On the Limits of Gate Elimination

Alexander Golovnev*[1], Edward A. Hirsch[2], Alexander Knop[3], and Alexander S. Kulikov[4]

1 **New York University, USA**
2 **St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**
3 **St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**
4 **St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences, Russia**

Abstract

Although a simple counting argument shows the existence of Boolean functions of exponential circuit complexity, proving superlinear circuit lower bounds for *explicit* functions seems to be out of reach of the current techniques. There has been a (very slow) progress in proving linear lower bounds with the latest record of $3\frac{1}{86}n - o(n)$. All known lower bounds are based on the so-called gate elimination technique. A typical gate elimination argument shows that it is possible to eliminate several gates from an optimal circuit by making one or several substitutions to the input variables and repeats this inductively. In this note we prove that this method cannot achieve linear bounds of cn beyond a certain constant c, where c depends only on the number of substitutions made at a single step of the induction.

1998 ACM Subject Classification F.1.1 Models of Computation

Keywords and phrases circuit complexity, lower bounds, gate elimination

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.46

1 Introduction

One of the most important and at the same time most difficult questions in theoretical computer science is proving circuit lower bounds. A binary Boolean circuit is a directed acyclic graph with nodes of in-degree either 0 or 2. Nodes of in-degree 0 are called inputs and are labeled by variables $x_1, \ldots, x_n$. Nodes of in-degree 2 are called gates and are labeled by binary Boolean functions. One of the nodes is additionally labeled as the output of the circuit. The output gate computes a Boolean function $\{0,1\}^n \to \{0,1\}$ in a natural way. The size of a circuit C is defined as the number of gates in C and is denoted by $\texttt{gates}(C)$. By $\texttt{inputs}(C)$ we denote the number of inputs of C. A circuit complexity measure μ is a function assigning each circuit a non-negative real number. In particular, `gates` and `inputs` are circuit complexity measures.

By B_n we denote the set of all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$. For a circuit complexity measure μ and a function $f \in B_n$, by $\mu(f)$ we denote the minimum value of $\mu(C)$ over all circuits C computing f. For example, $\texttt{gates}(f)$ is the miniumum size of a circuit computing f.

* This research is partially supported by NSF grant 1319051.

By comparing the number of small size circuits with the total number 2^{2^n} of Boolean functions of n variables, one concludes that almost all such functions have circuit size at least $\Omega(\frac{2^n}{n})$. This was shown by Shannon in 1949 [30]. However we still do not have an example of a function from NP that requires circuits of superlinear size. The currently strongest known lower bound is $(3+\frac{1}{86})n - o(n)$ [11].

The lack of strong lower bounds is a consequence of the lack of methods for proving lower bounds for general circuits. Practically, the only known method for proving lower bounds is the gate elimination method. We illustrate this method with a simple example. Consider the function $\mathrm{MOD}^n_{3,r}\colon \{0,1\}^n \to \{0,1\}$ which outputs 1 if and only if the sum of n input bits is congruent to r modulo 3. One can prove that $\texttt{gates}(\mathrm{MOD}^n_{3,r}) \geq 2n-4$ for any $r \in \{0,1,2\}$ by induction on n. The base case $n \leq 2$ clearly holds. Assume that $n \geq 3$ and consider an optimal circuit C computing $\mathrm{MOD}^n_{3,r}$ and its topologically first (with respect to some topological ordering) gate G. This gate is fed by two different variables x_i and x_j (if they were the same variable, the circuit would not be optimal). A crucial observation is that it cannot be the case that the out-degrees of both x_i and x_j are equal to 1. Indeed, in this case the whole circuit would depend on x_i and x_j through the gate G only. In particular, the four ways of fixing the values of x_i and x_j would give at most two different subfunctions (corresponding to $G=0$ and $G=1$), while $\mathrm{MOD}^n_{3,r}$ has three such different subfunctions: $\mathrm{MOD}^{n-2}_{3,0}$, $\mathrm{MOD}^{n-2}_{3,1}$, and $\mathrm{MOD}^{n-2}_{3,2}$. Assume, without loss of generality, that x_i has out-degree at least 2. We then substitute $x_i \leftarrow 0$, eliminate the gates fed by x_i from the circuit and proceed by induction. The eliminated gates are those fed by x_i. After the substitution, each such gate computes either a constant or a unary function of the other input of the gate, so can be eliminated. The resulting function computes $\mathrm{MOD}^{n-1}_{3,r}$. Thus we get by induction: $\texttt{gates}(\mathrm{MOD}^n_{3,r}) \geq \texttt{gates}(\mathrm{MOD}^{n-1}_{3,r}) + 2 \geq (2(n-1)-4)+2 = 2n-4$. This proof was given by Schnorr in 1984 [29]. In fact, it works for a wider class of functions $Q^n_{2,3}$ containing functions that have at least three different subfunctions with respect to any two variables.

This example reveals the main idea of the gate elimination process: a lower bound is proved inductively by finding at each step an appropriate substitution that eliminates many gates from the given circuit. At the same time, using just bit-fixing substitutions is not enough for proving even stronger than $2n$ lower bounds: the class $Q^n_{2,3}$ contains, in particular, a function THR^n_2 that outputs 1 iff $\sum_{i=1}^n x_i \geq 2$ whose circuit complexity is known to be at most $2n + o(n)$ [10] (see also Theorem 2.3 in [35]). For this reason, known proofs of stronger lower bounds use various additional tricks.

- One can use amortized analysis of the number of eliminated gates. For example, one can show that at each step one can either find a substitution that eliminates 3 gates *or* a pair of consecutive substitutions, the first one eliminating 2 gates and the next one eliminating 4 gates.
- They also substitute variables not just by constants but by affine functions, quadratic functions, and even arbitrary functions of other variables.
- In order to amortize for steps that eliminate too few gates, they also use more intricate complexity measures that combine the number of gates with the number of variables or other quantities.

