Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 119 (2016)

On the Limits of Gate Elimination

Alexander Golovnev* Edward A. Hirsch†‡ Alexander Knop†
Alexander S. Kulikov†

May 20, 2018

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 a circuit by making one or several substitutions to the input variables and repeats this inductively. In this paper 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.

*New York University
†St. Petersburg Department of Steklov Institute of Mathematics of the Russian Academy of Sciences
‡St. Petersburg State University

ISSN 1433-8092

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. Some k nodes are additionally labeled as the outputs of the circuit. The output gate compute a Boolean function $\{0,1\}^n \to \{0,1\}^k$ 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 $B_{n,k}$ we denote the set of all Boolean functions $f\colon \{0,1\}^n \to \{0,1\}^k$. B_n is a shorthand for $B_{n,1}$. 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.

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 [Sha49]. 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 $(3 + \frac{1}{86})n - o(n)$ [FGHK16].

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 $\mathrm{MOD}^n_{3,r} \in B_n$ 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 [Sch74]. 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 proven 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 $2n + o(n)$ [Dun84] (see also Theorem 2.3 in [Weg87]). 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 [Weg87]) 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: we design circuits such that no substitution (and even a constant number of substitutions) can reduce the size of the circuit by more than a constant number of gates. This, in turn, implies that one cannot prove a superlinear lower bound this way. For recently popular measures that combine the number of gates with the number of inputs we prove a stronger result: 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 (in all modern proofs $m = 1$ or 2).

The gate elimination method has also been recently used for proving average-case circuit lower bounds and upper bounds for Circuit #SAT [CK15, GKST16]. The limitation result of this work also applies to this line of research, implying that gate elimination cannot lead to strong improvements on the currently known results.

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 [Sto77] 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 [Blu84], following the work by Paul [Pau77], 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 [DK11] 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. [FGHK16] 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 [BK12]. 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 [GK16] using the circuit complexity measure $\mu_\alpha(C) = \texttt{gates}(C) + \alpha \cdot \texttt{inputs}(C)$ ($\alpha \geq 0$ is a constant) and quadratic substitutions.

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

Bound	Class	Measure	Substitutions
$2n$ [Sch74]	$Q^n_{2,3}$	`gates`	$x_i \leftarrow c$
$2.5n$ [Sto77]	symmetric	`gates`	$x_i \leftarrow c$, $\{x_i \leftarrow f, x_j \leftarrow f \oplus 1\}$
$3n$ [Blu84]	artificial	`gates`	arbitrary: $x_i \leftarrow f$
$3n$ [DK11]	affine disp.	$\texttt{gates} + \texttt{inputs}$	linear: $x_i \leftarrow \bigoplus_{j\in J} x_j \oplus c$
$3.01n$ [FGHK16]	affine disp.	$\texttt{gates} + \alpha\texttt{inputs} + \cdots$	linear: $x_i \leftarrow \bigoplus_{j\in J} x_j \oplus c$
$3.1n$ [GK16]	quadratic disp.	$\texttt{gates} + \alpha\texttt{inputs}$	quadratic: $x_i \leftarrow f$, $\deg(f) \leq 2$

It should be noted that most of the known gate elimination proofs can be restated in the following terms. One fixes a circuit complexity measure μ. One then shows that for **any function** f (of sufficiently large complexity) there exists a substitution $x_i \leftarrow \rho$ (or a few consecutive such substitutions) that either reduces $\mu(f)$ by a sufficient amount or trivializes f. One way to show this is to prove that even for **any circuit** C (again, possibly except for circuits of very low complexity) there exists such a substitution. (This is slightly more general because it suffices to prove it for optimal circuits only, but most proofs do it for all circuits.) This implies a lower bound on $\mu(f)$ for a function f that is resistant to sufficiently many substitutions of the form $x_i \leftarrow \rho$. We provide examples below.

1. Schnorr [Sch74] shows that if a circuit contains a variable x_i of out-degree at least 2, then at least two gates are eliminated by a substitution $x_i \leftarrow c$. He then argues that if a circuit contains variables of out-degree 1 only, then it cannot compute a function under consideration as in this case for two variables feeding a top-gate the whole circuit depends on these two variables only through this gate. Another way of using this argument is the following. If there is a top gate that depends on two out-degree 1 variables, then by an appropriate substitution of the form $x_i \leftarrow c$ or $x_i \leftarrow x_j \oplus c$, one kills this gate as

well as removes dependence on two variables (both x_i and x_j), rather than just one. Hence these two cases can be combined into one as follows: for any circuit C, there exists a substitution of the form $x_i \leftarrow c$ or $x_i \leftarrow x_j \oplus c$ that reduces $\texttt{gates}(C) + \texttt{inputs}(C)$ by at least 3 (or trivializes C). This implies a lower bound $2n - c$ for many symmetric functions.

2. Stockmeyer [Sto77] shows that for any circuit C, there exists either a substitution $x_i \leftarrow c$ reducing $\texttt{gates}(C) + \texttt{inputs}(C)$ by at least 4 or two substitutions of the form $x_i \leftarrow \rho, x_j \leftarrow \rho \oplus 1$ reducing $\texttt{gates}(C) + \texttt{inputs}(C)$ by at least 7. This implies a lower bound $2.5n - c$ for many symmetric functions.

