transformed into a circuit of size $2s + O(t \cdot \log t)$ that contains only $\log t + O(1)$ negation gates.

This transformation also provides a generic way to cast results for negation-limited circuits to the setting of negation-limited formulas. Informally, algorithms for circuits with $\log t$ negation gates will apply for formulas with t negation gates (with almost the same size and depth), and lower bound for circuits with $\log t$ negations will imply lower bounds for formulas with t negation gates (with almost the same size and depth). As an example, this allows us to cast the recent average-case lower bound for mNC^1 [Ros15], lower bounds for several cryptographic primitives [GMOR15], and the upper bound on learning circuits with few negations [BCO$^+$15] to the setting of negation-limited formulas as we elaborate in Section 4.2.1.

More Results

Lower bound on negation complexity of one-way permutations. We prove a lower bound for implementing one-way permutations by negation-limited formulas. Specifically, we show that every permutation on n bits that can be computed by a formula of size s that contains t negation gates can be inverted (on every image) in time $2^{2t} \cdot s$. This implies, in particular, that every implementation of a one-way permutation as a polynomial-size formula must contain at least $\omega(\log n)$ negation gates. As a comparison, Guo et al. [GMOR15] left open the question of whether one-way permutations are computable by circuits that contain a single negation gate.

Upper bound on the total influence. Total influence has many applications in various areas of theoretical computer science. Most relevant to our context, it serves as the main tool in recent studies of negation-limited circuits in computational learning [BCO$^+$15] and cryptography [GMOR15].

The literature on negation complexity defines a measure, $a(\cdot)$, called "alternation complexity" which denotes the maximal number of times a function $f\colon \{0,1\}^n \to \{0,1\}$ changes its value along a chain (i.e., a monotone sequence of strings) starting at 0^n and ending at 1^n. Blais et al. [BCO$^+$15] proved (using their inefficient decomposition theorem) that for any function f it holds that $\mathsf{Inf}(f) \leq O(a(f) \cdot \sqrt{n})$. We give a simple direct probabilistic argument for this fact.

1.2 Related Work

An inefficient decomposition theorem for negation-limited *circuits* into monotone circuits explicitly appeared in [BCO$^+$15]. They proved that any function f that can be computed using a circuit with t negations can be decomposed into $T+1$ 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.[4] An efficient version of this decomposition theorem (with related parameters) appeared explicitly in [GMOR15] and implicitly in [AM05, Ros15].

Besides the above, the power of negations in different models has been studied in many works including [Mar58, Nec62, Fis75, SW91, RW89, TNB96, BNT98, ST04, AM05, Mor09, IMT09]. For

[4] We refer to this decomposition as "inefficient" since the decomposed monotone components (i.e., the g_i's) may have exponential size.

more information on negations in complexity theory 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.

Efficient decomposition of negation-limited formulas. Using the theorem of Nechiporuk [Nec62] it is quite straightforward to cast the decomposition theorem of [BCO$^+$15] to the setting of negation-limited formulas which results in the same statement except that $T = O(t)$ (rather than $T = O(2^t)$). More precisely, it gives that and function f can be rewritten as $f(x) \equiv h(g_1(x), \ldots, g_T(x))$, where $T = O(t)$, h is either the parity function or its negation and each g_i is a monotone function. Unfortunately, such a decomposition is not enough for us since it is inefficient, in particular, it does not preserve the size or depth of the original formula. Note that the efficiency of the decomposition was mostly not an issue in [BCO$^+$15, GMOR15], whereas for us it is crucial since, for example, shrinkage is a combinatorial property that general circuits do not have (unlike formulas).

To overcome this we prove an *efficient* version in which the resulting formula has almost the same size and depth as the original one.[5] Technically, our 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].

As an applications of this theorem, we prove the shrinkage result and the transformation from negation-limited formulas to circuits. The shrinkage theorem 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. To get the transformation result, we use our efficient decomposition theorem for formulas, view h as a circuit on t inputs, and apply Fischer's transformation [Fis75] (see also [BNT98]) to implement h with $\lceil \log_2(t+1) \rceil$ negations.

Negation complexity of one-way permutations. Our lower bound on the number of negations required to compute one-way permutations relies crucially on the fact that the fan-out of formulas is 1. We take advantage of this fact together with Talagrand's inequality [Tal96] in a way that might be of independent interest. We emphasize that previously it was known that one-way permutations cannot be computed by a monotone circuit.