We give an overview of known lower bounds and used tricks in Section 2.

One can guess that the gate elimination method changes only the top of a circuit in few places and thus cannot eliminate many gates. In general, this intuition fails (it is easy to present examples where a single substitution greatly simplifies a function, in particular, every substitution to a function of the highest possible complexity $2^n/n$ (see Theorem 2.1

and below in [35]) lowers the complexity of this function almost twice as for a function of $n-1$ variables it cannot exceed $2^{n-1}/(n-1)+o(2^{n-1}/(n-1))$. However, in this paper we manage to make this intuition work for specially designed functions that compose gadgets satisfying certain rather general properties with arbitrary base functions. We show that certain formalizations of the gate elimination method cannot prove superlinear lower bounds. We prove that one cannot reduce the complexity of the designed functions by more than a constant using any constant number of substitutions of any type (that is, we allow to substitute variables by arbitrary functions). The complexity of a function may be counted as any complexity measure (i.e., a nonnegative function of a circuit) varying from the number of gates to any subadditive function. For recently popular measures that combine the number of gates with the number of inputs we prove a stronger result (namely, one cannot prove lower bounds beyond cn for a certain specific constant c; this constant may depend on the number m of consecutive substitutions made in one step of the induction but does not depend on the substitutions themselves, $m=1$ or 2 in modern proofs).

The paper is organized as follows. In Section 2 we list known proofs based on gate elimination, we discuss their differences and limits. Section 3 presents several examples that lead us to the main questions of this work. This section contains main results of the paper: provable limits of the gate elimination method for various complexity measures. Section 4 contains a brief overview of the known barriers for proving circuit lower bounds. Finally, Section 5 concludes the work with open questions.

2 Known Lower Bounds Proofs

Improving Schnorr's $2n$ lower bound proof mentioned above is already a non-trivial task. It can be the case that all variables in the given circuit feed two parity gates. In this case, substituting any variable by any constant eliminates just two gates from this circuit. In 1977, Stockmeyer [31] used the following clever trick to prove a $2.5n-\Theta(1)$ lower bound for many symmetric functions including all MOD_m^n functions for constant $m\geq 3$. The idea is to eliminate five gates by *two* consecutive substitutions. This time, instead of substituting $x_i\leftarrow c$ where $c\in\{0,1\}$ we substitute $x_i\leftarrow f, x_j\leftarrow f\oplus 1$ where f is an *arbitrary function* that does not depend on x_i and x_j. One should be careful with such substitutions as they potentially might produce a subfunction outside of the class of functions for which we are currently proving a lower bound by induction. At the same time, one can see that, for example, $\mathrm{MOD}_{3,0}^n$ function turns into $\mathrm{MOD}_{3,2}^{n-2}$ function under the substitution $x_i\leftarrow f, x_j\leftarrow f\oplus 1$. Indeed, this substitution just forces the sum of x_i and x_j to be equal to 1 (both over integers and over the field of size two).

In 1984, Blum [5], following the work by Paul [25], proved a $3n-o(n)$ lower bound for an artificially constructed Boolean function of $n+3\log n+3$ variables. The input of this function consists of n variables $X=\{x_1,\ldots,x_n\}$ and $3\log n+3$ variables A. The following "universality" property of this function is essential for Blum's proof: for any two variables $x_i,x_j\in X$ one can assign constants to variables from A to turn the output of the function to be equal to both $x_i\wedge x_j$ and $x_i\oplus x_j$. Blum first applies the standard gate elimination procedure to variables from X using a carefully chosen induction hypothesis that states a circuit size lower bound in terms of the number of variables from X that are still "alive": if there is a substitution $x_i\leftarrow f$ that eliminates at least three gates, perform this substitution and proceed inductively. Note that the used function allows to substitute variables from X by arbitrary functions, but at the same time one is allowed to substitute variables from X, but not from A. In the remaining case, Blum counts the number of gates of out-degree at

least 2: he shows that due to the special properties of the function, any circuit computing it must contain many such gates. This gives a lower bound on the size of a circuit.

In 2011, Demenkov and Kulikov [7] presented a different proof of essentially the same $3n - o(n)$ lower bound for a different function. The function they use is an affine disperser for dimension $d = o(n)$, which is by definition non-constant on any affine subspace of dimension at least d. This property allows to make at least $n - o(n)$ affine substitutions (that is, substitutions of the form $x_i \leftarrow \bigoplus_{j \in J} x_j \oplus c$ where $i \notin J \subseteq [n]$ and $c \in \{0, 1\}$) before the function trivializes. The proof also uses a non-standard circuit complexity measure: for a circuit C, $\mu(C) = \texttt{gates}(C) + \texttt{inputs}(C)$. This trick is used to amortize the case when by substituting one variable one also removes the dependence on another variable. One shows that for any circuit there is a substitution that reduces μ by at least 4 (or makes the whole circuit a constant). This implies, by induction, that for any circuit C computing an affine disperser for dimension $o(n)$,

$$\texttt{gates}(C) + \texttt{inputs}(C) \geq 4(n - o(n)), \tag{1}$$

which in turn implies that $\texttt{gates}(C) \geq 3n - o(n)$. To find an appropriate affine substitution, one considers the topologically first gate A that computes a non-linear binary operation. If A is fed by two variables x_i and x_j of out-degree 1, we substitute $x_i \leftarrow c$ to make A constant. This eliminates A and its successor from the circuit as well as the dependence on both x_i and x_j. Hence both `gates` and `inputs` are reduced by at least 2, and μ is reduced by at least 4. If, say, x_i has out-degree at least 2, we just substitute x_i by the constant that makes A constant: this eliminates the gates fed by x_i and all successors of A (at least three gates in total) and the dependence on x_i, hence μ is reduced by 4 again. In the remaining case, one of the inputs to A is a gate computing an affine function $\bigoplus_{j \in J} x_j \oplus c$. We make it constant by substituting $x_i \leftarrow \bigoplus_{j \in J \setminus \{i\}} x_j \oplus c'$. This eliminates this gate, the gate A, and the successors of A. Thus, μ is reduced by at least 4 again.