3. Demenkov and Kulikov [DK11] show that for any circuit C there exists an affine substitution reducing $\texttt{gates}(C) + \texttt{inputs}(C)$ by at least 4. This gives a lower bound $3n - o(n)$ for affine dispersers of dimension $o(n)$.

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 $f|_\rho(0) \neq f|_\rho(1)$. Additionaly, we denote by $\texttt{inputs}(f)$ the number of variables such that f depends on them.

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 $\mu(f) = \texttt{gates}(f) + \alpha \cdot \texttt{inputs}(f)$ for a constant $\alpha \geq 0$.

A gate elimination argument uses a certain nonnegative complexity measure μ, a family of substitutions $\mathcal{S}$, a constant $\texttt{gain}$, and includes proofs of the following statements:

1. (Measure usefulness.) If $\mu(f)$ is large, then $\texttt{gates}(f)$ is large.

2. (Induction step.) For every function f with $\texttt{inputs}(f) = n$, there is a substitution $\rho \in \mathcal{S}$ such that

   - either $f|_\rho$ becomes constant
   - or $\mu(f|_\rho) \leq \mu(f) - \texttt{gain}$.

To get a circuit lower bound from such a gate elimination argument, one presents a sequence $\mathcal{F} = \{f_i\}_{i=1}^{\infty}, f_i \in B_i$, of explicit functions such that each f_i is not trivialized by sufficiently many substitutions from $\mathcal{S}$ (for example, by $0.999 \cdot \texttt{inputs}(f_i)$ substitutions). This roughly means that one can make at least

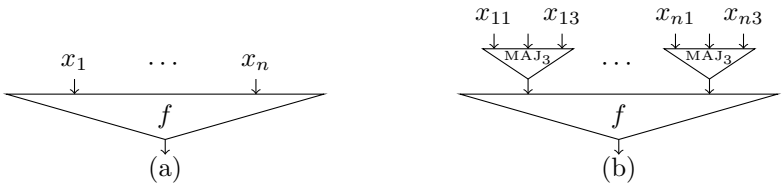

Figure 1: (a) A circuit for f. (b) A circuit for $f \diamond \mathrm{MAJ}_3$.

$0.999 \cdot \mathtt{inputs}(f)$ substitutions from $\mathcal{S}$ each time reducing the measure by $\mathtt{gain}$. This, in turn, gives a lower bound on the initial complexity of $\mathcal{F} = \{f_i\}_{i=1}^{\infty}$.

We note that for proving lower bounds, it would suffice to satisfy the induction step only for functions from $\mathcal{F}$ and functions obtained from them by substitutions, rather than for all functions. Since usually one does not know the structure of optimal circuits for specific function classes, one proves this step for all functions. To the best of our knowledge, the only known exception (which uses the properties of functions under consideration in the induction step) is the work of Blum [Blu84]. See examples in the end of Section 2.

In what follows, we prove that every gate elimination argument fails to prove a strong lower bound. We prove this by constructing functions such that no substitution (or a set of substitutions) can significantly reduce their complexity measures.

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 $f \in B_n$ and let $f \diamond \mathrm{MAJ}_3 \in B_{3n}$ be a function 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}:

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

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 $4.5(2m+1)$ [DKMM12], we get the following result.

Lemma 1. *For any $h \in B_n$ and any $m > 0$, the function $f = h \diamond \mathrm{MAJ}_{2m+1} \in B_{n(2m+1)}$ satisfies the following two properties:*

- *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, $\mathtt{gates}(f) - \mathtt{gates}(f|_\rho) \leq 4.5(2m+1)m$.*

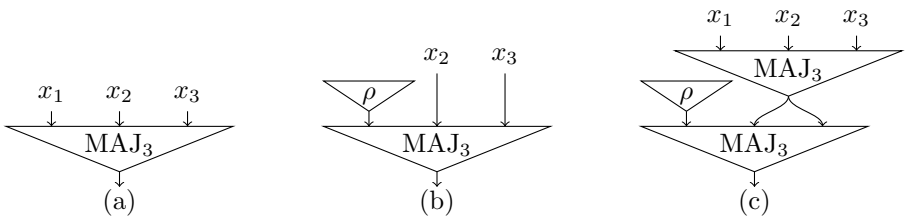

Figure 2: (a) A circuit computing the majority of three bits x_1, x_2, x_3. (b) A circuit resulting from substitution $x_1 \leftarrow \rho$. (c) By adding another gadget to a circuit with x_1 substituted, we force it to compute the majority of x_1, x_2, x_3.

Remark 1. *Note that from the Circuit Hierarchy Theorem (see, e.g., [Juk12]), 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 $\mu\colon \bigcup_{n,m\in\mathbb{N}} B_{n,m} \to \mathbb{R}$ is called a *subadditive complexity measure*, if

- for all functions $\bar{s}$ and $\bar{t}$, $\mu((\bar{s},\bar{t})) \le \mu(\bar{s}) + \mu(\bar{t})$, where $(\bar{s},\bar{t})$ is the "merge" of $\bar{s}$ and $\bar{t}$ that comprises (independent) inputs and outputs of both functions,
- for all functions f and $\bar{g}$, $\mu(h) \le \mu(f) + \mu(\bar{g})$, where $h(\bar{x},\bar{y}) = f(\bar{g}(\bar{x}),\bar{y})$.