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 central tool: the decomposition theorem for negation-limited formulas. In Sections 4.1 to 4.4 we give the statements of the shrinkage result, the transformation from negation-limited formulas to negation-limited circuits, the lower bound for one-way permutations, and the influence bound for negation-limited formulas.

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 $x \leftarrow \mathcal{X}$

[5] We note that our transformation is efficient in a strong sense: (1) it can be implemented in polynomial-time in the size of the input formula, and (2) it results with a formula of polynomial-size (close to the size of the input formula).

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).

Boolean 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 $f\colon \{0,1\}^n \to \{0,1\}$ 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 $f\colon \{0,1\}^n \to \{0,1\}^m$ 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* 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 2.1 ([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 and Influence

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 $f\colon \{0,1\}^n \to \mathbb{R}$. It follows that every function $f\colon \{0,1\}^n \to \mathbb{R}$ can be represented as

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

where $\widehat{f}\colon \{0,1\}^n \to \mathbb{R}$, 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 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 2.2 (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 and One-Way Permutations

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).

3 Efficient Decomposition for Negation-Limited Formulas

In this section we present our main tool, an efficient structural decomposition theorem for formulas which, intuitively, pushes all negation gates to the root of the formula.

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 $2s$.*

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

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 (i.e., does not include a NOT gate) or includes one NOT gate located at its root (i.e., it is anti-monotone). This process results with at most $t+1$ subformulas.

For each subformula F_i with more than two subformula children, find a subformula F_i' of F_i with exactly two subformula 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 subformula children. Continue doing this until all subformulas have at most two subformula children. This process results with the desired number of subformulas, $2(t+1)$, since the above process can happen at most the original number of subformulas. ∎

Proof of Theorem 3.1. Let F be as in the lemma. Apply the decomposition from Claim 3.2 on F to get the subformulas $F_1, \ldots, F_{T'}$, 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 subformula F_1, and that the subformula 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 $k_1^{(2)}$ and $k_1^{(3)}$ are the roots of F_2 and F_3, respectively, $k_2^{(1)}$ 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 3.2, 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 subformulas 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

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

Now, we observe that the right hand side can be rewritten as $F''(G_1, \ldots, G_6, z_2, z_3)$, where F'' is read-once and $G_1, \ldots, G_6$ 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. ■

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. Let $f\colon \{0,1\}^n \to \{0,1\}$ 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. A p-random restriction is a restriction as above 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 4.1 (Shrinkage exponent)**.** Let $\mathcal{F}$ be a class of formulas. The shrinkage exponent of $\mathcal{F}$ is said to be Γ if for any $F \in \mathcal{F}$

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

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. It is known that (1) $\Gamma = 2$ [Hås98, Tal14], (2) $\Gamma^* = \log_{\sqrt{5}-1} 2 \approx 3.27$ [DZ94, HRY95], and (3) for every $t \geq 0$ it holds that $\Gamma^* \geq \Gamma_t \geq \Gamma_{t+1} \geq \Gamma = 2$. 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 4.2. *Let F be a formula that contains $t > 0$ negation gates. It holds that*

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

Proof of Theorem 4.2. 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

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

■

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 4.2 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 $O(p^\Gamma \cdot t + 1)$. In the following corollary we state a shrinkage result parameterized by t, the number of negations, p, the restriction parameter, and $L(F)$, the formula size.

Corollary 4.3. *Let F be a formula that contains $t = t(L(F)) > 0$ negations and let $c > 0$ be a constant. Then, for $p \geq \sqrt[\Gamma_0]{(c \cdot t)/L(F)}$, it holds that*

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

4.2 Efficient Transformation from Negation-Limited Formulas to Circuits

In this section we show that negation-limited formulas can be transformed into negation-limited circuits with exponentially smaller number of negations with almost linear blowup in the size and depth. An inefficient transformation was previously known due to the theorems of Markov [Mar58] and Nechiporuk [Nec62].[6]

[6] By the theorem of Nechiporuk [Nec62], the decrease of a function computable by a formula with t negations is t. Then, by the theorem of Markov [Mar58], any function with decrease t is computable by a circuit with $\lceil \log(t+1) \rceil$ negations. The size of the resulting circuit, however, is unbounded (i.e., it can be exponential in the number of inputs).