Find et al. [11] pushed the lower bound $3n - o(n)$ for affine dispersers further to $(3 + \frac{1}{86})n - o(n)$ by using several new tricks. They generalize the computational model to allow cycles in circuits, use quadratic substitutions (that are turned into affine substitutions in the end of the gate elimination process), and use a carefully chosen circuit complexity measure which besides the number of gates and inputs also depends on the number of certain local bottleneck configurations and the number of quadratic substitutions.

The first explicit construction of an affine disperser for sublinear dimension ($d = o(n)$) was presented relatively recently by Ben-Sasson and Kopparty [4]. While such constructions of higher degree dispersers for sublinear dimension are not yet known, these dispersers do exist, and a lower bound of $3.1n$ has been shown for them in [13] using the circuit complexity measure $\mu_\alpha(C) = \texttt{gates}(C) + \alpha \cdot \texttt{inputs}(C)$ ($\alpha > 0$ is a constant) and quadratic substitutions.

We summarize the discussed lower bounds proofs in the table below.

Bound	Class of functions	Measure	Substitutions
$2n$ [29]	$Q^n_{2,3}$	`gates`	$x_i \leftarrow c$
$2.5n$ [31]	symmetric	`gates`	$x_i \leftarrow c$, $\{x_i \leftarrow f, x_j \leftarrow f \oplus 1\}$
$3n$ [5]	artificial	`gates`	arbitrary: $x_i \leftarrow f$
$3n$ [7]	affine dispersers	$\texttt{gates} + \texttt{inputs}$	linear: $x_i \leftarrow \bigoplus_{j \in J} x_j \oplus c$
$3.01n$ [11]	affine dispersers	$\texttt{gates} + \alpha\texttt{inputs} + \cdots$	quadratic: $x_i \leftarrow f$, $\deg \leq 2$
$3.1n$ [13]	quadratic dispersers	$\texttt{gates} + \alpha\texttt{inputs}$	quadratic: $x_i \leftarrow f$, $\deg \leq 2$

It is also interesting to note that there is a trivial limitation for the first three proofs in the table above: the corresponding classes of functions contain functions of linear circuit

complexity. The class $Q^n_{2,3}$ contains the function THR^n_2 (that outputs 1 iff the sum of n input bits is at least 2) of circuit size $2n + o(n)$. The class of symmetric functions used by Stockmeyer contains the function MOD^n_4 whose circuit size is at most $2.5n + \Theta(1)$. The circuit size of Blum's function is upper bounded by $6n + o(n)$. At the same time it is not known whether there are affine dispersers of sublinear dimension that can be computed by linear size circuits.

3 Limits of Gate Elimination

3.1 Notation

Let $X = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A *substitution* ρ of a set of variables $R \subseteq X$ is a set of $|R|$ restrictions of the form

$$r_i = f_i(x_1, \ldots, x_n),$$

one restriction for each variable $r_i \in R$, where f_i depends only on variables from $X \setminus R$. The degree of a substitution is the maximum degree of f_i's represented as Boolean polynomials. The size of a substitution is $|R|$. Substitutions of size m are called m-substitutions.

Given an m-substitution ρ and a function f, one can naturally define a new function $f|_\rho$ that has m fewer arguments than f.

A function f *depends* on a variable x if there is a substitution ρ of constants to all other variables such that $f|_\rho(0) \neq f|_\rho(1)$.

As we saw in Section 2, gate elimination proofs sometimes track sophisticated *complexity measure* μ rather than just number of gates, for example, the measure $\mu(f) = \mathtt{gates}(f) + \alpha \cdot \mathtt{inputs}(f)$ for a constant α.

A gate elimination argument uses a certain nonnegative complexity measure μ, a family of substitutions $\mathcal{S}$, a family of functions $\mathcal{F}$, a function $\mathtt{gain}\colon \mathbb{N} \to \mathbb{R}$, and a certain predicate $\mathtt{stop}$, and includes proofs of the following statements:

1. (Measure usefulness.) If $\mu(f)$ is large, then $\mathtt{gates}(f)$ is large.
2. (Invariance.) For every $f \in \mathcal{F}$ and $\rho \in \mathcal{S}$, either $f|_\rho \in \mathcal{F}$ or $\mathtt{stop}(f|_\rho)$.
3. (Induction step.) For every $f \in \mathcal{F}$ with $\mathtt{inputs}(f) = n$, there is a substitution $\rho \in \mathcal{S}$ such that $\mu(f|_\rho) \leq \mu(f) - \mathtt{gain}(n)$. (In known proofs, $\mathtt{gain}(n)$ is constant.)

The family must contain functions f such that $\mathtt{stop}(f|_{\rho_1,\ldots,\rho_s})$ is not reached for sufficiently many substitutions from $\mathcal{S}$ (for example, for $s = 0.999 \cdot \mathtt{inputs}(f)$ substitutions).

In what follows, we prove that every gate elimination argument fails to prove a strong lower bound, for many functions of (virtually) arbitrarily large complexity.

3.2 Introductory Example

We start by providing an elementary construction of functions that are resistant with respect to any constant number of arbitrary substitutions, i.e., such substitutions eliminate only a constant number of gates. In the next sections, we generalize this construction to capture other complexity measures.