That is, if h can be computed by application of some (multi-output) function $\bar{g}$ to a part of the inputs, and then evaluating f, then the measure of h must not exceed the sum of measures of f and $\bar{g}$. 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 $h = f \diamond g \in B_{nk}$ we denote the function resulting from f by replacing each of its input variables by g 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 1 (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 2 we show that almost all Boolean functions satisfy an even stronger requirement of stability.

Theorem 1. *Let μ be a subadditive measure, $f \in B_n$ be any function, $g \in B_k$ be a weakly m-stable function, and $h = f \diamond g \in B_{nk}$. 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

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

Let us focus on 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, $r \leq m$. From the definition of weakly m-stable function, there exist substitutions τ_0 and τ_1 to the variables $x_{1r+1}, \ldots, x_{1k}$ such that $g|_{\rho\tau_0} = 0$ and $g|_{\rho\tau_1} = 1$. We take the circuit H and add a circuit computing $g(x_{11}, \ldots, x_{1k})$. Now, for every variable $x \in \{x_{1r+1}, \ldots, x_{1k}\}$ in the circuit H, we wire $g(x_{11}, \ldots, x_{1k}) \oplus \tau_0(x)$ instead of x if $\tau_0(x) \neq \tau_1(x)$, and wire $\tau_0(x)$ otherwise; i.e. we set the value of x equal to $\tau_0(x)$ if $g(x_{11}, \ldots, x_{1k}) = 0$ and equal to $\tau_1(x)$ otherwise. That is, we set x_{1r+1}, $\ldots$, x_{1k} in such a way that $g|_\rho(x_{1r+1}, \ldots, x_{1k}) = g(x_{11}, \ldots, x_{1k})$. Thus, we added one instance of a circuit computing the gadget g and "repaired" $g(x_{11}, \ldots, x_{1k})$.

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, in particular, h is a composition of the function $h|_\rho$ computed by H and the multi-output function $\overline{g^{(m)}}$ comprising m instances of g. By subadditivity of μ, $\mu(h) - \mu(h|_\rho) \leq m \cdot \mu(g)$. □

This theorem gives tight (up to a multiplicative constant) bounds for constant m. Indeed, an m-substitution can always eliminate at least m gates, and Theorem 1 gives a constant upper bound on the number of eliminated gates, too. For $m = O(\sqrt{nk})$ this theorem also yelds a nontrivial bound. For example, if we take $m = n$, $k = 2n + 1$ and an n-stable function $g = \mathrm{MAJ}_{2n+1}$, then we have:

Corollary 1. *There is a constant $c > 0$ such that if $f \in B_n$, and $h = f \diamond \mathrm{MAJ}_{2n+1}$ ($h \in B_N$ where $N = 2n^2 + n$). Then for every n-substitution ρ, $\mathtt{gates}(h) - \mathtt{gates}(h|_\rho) \leq cN$.*

Using similar constructions and error correcting codes we can extend this corollary to larger substitutions (this was suggested to us by A. Shen). A binary error-correcting code with relative distance δ, codeword length N and message length n is a linear function $S : \{0,1\}^n \to \{0,1\}^N$ such that for any $x \neq y \in \{0,1\}^n$, $\frac{\Delta(S(x),S(y))}{N} \geq \delta$.

We say that algorithm (circuit) E encodes a binary error-correcting code $S : \{0,1\}^n \to \{0,1\}^N$ iff $E(x) = S(x)$, for every $x \in \{0,1\}^n$.

We also say that algorithm (circuit) D decodes a binary error-correcting code $S : \{0,1\}^n \to \{0,1\}^N$ and corrects ϵN errors iff for any $y \in \{0,1\}^N$ and x such that $\frac{\Delta(S(x),y)}{N} \leq \epsilon$, $D(y) = x$.

Results of [GI05] and a classical transformation from Turing machines to circuits shows that the following theorem is true.

Theorem 2 ([GI05])**.** *For any $n \in \mathbb{N}$ and $\epsilon > 0$, there are a binary error-correcting code S with relative distance $1 - \epsilon$, codeword length $N = O(n)$ and message length n, and circuits D and E such that D decodes S and corrects $\frac{1-\epsilon}{2}N$ errors, E encodes S, the size of D is $O(n \log n)$, and the size of E is $O(n)$.*

Theorem 3. *Let S be a binary error-correcting code with relative distance 2ϵ, codeword length N and message length n. Let $D \in B_{N,n}$ be a Boolean circuit of size $d(N)$ decoding S correcting ϵN errors, and $E \in B_{n,N}$ be a Boolean circuit of size $e(n)$ encoding S. Let $f \in B_n$ and $h = f \circ D$ ($h \in B_N$) be a composition of f and D. Then*

1. *$\mathtt{gates}(h) \geq \mathtt{gates}(f) - e(n)$,*
2. *for every ϵN-substitution ρ, $\mathtt{gates}(h) - \mathtt{gates}(h|_\rho) \leq e(n) + d(N)$.*

Proof. Let us prove that $\mathtt{gates}(h) \geq \mathtt{gates}(f) - e(n)$. Let C be a circuit computing h. Let us consider the composition of two circuits $C'(x) = C(E(x))$. Note that C' computes f and the size of C' equals $\mathtt{gates}(h) + e(n)$.