Theorem 4.4. *Let $F\colon \{0,1\}^n \to \{0,1\}$ be a formula of size s and depth d and t negations, then there is a circuit C of size s', depth d' and t' negations computing F such that $s' = 2s + O(t \log t)$, $d' = d + O(\log t)$ and $t' = \log t + O(1)$.*

Fischer's theorem [Fis75] can efficiently transform negation-limited formulas with t negations into negation-limited circuits with $\log n$ negations. Our theorem combines Fischer's theorem and our decomposition theorem (Theorem 3.1) to efficiently transform the negation-limited formulas with t negations into negation-limited circuits with $\log t$ negations.

Proof. Our decomposition theorem (Theorem 3.1) states that the function f computed by F can be written as $f(x) = h(g_1(x), \ldots, g_T(x))$ where $T \leq 15(t+1)$, $g_1, \ldots, g_T\colon \{0,1\}^n \to \{0,1\}$ are computable by monotone formulas of total size at most $2s$ (also depth at most d) and $h\colon \{0,1\}^T \to \{0,1\}$ is computable by a read-once formula. We use the efficient version of Markov's theorem to get a circuit with few negations that compute h.

Proposition 4.5 ([Fis75, BNT98])**.** *If a function on n variables can be computed by circuit over a basis that includes* AND*,* OR *and* NOT *gates of size s and depth d, then it can be computed by a circuit of size at most $2s + O(n \log n)$ and depth $d + O(\log n)$ using at most $\lceil \log(n+1) \rceil$ negations.*

The read-once formula computing h has input size T so that size is at most T and depth is at most $\log T$. By the above theorem, we conclude that h can be computed by a circuit of size at most $2T + O(T \log T) = O(t \log t)$ and depth $O(\log T)$ using at most $\lceil \log(T+1) \rceil = \log t + O(1)$ negations. It is easy to see we can compose the circuit for h with formulas for $g_1, \ldots, g_T$ to compute f. Since $g_1, \ldots, g_T$ are computable by monotone formulas of total size at most $2s$ and depth at most d, we can further conclude that f are computable by a circuit of size at most $2s + O(t \log t)$, depth $d + O(\log t)$ and $O(\log t)$ negations. ∎

4.2.1 Applications

In this section we exemplify the usefulness of Theorem 4.4.

Average-case lower bounds for negation-limited formulas. An average-case computation (a.k.a. approximate computation) of a function $f\colon \{0,1\}^n \to \{0,1\}$ 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. His bound for mNC^1 extends to circuits in NC^1 with at most $(1/2 - \varepsilon) \log n$ negations. Using Theorem 4.4 and [Ros15], we get the following corollary.

Corollary 4.6. *For every $\varepsilon > 0$, there is an explicit function $f\colon \{0,1\}^n \to \{0,1\}$ such that for every polynomial-size formula F with $n^{1/2-\varepsilon}$ negations, it holds that $\Pr_{x \leftarrow \{0,1\}^n}[F(x) = f(x)] \leq 1/2 + o(1)$.*

We remark that Corollary 4.6 crucially relies on that the transformation in Theorem 4.4 is efficient.

Cryptography in negation-limited formulas. One of the goals of cryptography is to study how simple cryptographic primitives can be, where simplicity can be measured by e.g., the required assumptions, the circuit depth and more. Recently, Guo et al. [GMOR15] (following on [GI12])

proved lower bounds on the number of negations required to represent many cryptographic primitives as circuits. 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] and concrete examples in [Hås87, NR04]). Using Theorem 4.4, one can easily cast some of the results of [GMOR15] to the setting of negation-limited formulas and obtain *exponentially higher* lower bounds on several primitives including pseudorandom functions, hardcore predicates and extractors. (We refer the reader to [GMOR15] for the relevant notation and definitions.)

Corollary 4.7. *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.*

Corollary 4.8. *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.*

Corollary 4.9. *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 $t = \Omega(n^\alpha)$.*[7]

Uniform-distribution learnability of negation-limited formulas. 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+15] 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 $n^{O(2^t\cdot\sqrt{n}/\varepsilon)}$. Using Theorem 4.4 we obtain the following corollary.