Consider a function $f\colon \{0,1\}^n \to \{0,1\}$ and let $f \diamond \mathrm{MAJ}_3$ be a function of $3n$ variables resulting from f by replacing each of its input variables x_i by the majority function of three fresh variables x_{i1}, x_{i2}, x_{i3}:

$$(f \diamond \mathrm{MAJ}_3)(x_{11}, x_{12}, \ldots, x_{n3}) = f(\mathrm{MAJ}_3(x_{11}, x_{12}, x_{13}), \ldots, \mathrm{MAJ}_3(x_{n1}, x_{n2}, x_{n3}))\,,$$

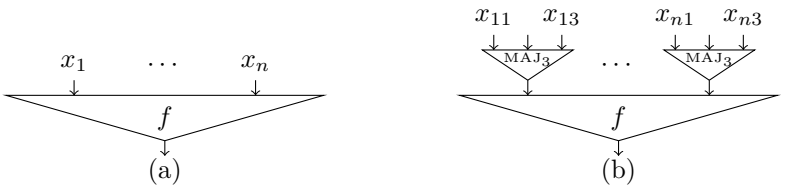

Figure 1 (a) A circuit for f. (b) A circuit for $f \diamond \mathrm{MAJ}_3$.

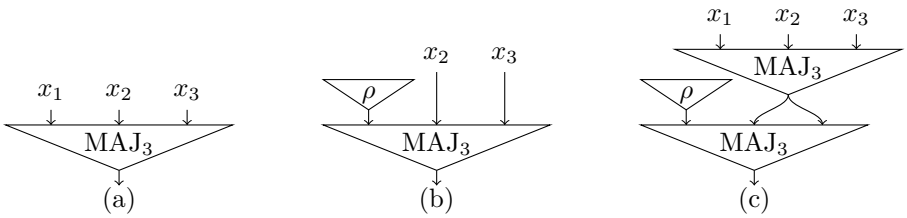

Figure 2 (a) A circuit computing the majority of three bits x_1, x_2, x_3. (b) A circuit resulting from substitution $x_1 \leftarrow \rho$. (c) By adding another gadget to a circuit with x_1 substituted, we force it to compute the majority of x_1, x_2, x_3.

see Fig. 1. Consider a circuit C of the smallest size computing $f \diamond \mathrm{MAJ}_3$. We claim that no substitution $x_{ij} \leftarrow \rho$, where ρ is any function of all the remaining variables, can remove from C more than 5 gates: $\mathtt{gates}(C) - \mathtt{gates}(C|_{x_{ij} \leftarrow \rho}) \leq 5$. We are going to prove this by showing that one can attach a gadget of size 5 to the circuit $C|_{x_{ij} \leftarrow \rho}$ and obtain a circuit that computes f. This is explained in Fig. 2. Formally, assume, without loss of generality, that the substituted variable is x_{11}. We then take a circuit C' computing $f|_{x_{11} \leftarrow \rho}$ and use the value of a gadget computing $\mathrm{MAJ}_3(x_{11}, x_{12}, x_{13})$ instead of x_{12} and x_{13}. This way we suppress the effect of the substitution $x_{11} \leftarrow \rho$, and the resulting circuit C'' computes the initial function $f \diamond \mathrm{MAJ}_3$. Since the majority of three bits can be computed in five gates, we get:

$$\mathtt{gates}(C) \leq \mathtt{gates}(C'') \leq \mathtt{gates}(C|_{x_{11} \leftarrow \rho}) + 5\,.$$

This trick can be extended from 1-substitution to m-substitutions in a natural way. For this, we use gadgets computing the majority of $2m+1$ bits instead of just three bits. We can then suppress the effect of substituting any m variables by feeding the values to $m+1$ of the remaining variables. Taking into account the fact that the majority of $2m+1$ bits can be computed by a circuit of size $4.5(2m+1)$ [8], we get the following result.

▶ **Lemma 1.** *For any $m > 0$, for any function h of n inputs, there exists a function $f = h \diamond \mathrm{MAJ}_{2m+1}$ of $n(2m+1)$ variables, such that*

- *Circuit complexity of f is close to that of h: $\mathtt{gates}(h) \leq \mathtt{gates}(f) \leq \mathtt{gates}(h) + 4.5(2m+1)n$,*
- *For any m-substitution ρ, $\mathtt{gates}(f) - \mathtt{gates}(f|_\rho) \leq 4.5(2m+1)m$.*

▶ Remark. Note that from the Circuit Hierarchy Theorem (see, e.g., [18]), one can find h of virtually any circuit complexity from n to $2^n/n$.

3.3 Subadditive Measures

In this section we generalize the result of Lemma 1 to *arbitrary* subadditive measures. A function $\mu\colon B_n \to \mathbf{R}$ is called a *subadditive complexity measure*, if for all functions f and g, $\mu(h) \le \mu(f) + \mu(g)$, where $h(\bar{x}, \bar{y}) = f(g(\bar{x}), \dots, g(\bar{x}), \bar{y})$. That is, if h can be computed by application some function g to some of the the inputs, and then evaluating f, then the measure of h must not exceed the sum of measures of f and g. Clearly, the measures $\mu(f) = \texttt{gates}(f)$ and $\mu_\alpha(f) = \texttt{gates}(f) + \alpha \cdot \texttt{inputs}(f)$ are subadditive, and so are many other natural measures.

Let $f \in B_n$ and $g \in B_k$. Then by $h = f \diamond g$ we denote the function of nk variables resulting from f by replacing each of its input variables by h applied to k fresh variables.

Our main construction is such a composition of a function f (typically, of large circuit complexity) and a gadget g that is chosen to satisfy certain combinatorial properties. Note that since we show a limitation of the proof method rather than a proof of a lower bound, we do not necessarily need to present explicit functions.

In this section we use gadgets that satisfy the following requirement: For every set of variables Y of size m, we can force the value of the gadget to be 0 and 1 by assigning constants only to the remaining variables.