Now let us prove that for every ϵN-substitution ρ, $\mathtt{gates}(h) - \mathtt{gates}(h|_\rho) \leq e(n) + d(N)$. Let C be a circuit computing $h|_\rho$. Let us consider the circuit $C''(x) = C(E(D(x)))$ (to abuse the notation, we assume that C just ignores the substituted inputs). Note that C'' computes h and the size of C'' equals $\mathtt{gates}(h|_\rho) + e(n) + d(N)$. □

Corollary 2. *For any $\epsilon > 0$, there is a function $g_n \in B_{N,n}$ (where N depends on n) such that for any Boolean function $f \in B_n$ and $h = f \circ g_n$ it holds that*

1. $\texttt{gates}(h) \geq \texttt{gates}(f) - O(n)$,
2. *for every $(\frac{1}{2} - \epsilon) \cdot N$-substitution ρ, $\texttt{gates}(h) - \texttt{gates}(h|_\rho) \leq O(N \log(N))$.*

Proof. The corollary follows from Theorem 3 and Theorem 2. □

Remark 2. *To complement the result from Corollary 2, we note that any function $h \in B_N$ can be trivialized by $\lceil N/2 \rceil$ substitutions.*

Proof. Let $n = \lfloor N/2 \rfloor$ and $m = \lceil N/2 \rceil$. Let us first show that either

- for every $x \in \{0,1\}^n$, there exists a $y = y(x) \in \{0,1\}^m$ s.t. $h(x,y) = 0$, or
- for every $y \in \{0,1\}^m$, there exists an $x = x(y) \in \{0,1\}^n$ s.t. $h(x,y) = 1$.

Assume for the sake of contradiction that neither of the previous statements holds. Then there exists an x_1 s.t. for all y, $h(x_1, y) = 1$. Similarly, there exists a y_1 s.t. for all x, $h(x, y_1) = 0$. Taking $y = y_1$ and $x = x_1$ we get a contradiction: $1 = h(x_1, y_1) = 0$.

Assume that the first statement holds (the other case is analogous): for every $x \in \{0,1\}^n$, there exists a $y = y(x) \in \{0,1\}^m$ s.t. $h(x,y) = 0$. Let ρ be the following m-substitution: $(v_{n+1}, \ldots, v_N) = y(v_1, \ldots, v_n)$. By the definition of $y(x)$, we have $h|_\rho(v_1, \ldots, v_N) = 0$. □

A decision tree is a simple and natural model for computing Boolean functions. It is a full binary tree (each node has either zero or two children). The computation starts at the root and at every internal node it queries the value of an input variable x_i. The two outgoing edges from the node are labeled by two possible values for x_i. This way the computation reaches a leaf and the label of this leaf (0 or 1) is defined to be the value of the function computed by this decision tree. See [Juk12, Chapter 14] for an overview of decision trees.

It is not difficult to see that the minimum size of a decision tree computing parity of n input bits is $2^{\Theta(n)}$: every path from the root to a leaf has to query all input bits. At the same time, if we allow at each internal node to query not just whether x_i is equal to $c \in \{0,1\}$, but whether x_i is equal $\bigoplus_{j \in J} x_j \oplus c$, then the parity can be computed by a decision tree with just two leaves. It is natural to ask what happens in the most general case where at each internal node one is allowed to ask whether x_i is equal to ρ where ρ is an *arbitrary* function. It turns out that Corollary 2 implies that even in this case there exist functions of exponential decision tree size.

Corollary 3. *There exists a function $f \in B_n$ such that any decision tree of f has size at least $2^{\Omega(n)}$ even if branchings $x \leftarrow \rho$ and $x \leftarrow \rho \oplus 1$ (where ρ is an arbitrary function) are allowed.*

Proof. Use any function of exponential complexity and apply Corollary 2. □

3.4 Measures that count inputs

The number of gates is not the only circuit complexity measure that is used in gate elimination proofs. In some bottleneck cases, it is not possible to find a substitution killing many gates, but it is still possible to make a substitution that reduces some other complexity parameter of a circuit significantly. One such parameter is the number of inputs of a circuit. In [DK11] it is used as follows. Assume that two variables x and y feed an $\wedge$-gate. If one of them (say, x) has out-degree at least 2, one easily eliminates at least three gates: assign $x \leftarrow 0$, this kills all successors of x (at least two gates) and also makes the $\wedge$-gate constant, so its successors are also eliminated (at least one gate). If, on the other hand, both x and y have out-degree 1, it is not clear how to eliminate more than two gates by assigning x or y. One notes, however, that the substitution $x \leftarrow 0$ eliminates not only two gates, but also two inputs: x is assigned, while y is just not needed anymore as the only gate that is fed by y turns into a constant under $x \leftarrow 0$. If one deals with