Corollary 4.10. *There is a uniform-distribution learning algorithm that learns any Boolean function f on n variables that can be computed by a formula with t negations to error ε in time $n^{O(t\cdot\sqrt{n}/\varepsilon)}$.*

4.3 One-Way Functions and Permutations in Negation-Limited Formulas

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 4.11) 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, in Theorem 4.12, we show that any one-way permutation is not computable by a formula that has $O(\log n)$ negations.

Observation 4.11. *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

[7] We remark that the above bound can be further improved to $\Omega(m \cdot n^\alpha)$ if one combines the proof of [GMOR15] (with slight modifications) and the total influence upper bound given in Theorem 4.13.

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 4.12. *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 2.2),

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

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

$$n \geq \Big|\bigcup_{i \in S} I_i\Big| = \sum_{i \in S} |I_i| = \sum_{i \in S_1} |I_i| + \sum_{i \in S \setminus S_1} |I_i| \,. \tag{4.2}$$

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 (4.2), 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. ■

4.4 Total Influence of Negation-Limited Formulas

In this section we prove a general connection between total influence and negation complexity of Boolean functions.

Theorem 4.13. *For any function $f\colon \{0,1\}^n \to \{0,1\}$, it holds that $\mathsf{Inf}(f) \leq O\left(a(f) \cdot \sqrt{n}\right)$, where $\mathsf{Inf}(f)$ is the total influence of f and $a(f)$ is the alternating complexity of f.*

In particular, if f can be computed by a formula with t negations, then $\mathsf{Inf}(f) \leq O\left(t \cdot \sqrt{n}\right)$.

A proof of Theorem 4.13 was given (somewhat implicitly in this full generality) in [BCO+15] using their (inefficient) decomposition theorem. We show a direct probabilistic proof for Theorem 4.13 which arguably simplifies their arguments and might be of independent interest.

We note that the bound in Theorem 4.13 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 $f\colon \{0,1\}^n \to \{0,1\}$ 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.

Proof of Theorem 4.13. The "In particular" part of the above theorem follows by Nechiporuk's theorem (see Theorem 2.1). We proceed with the main part.

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

$$\frac{\mathsf{Inf}(f)}{n} = \Pr_{x \leftarrow \{0,1\}^n, i \leftarrow [n]}[f(x) \neq f(x^{\oplus i})]. \tag{4.3}$$

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 (4.3), we get that

$$\frac{\mathsf{Inf}(f)}{n}=\Pr_{\mathcal{X}\leftarrow\mathcal{C},(x_i,x_{i+1})\leftarrow\mathcal{X}}[f(x_i)\neq f(x_{i+1})]. \tag{4.4}$$

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

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

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 (4.4) we get that $\mathsf{Inf}(f)\leq O(a(f)\cdot\sqrt{n})$. ∎

5 Open Problems

In this paper we study the power of negation gates in the model of Boolean De Morgan formulas. Our shrinkage result (Theorem 4.2) implies that as long as $t\ll L(F)$, the shrinkage exponent of F is essentially Γ_0, the shrinkage exponent of monotone formulas. In addition, we showed that formulas with t negation gates can be efficiently transformed into circuits with roughly $\log t$ negation gates without incurring significant blow-up in size or depth.

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 thank Andrej Bogdanov, Moni Naor, Alon Rosen and Eylon Yogev for inspiring discussions. Finally, we thank the reviewers for their constructive comments.

References

[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, 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.

[BCO+15] Eric Blais, Clément L. Canonne, Igor Carboni Oliveira, Rocco A. Servedio, and Li-Yang Tan. Learning circuits with few negations. *To appear in Proceedings of the 19th International Workshop on Randomization and Computation, RANDOM*, 2015. Available at `http://arxiv.org/abs/1410.8420`.

[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.

[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.

[GMOR15] Siyao Guo, Tal Malkin, Igor Carboni Oliveira, and Alon Rosen. The power of negations in cryptography. In *Proceedings of the 12th Theory of Cryptography Conference, TCC*, pages 36–65, 2015.

[GP14] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Proceedings of the 46th Annual Symposium on Theory of Computing, STOC*, pages 847–856, 2014.

[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.

[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.

[Mar58] Andrey 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] Eduard 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.

[Ros15] Benjamin Rossman. Correlation bounds against monotone NC^1. In *Proceedings of the 30th Conference on Computational Complexity, CCC*, pages 392–411, 2015.

[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.

[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] Bella 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.

[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