▶ **Definition 2** (weakly m-stable function)**.** A function $g(X)$ is weakly m-stable if, for every $Y \subseteq X$ of size $|Y| \le m$, there exist two assignments $\tau_0, \tau_1\colon X \setminus Y \to \{0,1\}$ to the remaining variables, such that $g|_{\tau_0}(Y) \equiv 0$ and $g|_{\tau_1}(Y) \equiv 1$. That is, after the assignment τ_0 (τ_1), the function does not depend on the remaining variables Y.

It is easy to see that MAJ_{2m+1} is a weakly m-stable function. In Lemma 6 we show that almost all Boolean functions satisfy an even stronger requirement of stability.

▶ **Theorem 3.** *Let μ be a subadditive measure, f be a Boolean function, g be a weakly m-stable function, and $h = f \diamond g$. Then for every m-substitution ρ, $\mu(h) - \mu(h|_\rho) \le m \cdot \mu(g)$.*

Proof. Similarly to Lemma 1, we use a circuit H for the function $h|_\rho$ to construct a circuit C for h. Let

$$h(x_{11}, x_{12}, \dots, x_{nk}) = f(g(x_{11}, \dots, x_{1k}), \dots, g(x_{n1}, \dots, x_{nk})).$$

Let us focus on the variables $x_{11}, \dots, x_{1k}$. Assume, without loss of generality, that the variables $x_{11}, \dots, x_{1r}$ are substituted by ρ. Since ρ is an m-substitution, $r \le m$. From the definition of weakly m-stable function, there exist substitutions τ_0 and τ_1 to the variables $x_{1r+1}, \dots, x_{1k}$, such that $g|_{\rho\tau_0} = 0$ and $g|_{\rho\tau_1} = 1$. We take the circuit H and add a circuit computing $g(x_{11}, \dots, x_{1k})$. Now, for every variable $x \in \{x_{1r+1}, \dots, x_{1k}\}$ in the circuit H, we wire $g(x_{11}, \dots, x_{1k}) \oplus \tau_0(x)$ instead of x if $\tau_0(x) \ne \tau_1(x)$, and wire $\tau_0(x)$ otherwise. That is, we set $x_{1r+1}, \dots, x_{1k}$ in such a way that $g|_\rho(x_{1r+1}, \dots, x_{1k}) = b = g(x_{11}, \dots, x_{1k})$. Thus, we added one instance of a circuit computing the gadget g and "repaired" $g(x_{11}, \dots, x_{1k})$.

Now we repeat this procedure for each of the n inner functions g that have at least one variable substituted by ρ. Since ρ is an m-substitution, there are at most m gadgets we need to repair. Thus, we can compute h using the circuit H and m instances of a circuit computing g. From subadditivity of μ, $\mu(h) - \mu(h|_\rho) \le m \cdot \mu(g)$. ◀

3.4 Measures that count inputs

The results of the previous section prove that no subadditive complexity measure can prove a lower bound of more than $n\mu(g)$, where the gadget g depends only on m. For $g = \mathrm{MAJ}_{2m+1}$

and measure $\mu(g) = \texttt{gates}(g)$ Lemma 1 gives $4.5(2m+1)n$ as a specific linear bound barrier that gate elimination cannot overcome. However, since $\mu(g)$ depends on the measure μ, it does not exclude a possibility that there is a sequence of complexity measures allowing to prove better and better bounds. One such natural sequence is based on the circuit measure $\mu_\alpha(C) = \texttt{gates}(C) + \alpha \cdot \texttt{inputs}(C)$ for a constant $\alpha \geq 0$ (used, for example, in [7, 13]). Indeed, for growing α, the method of the previous section gives growing bounds, and if one proves that it is possible to eliminate, say, $c_1 > 0$ gates and $c_2 > 1$ variables per substitution, then after $n - o(n)$ substitutions that would give us $\mu(C) \geq (n - o(n))(c_1 + \alpha c_2) = n(c_1 + \alpha c_2) - o(n)$. This would imply that $\texttt{gates}(C) \geq n(c_1 + \alpha(c_2 - 1)) - o(n)$, an arbitrary linear lower bound. Note that does not require a sequence of gate elimination proofs, just a single proof and a sequence of complexity measures.

In this section in order to show that such a measure cannot prove growing linear bounds, we construct a function f such that any m-substitution reduces the measure by a constant number c_m of gates and at most m inputs. This prevents anyone from proving a better than $c_m n$ bound with it.

▶ **Definition 4** (m-stable function)**.** A function $g(X)$ is m-stable if, for every $Y \subseteq X$ of size $|Y| \leq m+1$ and every $y \in Y$, there exists an assignment $\tau \colon X \setminus Y \to \{0,1\}$ to the remaining variables such that $g|_\tau(Y) \equiv y$ or $g|_\tau(Y) \equiv \neg y$. That is, after the assignment τ, the function depends only on the variable y.

It is now easy to see that every m-stable function is a weakly m-stable function.

▶ **Theorem 5.** *Let f be a Boolean function, g be an m-stable function, and $h = f \diamond g$. Then for every m-substitution ρ, $\mu_\alpha(h) - \mu_\alpha(h|_\rho) \leq m \cdot (\texttt{gates}(g) + \alpha)$.*

Proof. Since g is m-stable, Theorem 3 implies that $\texttt{gates}(h) - \texttt{gates}(h|_\rho) \leq m \cdot \texttt{gates}(g)$. It remains to show that $\texttt{inputs}(h) - \texttt{inputs}(h|_\rho) \leq m$. Thus, it suffices to prove that if f depends on x_i and ρ does not substitute $x_{i,j}$, then $h|_\rho$ depends on $x_{i,j}$. Let

$$h(x_{11}, x_{12}, \ldots, x_{nk}) = f(g(x_{11}, \ldots, x_{1k}), \ldots, g(x_{n1}, \ldots, x_{nk})).$$

Assume f depends on its first input. Since g is not constant, there exists a substitution η to the variables $\{x_{21}, \ldots, x_{2k}, \ldots, x_{n1}, \ldots, x_{nk}\}$ such that $h|_\eta(x_{11}, \ldots, x_{1k})$ is not constant.