a function that is resistant w.r.t. any $n-k$ substitutions (and usually $k=o(n)$), then the situation like the one above (when by assigning one variable one makes a circuit independent of some other variable, too) can only appear k times. Indeed, if k such substitutions can be made, then the circuit (and hence the function) trivializes after $k+(n-2k)=n-k$ substitutions (contradicting the fact that it is stable to any $n-k$ substitutions). Usually we have $k=o(n)$ which implies that this situation happens only $o(n)$ times. A convenient way of exploiting this fact is to incorporate the number of inputs into the circuit complexity measure. Namely, [DK11] uses the following measure: $\mu(C)=\texttt{gates}(C)+\texttt{inputs}(C)$. Then, to prove a lower bound $\texttt{gates}(C)\geq 3n-o(n)$ it is enough to prove that $\mu(C)\geq 4n-o(n)$. For this, in turn, one shows that it is always possible to find a substitution that reduces μ by at least 4. For the two cases discussed above it is easy: in the former case, we remove three gates and one input (hence, μ is reduced by 4), in the latter one, we remove two gates and two inputs (μ is reduced by 4 again). In [GKST16, GK16] a more general measure is used: $\mu_\alpha(C)=\texttt{gates}(C)+\alpha\cdot\texttt{inputs}(C)$, where $\alpha\geq 0$ is a constant. A typical m-substitution reduces μ_α by $k+\alpha m$ where k is the number of gates eliminated. If, however, a substitution removes more than m inputs, then μ_α is reduced by at least $\alpha(m+1)$. By choosing a large enough value for α, one ensures that $\alpha(m+1)\geq k+\alpha m$.

For example, in Lemma 1 we show that there are circuits where no substitution can eliminate too many gates. But this claim does not exclude the following possibility: Assume that for some circuits we can eliminate $\log n$ gates, and for the remaining circuits we cannot eliminate even 2 gates, but we can eliminate 2 inputs. Then by setting $\alpha\approx\log n$ and considering the measure $\mu_\alpha(C)=\texttt{gates}(C)+\alpha\cdot\texttt{inputs}(C)$ one would prove a superlinear lower bound.

In this section, we construct gadgets against such measures. Namely, we construct a function f such that any m-substitution reduces the number of gates by a constant c_m and reduces the number of inputs by m. This prevents anyone from proving a better than $c_m n$ bound using these measures.

Definition 2 (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 weakly m-stable.

Theorem 4. *Let f be a Boolean function, g be an m-stable function, and $h=f\diamond g$. Then for every $\alpha\geq 0$, 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 1 implies that $\texttt{gates}(h)-\texttt{gates}(h|_\rho)\leq m\cdot\texttt{gates}(g)$. It remains to show that $\texttt{inputs}(h)-\texttt{inputs}(h|_\rho)=m$. Thus, it suffices to prove that if h depends on x_{ij} and ρ does not substitute $x_{i,j}$, then $h|_\rho$ depends on $x_{i,j}$. Let

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

Without loss of generality let $i=1$. Let R be the set of variables x_{st} for $s>1$ substituted by ρ. There exists a substitution η to the variables $\{x_{21},\ldots,x_{2k},\ldots,x_{n1},\ldots,x_{nk}\}\setminus R$ such that $h|_\eta(x_{11},\ldots,x_{1k})$ does not depend on the variables in R and is not a constant: by the definition of m-stability we can force the instances of the gadget g except for the first one to produce any desired assignment for the inputs of f (all but the first one).

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 $\{x_{1,r+1},\ldots,x_{1k}\}\setminus\{x_{1j}\}$ 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 2. *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, $|Y| \le m+1$, 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 2, together with Theorem 4, 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 4. *For any $m > 0$, there exist $k > 0$ and $g \in B_k$ such that for any f of n inputs, the function $h = f \diamond g \in B_{nk}$ satisfies:*

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

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

Although Lemma 2 proves the existence of m-stable functions, their circuit complexities may be large (though constant). To optimize these constants, one can use explicit constructions of m-stable functions.

Lemma 3. *For any m, there exists an m-stable function of $n = \Theta(m)$ inputs of circuit complexity at most $O(m^2 \log m)$.*

Proof. Let n be a power of two, and let $C\colon \{1,\dots,n\} \to \{0,1\}^n$ be the Walsh–Hadamard error correcting code (a code with distance $\frac{n}{2}$, see, e.g., [AB09, Section 19.2.2]). We define a function $g_C\colon \{0,1\}^n \to \{0,1\}$ in the following way. Given an input x, we first find the nearest codeword $C(i)$ to x (any of them in the case of a tie), and then output the ith bit of the input: $g_C(x) = x_i$. It is easy to see that g_C can be computed in randomized linear time $O(n)$, thus, it can be computed by a circuit of size $O(n^2 \log n)$ (see, e.g., [Adl78]).

Let us show that g_C is $\left(\frac{n}{4} - 2\right)$-stable. To this end we show that for any set $Y \subseteq \{x_1,\dots,x_n\}$, $|Y| \le \left(\frac{n}{4} - 1\right)$, for any $y \in Y$, there exists an assignment to the remaining variables that forces g_C to compute y. Without loss of generality, assume that $Y = \{x_1,\dots,x_{n/4-1}\}$ and that $y = x_1$. Let us fix the last $3n/4+1$ bits to be equal to the corresponding bits of $C(1)$. Namely, we set $(x_{n/4},\dots,x_n) = (C(1)_{n/4},\dots,C(1)_n)$. After these substitutions, any input x has distance less than $n/4$ to the codeword $C(1)$, thus $C(1)$ is the nearest codeword. This implies that $g_C(x)$ always outputs $y = x_1$. □

Corollary 5. *For any $m > 0$, there exists a function g of $k = O(m)$ inputs such that for any function h of n inputs, the function $h = f \diamond g$ of nk inputs satisfies:*

- *Circuit complexity of h is close to that of f: $\mathtt{gates}(f) \le \mathtt{gates}(h) \le \mathtt{gates}(f) + O(m^2 n \log m)$,*
- *For any m-substitution ρ and real $\alpha \ge 0$, $\mu_\alpha(h) - \mu_\alpha(h|_\rho) \le O(m^3 \log m) + \alpha m$.*

A computer-assisted search gives a 1-stable function of 5 inputs that can be computed with 11 gates.

Lemma 4. *There exists a 1-stable function $g_1^{\mathrm{st}}\colon \{0,1\}^5 \to \{0,1\}$ of circuit complexity at most 11.*

Proof. The truth table of the function g_1^{st} is shown below.

x_0	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
x_1	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
x_2	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
x_3	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
x_4	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$g_1^{\mathrm{st}}(x_0, x_1, x_2, x_3, x_4)$	0 0 0 0 0 0 1 1 0 1 0 1 1 1 1 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 0 0

It can be checked that for any $i, j \in \{0, 1, 2, 3, 4\}$, where $i \neq j$, there exist $c_1, c_2, c_3 \in \{0, 1\}$ such that when the three remaining variables are assigned the values c_1, c_2, c_3, the function g_1^{st} turns into x_i. For example, under the substitution $\{x_0 \leftarrow 0, x_2 \leftarrow 0, x_4 \leftarrow 1\}$ the function g_1^{st} is equal to x_1. Appendix on page 13 contains two short scripts which (i) verify the required property of the function g_1^{st}, and (ii) demonstrate that g_1^{st} can be computed by a circuit of size 11. □

This lemma implies that the basic gate elimination argument is unable to prove a lower bound of $11n$ using the measure μ_α and 1-substitutions. (Note that almost all known proofs use either 1- or 2-substitutions.)

Corollary 6. *For any function f of n inputs, assume function $h = f \diamond g_1^{\mathrm{st}}$ (it has $5n$ inputs). Then*

1. *The complexity of h is close to that of f: $\mathtt{gates}(f) \leq \mathtt{gates}(h) \leq \mathtt{gates}(f) + 11n$;*
2. *For any 1-substitution ρ and real $\alpha \geq 0$, $\mu_\alpha(h) - \mu_\alpha(h|_\rho) \leq 11 + \alpha$.*

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 [BGS75, For94] 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 [AW09] 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 [RR94] 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 [Hås86] 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 of predicates (i.e., a class containing at least a $\frac{1}{n}$ fraction of B_n), for any effective measure (computable in time $2^{O(n)}$) and effective family of substitutions $\mathcal{S}$ (i.e., a family of substitutions enumerable in time $2^{O(n)}$), gate elimination using the family $\mathcal{S}$ of substitutions cannot prove lower bounds better than $O(n)$. We note that the measures considered in this work are not known to be effective.

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 [Blu84] is universal for the family $\mathcal{F} = \{x_i \oplus x_j, x_i \wedge x_j | 1 \leq i, j \leq n\}$. Nigmatullin [Nig85, Nig90] 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 [Val76] proves a linear upper bound on the circuit complexity of universal functions for these simple families.

Vadhan and Williams [VW13] note that the inequality (1) is tight for the inner product function. This implies that the approach from [DK11] 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. [DKKY10] 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 proven by Iwama and Morizumi [IM02] for $(n - o(n))$-*mixed* functions. Amano and Tarui [AT11] 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 proven by Nechiporuk [Nec66]. The proof of Nechiporuk is based on counting different *subfunctions* of the given function. It is known that this argument cannot lead to a superquadratic lower bound (see, e.g., Section 6.5 in [Juk12]).

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 [Hås98], Tal [Tal14], Dinur and Meir [DM16]). 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., [Weg87]). 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 [Khr71] uses this approach to prove an $\Omega(n^2)$ lower bound on the size of De Morgan formulas for parity. Unfortunately, many natural classes of formal complexity measures cannot lead to stronger lower bounds. Hrubes et al. [HJKP10] 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 and g it satisfies $\mu(f \vee g) + \mu(f \wedge g) \leq \mu(f) + \mu(g)$. Razborov [Raz90] uses a submodular measure based on matrix parameters to prove superpolynomial lower bounds on the size of monotone formulas. In a subsequent work, Razborov [Raz92] shows that submodular measures cannot yield superlinear lower bounds for non-monotone formulas. The *drag-along principle* [RR94, Lip10] 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 [NW95]) are used to prove a lower bound of $n \log n$ on De Morgan formula size, but it is proven that these measures cannot lead to stronger bounds.