Let us consider the variables $x_{11}, \ldots, x_{1k}$. Assume, without loss of generality, that the variables $x_{11}, \ldots, x_{1r}$ are substituted by ρ. Since ρ is an m-substitution, $r \leq m$. Now we want to show that for every $j > r$, $h|_\rho$ depends on x_{1j}. From the definition of an m-stable function, there exists a substitution τ to $\{x_{1,r+1}, \ldots, x_{1k}\} \setminus \{x_{ij}\}$ such that $g|_{\rho\tau}(x_{1j})$ is not constant ($g|_{\rho\tau} = x_{1j}$ or $g|_{\rho\tau} = \neg x_{1j}$). Now, we compose the substitutions η and τ, which gives us that $h|_{\rho\tau\eta}(x_{1j})$ is not constant. This implies that the function $h|_\rho$ depends on the variable x_{1j}. ◀

Now we show that for a fixed m, almost all Boolean functions are m-stable.

▶ **Lemma 6.** *For $m \geq 1$ and $k = \Omega(2^m)$, a random $f \in B_k$ is m-stable almost surely.*

Proof. Let X denote the set of k input variables. Let us fix a set Y, $|Y| \leq m+1$, and a variable $y \in Y$. Now let us count the number of functions that do not satisfy the definition of m-stable function for this fixed choice of Y and y. Thus, for each assignment to the variables from $X \setminus Y$, the function must not be y nor $\neg y$. There are 2^{k-m-1} assignments to the variables $X \setminus Y$, and at most $(2^{2^{m+1}} - 2)$ functions of $(m+1)$ variables that are not y nor $\neg y$. Thus, there are at most $(2^{2^{m+1}} - 2)^{2^{k-m-1}}$ functions that do not satisfy the definition of

m-stable function for this fixed choice of Y and y. Now, since there are $\binom{k}{m+1} \cdot (m+1)$ ways to choose Y and y, the union bound implies that a random function is not m-stable with probability at most

$$\frac{\binom{k}{m+1}(m+1)(2^{2^{m+1}}-2)^{2^{k-m-1}}}{2^{2^k}} \le k^{m+2} \cdot \left(\frac{2^{2^{m+1}}-2}{2^{2^{m+1}}}\right)^{2^{k-m-1}} \le$$

$$\exp\left((m+2)\ln k - 2^{k-m-2^{m+1}}\right) = o(1)$$

for $k = \Omega(2^m)$. ◀

Lemma 6, together with Theorem 5, provides a class of functions such that any m-substitution decreases the measure μ_α by at most a fixed constant (which may depend on m but not on α).

▶ **Corollary 7.** *For any $m > 0$, there exists $k > 0$ and a function g of k inputs, such that for any function h of n inputs, the function $f = h \diamond g$ of nk inputs satisfies:*

- *Circuit complexity of f is close to that of h: $\texttt{gates}(h) \le \texttt{gates}(f) \le \texttt{gates}(h) + \texttt{gates}(g) \cdot n$,*
- *For any m-substitution ρ and real $\alpha > 0$, $\mu_\alpha(f) - \mu_\alpha(f|_\rho) \le \texttt{gates}(g) \cdot m + \alpha m$.*

Thus, for many functions gate elimination with m-substitutions and μ_α measures can prove only $O(n)$ lower bounds.

▶ Remark. Although Lemma 6 proves the existence of m-stable functions, their circuit complexities might be large (though constant). To optimize these constants, one can use explicit constructions of m-stable functions. For example, for $m = 1$ one can use an error correcting code $C\colon \{1,\dots,7\} \to \{0,1\}^8$ with distance 4. Let us define a function $g_C\colon \{0,1\}^8 \to \{0,1\}$ as follows:

1. $g_C(C(i)) = 0$ and $g_C(C(i)^{\oplus i}) = 0$ for all i, where $x^{\oplus i}$ inverts the i-th coordinate of the vector x;
2. $g_C(C(i)^{\oplus j}) = 1$ and $g_C(C(i)^{\oplus i,j}) = 1$ for all $j \ne i$.

It is easy to see that g_C is 1-stable. This construction can also be easily generalized to larger m.

A computer-assisted search gives a 1-stable function of 5 inputs that can be computed with 11 gates, which means that for 1-substitutions one cannot prove a lower bound stronger than $11n$.

4 Known Limitations for Various Circuit Models

Although there is no known argument limiting the power of gate elimination, there are many known barriers in proving circuit lower bounds. In this section we list some of them. This list does not pretend to cover all known barriers in proving lower bounds, but we try to show both fundamental barriers in proving strong bounds and limits of specific techniques.

Baker, Gill, and Solovay [3, 12] present the *relativization* barrier that shows that any solution to the P versus NP question must be non-relativizing. In particular, they show that the classical diagonalization technique is not powerful enough to resolve this question. Aaronson and Wigderson [1] present the *algebrization* barrier that generalizes relativization. For instance, they show that any proof of superlinear circuit lower bound requires non-algebrizing techniques. The *natural proofs* argument by Razborov and Rudich [28] shows that a "natural" proof of a circuit lower bound would contradict the conjecture that strong

one-way functions exist. In particular, this argument shows that the *random restrictions* method [14] is unlikely to prove superpolynomial lower bounds. The natural proofs argument implies the following limitation for the gate elimination method. If subexponentially strong one-way functions exist, then for any large class $\mathcal{P}$ of functions (fraction of elements of $\mathcal{P}$ is greater than $\frac{1}{n}$), for any effective measure (computable in time $2^{O(n)}$) and effective family of substitutions $\mathcal{S}$ (the family of substitutions used by the gate elimination algorithm is enumerable in time $2^{O(n)}$), gate elimination cannot prove lower bounds better than $O(n)$. Note that there are currently no known algorithms computing the measures considered in this paper in time $2^{O(n)}$.