5 Conclusion and Further Directions

In this paper we show that there are functions of virtually arbitrary complexity that even after several substitutions their circuit complexity reduces only by a constant number of gates 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. Most proofs indeed do not use functions properties for the analysis (properties of the function are only used for estimating *how many substitutions* the function can 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, [FGHK16] 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 other large classes.*

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

• *Allow the 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 results to linear maps.*

Corollary 2 implies that for any $m = O(\sqrt{N})$, there exist a hard function of N arguments whose circuit complexity after m substitutions drops by a linear number $O(N)$ of gates only. On the other hand, it is easy to see that any function f can be trivialized by $N/2$ substitutions (And this bound is tight, since MAJ_N does not become constant until $N/2$ variables are substituted.) These two observations leave a gap between $O(\sqrt{N})$ and $N/2$, which leads us to the following open question:

• *Does there exist a nonlinear complexity function f of N arguments such that after any $m = \omega(\sqrt{N})$ substitutions its circuit complexity drops by $O(N)$ gates only?*

To construct such a function, it suffices to construct error correcting codes with linear size encoders/decoders.

Finally, there is an informal connection between lower bounds for algorithms (such as splitting (DPLL) algorithms) and functions resistant to gate elimination (cf, e.g., Corollary 3), and algorithms for CircuitSAT and #CircuitSAT are of this type. It would be interesting to formalize this connection.

• *Use functions resistant to gate elimination as candidates for lower bounds on the running time. Convert them into formulas that can be used as candidates for lower bounds for proof systems.*

Acknowledgements

We are grateful to Navid Talebanfard for fruitful discussions, to Fedor Petrov for pointing out the existence of a trivializing $N/2$-substitution for every function, and to Alexander Shen for suggesting to use error-correcting codes in Theorem 3. We are very thankful to the anonymous reviewers for their valuable comments.

The research leading to these results was conducted in the laboratory of algorithmic methods founded under the grant 14.Z50.31.0030 of the Government of the Russian Federation.

A Scripts for the proof of Lemma 4

First, we show that for any $i, j \in \{0,1,2,3,4\}$, where $i \neq j$, there exist $c_1, c_2, c_3 \in \{0,1\}$ such that when the three remaining variables are assigned the values c_1, c_2, c_3, the function g_1^{st} computes x_i. The following `Python` script specifies the values of c_1, c_2, c_3 for all (ordered) pairs (i, j) and ensures that the function g_1^{st} satisfies the required property.

```
import itertools

tt=[0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,0,1,0,0,0]

M = {(0,1): (0,1,0), (0,2): (0,0,1), (0,3): (0,0,1), (0,4): (0,0,1),
     (1,0): (1,0,0), (1,2): (0,0,1), (1,3): (0,0,1), (1,4): (0,1,0),
     (2,0): (1,0,0), (2,1): (0,1,0), (2,3): (0,1,0), (2,4): (0,0,1),
     (3,0): (0,1,0), (3,1): (1,0,0), (3,2): (1,0,0), (3,4): (0,0,1),
     (4,0): (1,0,0), (4,1): (1,0,0), (4,2): (1,0,0), (4,3): (0,1,0)}

for (i,j) in itertools.permutations(range(5), 2):
    c1=M[(i,j)][0]
    c2=M[(i,j)][1]
    c3=M[(i,j)][2]

    (v1,v2,v3)=[v for v in range(5) if (v!=i and v!=j)]

    for a in range(1 << 5):
        assert((a>>v1)&1!=c1 or
               (a>>v2)&1!=c2 or
               (a>>v3)&1!=c3 or
               tt[a]==(a>>i)&1
               )
```

The fact that this function can be computed by a circuit of size 11 is justified by the following script.

```
import itertools

tt=[0,0,0,0,0,0,1,1,0,1,0,1,1,1,1,0,0,1,1,1,0,1,1,0,0,1,1,0,1,0,0,0]

for (x0,x1,x2,x3,x4) in itertools.product(range(2), repeat=5):
    g1=(x1+x2)%2       # x1 xor x2
    g2=(x2+x3)%2       # x2 xor x3
    g3=(x0+x2)%2       # x0 xor x2
    g4=(x1+x4)%2       # x1 xor x4
    g5=1-(1-g2)*(1-g4) # g2 or g4
    g6=1-(1-x3)*(1-x4) # x3 or x4
    g7=x3*g4           # x3 and g4
    g8=g1*(1-g7)       # g1 and (not g7)
    g9=(1-g3)*g6       # (not g3) and g6
    g10=g5*(1-g9)      # g5 and (not g9)
    g11=(g8+g10)%2     # g8 xor g10

    assert(g11==tt[x0+2*x1+4*x2+8*x3+16*x4])
```

References

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.

[Adl78] Leonard M. Adleman. Two theorems on random polynomial time. In *19th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 75–83. IEEE, 1978.

[AT11] Kazuyuki Amano and Jun Tarui. A well-mixed function with circuit complexity 5n: Tightness of the lachish-raz-type bounds. *Theor. Comput. Sci.*, 412(18):1646–1651, 2011.

[AW09] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. *TOCT*, 1(1):2:1–2:54, 2009.

[BGS75] Theodore P. Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM J. Comput.*, 4(4):431–442, 1975.

[BK12] Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012.

[Blu84] Norbert Blum. A boolean function requiring $3n$ network size. *Theor. Comput. Sci.*, 28:337–345, 1984.

[CK15] Ruiwen Chen and Valentine Kabanets. Correlation bounds and #SAT algorithms for small linear-size circuits. In *Computing and Combinatorics - 21st International Conference (COCOON)*, pages 211–222, 2015.

[DK11] 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.

[DKKY10] Evgeny Demenkov, Arist Kojevnikov, Alexander S. Kulikov, and Grigory Yaroslavtsev. New upper bounds on the boolean circuit complexity of symmetric functions. *Inf. Process. Lett.*, 110(7):264–267, 2010.

[DKMM12] Evgeny Demenkov, Alexander S. Kulikov, Ivan Mihajlin, and Hiroki Morizumi. Computing all MOD-functions simultaneously. In *7th International Computer Science Symposium in Russia (CSR)*, volume 7353, pages 81–88. Springer, 2012.

[DM16] Irit Dinur and Or Meir. Toward the KRW composition conjecture: Cubic formula lower bounds via communication complexity. In *31st Conference on Computational Complexity (CCC)*, pages 3:1–3:51, 2016.

[Dun84] Paul E. Dunne. *Techniques for the analysis of monotone Boolean networks.* PhD thesis, University of Warwick, Coventry, UK, 1984.

[FGHK16] M. G. Find, A. Golovnev, E. A. Hirsch, and A. S. Kulikov. A better-than-$3n$ lower bound for the circuit complexity of an explicit function. In *57th Symposium on Foundations of Computer Science (FOCS)*, pages 89–98. IEEE, 2016.

[For94] Lance Fortnow. The role of relativization in complexity theory. *Bulletin of the EATCS*, 52:229–243, 1994.

[GI05] Venkatesan Guruswami and Piotr Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. Information Theory*, 51(10):3393–3400, 2005.

[GK16] 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)*, pages 405–411, 2016.

[GKST16] Alexander Golovnev, Alexander S. Kulikov, Alexander V. Smal, and Suguru Tamaki. Circuit size lower bounds and #SAT upper bounds through a general framework. In *41st International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 45:1–45:16, 2016.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 6–20, 1986.

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

[HJKP10] Pavel Hrubes, Stasys Jukna, Alexander S. Kulikov, and Pavel Pudlák. On convex complexity measures. *Theor. Comput. Sci.*, 411(16-18):1842–1854, 2010.

[IM02] Kazuo Iwama and Hiroki Morizumi. An explicit lower bound of 5n - o(n) for boolean circuits. In *Mathematical Foundations of Computer Science (MFCS)*, pages 353–364, 2002.

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

[Khr71] Valeriy M. Khrapchenko. Method of determining lower bounds for the complexity of P-schemes. *Mathematical Notes*, 10(1):474–479, 1971.

[Lip10] Richard J. Lipton. *The $\mathcal{P} = \mathcal{NP}$ Question and Gödel's Lost Letter*. Springer Science & Business Media, 2010.

[Nec66] Edward I. Nechiporuk. On a Boolean function. *Doklady Akademii Nauk. SSSR*, 169(4):765–766, 1966.

[Nig85] R. G. Nigmatullin. Are lower bounds on the complexity lower bounds for universal circuits. In *Fundamentals of Computation Theory, (FCT)*, pages 331–340, 1985.

[Nig90] Roshal G. Nigmatullin. *Complexity lower bounds and complexity of universal circuits*. Kazan University, 1990.

[NW95] Ilan Newman and Avi Wigderson. Lower bounds on formula size of boolean functions using hypergraph entropy. *SIAM J. Discrete Math.*, 8(4):536–542, 1995.

[Pau77] Wolfgang J. Paul. A $2.5n$-lower bound on the combinational complexity of boolean functions. *SIAM J. Comput.*, 6(3):427–443, 1977.

[Raz90] Alexander A. Razborov. Applications of matrix methods to the theory of lower bounds in computational complexity. *Combinatorica*, 10(1):81–93, 1990.

[Raz92] Alexander A. Razborov. On submodular complexity measures. In *Poceedings of the London Mathematical Society Symposium on Boolean Function Complexity*, pages 76–83. Cambridge University Press, 1992.

[RR94] Alexander A. Razborov and Steven Rudich. Natural proofs. In *26th Annual ACM Symposium on Theory of Computing (STOC)*, pages 204–213. ACM, 1994.

[Sch74] Claus-Peter Schnorr. Zwei lineare untere schranken für die komplexität boolescher funktionen. *Computing*, 13(2):155–171, 1974.

[Sha49] Claude E. Shannon. The synthesis of two-terminal switching circuits. *Bell Systems Technical Journal*, 28:59–98, 1949.

[Sto77] Larry J. Stockmeyer. On the combinational complexity of certain symmetric boolean functions. *Mathematical Systems Theory*, 10:323–336, 1977.

[Tal14] Avishay Tal. Shrinkage of de morgan formulae by spectral techniques. In *55th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 551–560, 2014.

[Val76] Leslie G. Valiant. Universal circuits (preliminary report). In *8th Annual ACM Symposium on Theory of Computing, (STOC)*, pages 196–203, 1976.

[VW13] Salil Vadhan and Ryan Williams. Personal communication, 2013.

[Weg87] Ingo Wegener. *The complexity of Boolean functions*. Wiley-Teubner, 1987.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il