Let $\mathcal{F}$ be a family of Boolean functions of n variables. Let X and Y be disjoint sets of input variables, and $|X| = n$. Then a Boolean function $UF(X, Y)$ is called *universal* for the family $\mathcal{F}$ if for every $f(X) \in \mathcal{F}$, there exists an assignment c of constants to the variables Y, such that $UF(X, c) = f(X)$. For example, it can be shown that the function used by Blum [5] is universal for the family $\mathcal{F} = \{x_i \oplus x_j, x_i \wedge x_j | 1 \leq i, j \leq n\}$. Nigmatullin [23, 24] shows that many known proofs can be stated as lower bounds for universal functions for families of low-complexity functions. At the same time, Valiant [34] proves a linear upper bound on the circuit complexity of universal functions for these simple families.

Vadhan and Williams [33] note that the inequality (1) is tight for the inner product function. This implies that the approach from [7] described in Section 2 cannot yield stronger bounds.

There are known linear upper bounds on circuit complexity of some specific functions and even classes of functions. For example, Demenkov et al. [6] show that each *symmetric function* (i.e., a function that depends only on the sum of its inputs over the integers) can be computed by a circuit of size $4.5n + o(n)$. This, in turn, implies that no gate elimination argument for a class of functions that contains a symmetric function can lead to a superlinear lower bound.

The basis U_2 is the basis of all binary Boolean functions without parity and its negation. The strongest known lower bound for circuits over the basis U_2 is $5n - o(n)$. This bound is proved by Iwama and Morizumi [17] for $(n - o(n))$-*mixed* functions. Amano and Tarui [2] construct an $(n - o(n))$-mixed function whose circuit complexity over U_2 is $5n + o(n)$.

A formula is a circuit where each gate has out-degree one. The best known lower bound of $n^{2-o(1)}$ on formula size is proved by Nechiporuk [21]. The proof of Nechiporuk is based on counting different *subfunctions* of given function. It is known that this argument cannot lead to a superquadratic lower bound (see, e.g., Section 6.5 in [18]).

A De Morgan formula is a formula with AND and OR gates, whose inputs are variables and their negations. The best known lower bound for De Morgan formulas is $n^{3-o(1)}$ (Håstad [15], Tal [32], Dinur and Meir [9]). The original proof of this lower bound by Håstad is based on showing that the shrinkage exponent Γ is at least 2. This cannot be improved since Γ is also at most 2 as can be shown by analyzing the formula size of the parity function.

Paterson introduces the notion of formal complexity measures for proving De Morgan formula size lower bounds (see, e.g., [35]). A formal complexity measure is a function $\mu\colon B_n \to \mathbb{R}$ that maps Boolean functions to reals, such that

1. for every literal x, $\mu(x) \leq 1$;
2. for all Boolean functions f and g, $\mu(f \wedge g) \leq \mu(f) + \mu(g)$ and $\mu(f \vee g) \leq \mu(f) + \mu(g)$.

It is known that De Morgan formula size is the largest formal complexity measure. Thus, in order to prove a lower bound on the size of De Morgan formula, it suffices to define a formal complexity measure and show that an explicit function has high value of measure. Khrapchenko [19] uses this approach to prove an $n^{2-o(1)}$ lower bound on

the size of DeMorgan formulas for parity. Unfortunately, many natural classes of formal complexity measures cannot lead to stronger lower bounds. Hrubes et al. [16] prove that *convex* measures (including the measure used by Khrapchenko) cannot lead to superquadratic bounds. A formula complexity measure μ is called *submodular*, if for all functions f, g it satisfies $\mu(f \vee g) + \mu(f \wedge g) \leq \mu(f) + \mu(g)$. Razborov [26] uses a submodular measure based on matrix parameters to prove superpolynomial lower bounds on the size of monotone formulas. In a subsequent work, Razborov [27] shows that submodular measures cannot yield superlinear lower bounds for non-monotone formulas. The *drag-along principle* [28, 20] shows that no useful formal complexity measure can capture specific properties of a function. Namely, it shows that if a function has measure m, then a random function with probability $1/4$ has measure at least $m/4$. Measures based on graph entropy (Newman and Wigderson [22]) are used to prove a lower bound of $n \log n$ on DeMorgan formula size, but it is proved that these measures cannot lead to stronger bounds.

5 Conclusion and Further Directions

In this paper we have demonstrated that there are functions of virtually arbitrary complexity that even after several substitutions do not allow to reduce their complexity more than by a constant number of gates (and at most one variable they depend upon), or a constant amount of a subadditive complexity measure.

This puts a barrier on gate elimination proofs that do not use specific properties of the functions while analyzing how their circuits degrade after substitutions. Indeed, in most proofs it is usually the case (properties of the function are used for estimating *how many substitutions* can the function withstand).

However, there is one exception: in order to estimate the number of "bad" local situations on the top of a circuit computing the function, [11] uses the fact that the function is an affine disperser. While we believe that in this particular case it can be overcome, there may be new techniques exploiting the function properties. Thus the first open question is:

- *Show that interesting classes of functions contain functions resistant to gate elimination. For example, it would be interesting to show that the class of affine dispersers, or more generally every large enough class of functions, contains a series of functions resistant to gate elimination.*

Another possible direction is to extend the result to other possible complexity measures, because some syntactic measures can lack subadditivity (for example, composition can in principle introduce more "bad" local situations). One can imagine, for example, "local" measures that count specific small patterns in a circuit.

- *Extend the result to local complexity measures or another wide class.*

While the results of this paper capture all types of substitutions, another possible directions is:

- *Allow induction to descend to arbitrary varieties instead of the varieties described by substitutions (for example, allow restrictions of the form $xy = zt$).*

The situation might become much easier if we switch from arbitrary Boolean functions to n-bit linear maps $\{0,1\}^n \to \{0,1\}^n$. They have non-linear complexity in principle but, again, we do not have non-linear lower bounds for explicit functions. Can gate elimination prove non-linear bounds here? What if we restrict ourselves to linear operations in the circuit and linear substitutions? The gadgets used in this paper are non-linear and thus cannot help.

- *Extend the result to linear maps.*

We show that there exist functions such that after a constant number of substitutions the complexity of these functions decreases only by a constant. How far can it be strengthened w.r.t. the number of substitutions?

- *Does there exist a function f of nonlinear complexity such that after $m = \Omega(n)$ substitutions its circuit complexity drops by $O(m)$ gates only?*

References

1. Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory (TOCT)*, 1(1):2, 2009.
2. Kazuyuki Amano and Jun Tarui. A well-mixed function with circuit complexity $5n \pm o(n)$: Tightness of the Lachish–Raz-type bounds. In *Proceedings of Theory and Applications of Models of Computation*, volume 4978 of *Lecture Notes in Computer Science*, pages 342–350. Springer, 2008.
3. Theodore Baker, John Gill, and Robert Solovay. Relativizations of the $\mathcal{P} =? \mathcal{NP}$ question. *SIAM Journal on computing*, 4(4):431–442, 1975.
4. Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 65–74, 2009.
5. Norbert Blum. A boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.
6. Evgeny Demenkov, Arist Kojevnikov, Alexander S. Kulikov, and Grigory Yaroslavtsev. New upper bounds on the Boolean circuit complexity of symmetric functions. *Information Processing Letters*, 110(7):264–267, 2010.
7. Evgeny Demenkov and Alexander S. Kulikov. An elementary proof of a $3n - o(n)$ lower bound on the circuit complexity of affine dispersers. In *Proceedings of International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 256–265, 2011.
8. Evgeny Demenkov and Alexander S. Kulikov. Computing All MOD-Functions Simultaneously. *Computer Science – Theory and Applications*, pages 81–88, 2012.
9. Irit Dinur and Or Meir. Toward the krw composition conjecture: Cubic formula lower bounds via communication complexity. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 50. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.
10. Paul E. Dunne. *Techniques for the analysis of monotone Boolean networks*. PhD thesis, University of Warwick, 1984.
11. Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. A better-than-$3n$ lower bound for the circuit complexity of an explicit function. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:166, 2015.
12. Lance Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, pages 1–15, 1994.
13. Alexander Golovnev and Alexander S. Kulikov. Weighted gate elimination: Boolean dispersers for quadratic varieties imply improved circuit lower bounds. In *Innovations in Theoretical Computer Science, ITCS '16*, pages 405–411, 2016.
14. Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20. ACM, 1986.
15. Johan Håstad. The shrinkage exponent is 2. In *Foundations of Computer Science, 1993. Proceedings., 34th Annual Symposium on*, pages 114–123. IEEE, 1993.
16. Pavel Hrubeš, Stasys Jukna, Alexander Kulikov, and Pavel Pudlak. On convex complexity measures. *Theoretical Computer Science*, 411(16):1842–1854, 2010.

17 Kazuo Iwama and Hiroki Morizumi. An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits. In *Proceedings of International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 2420 of *Lecture Notes in Computer Science*, pages 353–364. Springer, 2002.

18 Stasys Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

19 Valeriy M. Khrapchenko. Method of determining lower bounds for the complexity of P-schemes. *Mathematical Notes*, 10(1):474–479, 1971.

20 Richard J. Lipton. *The $\mathcal{P} = \mathcal{NP}$ Question and Gödel's Lost Letter*. Springer Science & Business Media, 2010.

21 Edward I. Nechiporuk. On a Boolean function. *Doklady Akademii Nauk. SSSR*, 169(4):765–766, 1966.

22 Ilan Newman and Avi Wigderson. Lower bounds on formula size of boolean functions using hypergraph entropy. *SIAM Journal on Discrete Mathematics*, 8(4):536–542, 1995.

23 Roshal G. Nigmatullin. Are lower bounds on the complexity lower bounds for universal circuits? In *Fundamentals of Computation Theory*, pages 331–340. Springer, 1985.

24 Roshal G. Nigmatullin. *Complexity lower bounds and complexity of universal circuits*. Kazan University, 1990.

25 Wolfgang J. Paul. A $2.5n$-lower bound on the combinational complexity of boolean functions. *SIAM J. Comput.*, 6(3):427–443, 1977.

26 Alexander A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990.

27 Alexander A. Razborov. On submodular complexity measures. In *Poceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 76–83, New York, NY, USA, 1992. Cambridge University Press.

28 Alexander A. Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

29 Claus-Peter Schnorr. Zwei lineare untere schranken für die komplexität boolescher funktionen. *Computing*, 13(2):155–171, 1974.

30 Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

31 Larry J. Stockmeyer. On the combinational complexity of certain symmetric boolean functions. *Mathematical Systems Theory*, 10:323–336, 1977.

32 Avishay Tal. Shrinkage of De Morgan formulae by spectral techniques. In *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*, pages 551–560. IEEE, 2014.

33 Salil Vadhan and Ryan Williams. Personal communication, 2013.

34 Leslie G. Valiant. Universal circuits (preliminary report). In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 196–203. ACM, 1976.

35 Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987.