Constant Depth Formula and Partial Function Versions of MCSP are Hard

Rahul Ilango
MIT
Cambridge, MA, USA
rilango@mit.edu

***Abstract*—Attempts to prove the intractability of the Minimum Circuit Size Problem (MCSP) date as far back as the 1950s and are well-motivated by connections to cryptography, learning theory, and average-case complexity. In this work, we make progress, on two fronts, towards showing MCSP is intractable under worst-case assumptions.**

While Masek showed in the late 1970s that the version of MCSP for DNF formulas is NP-hard, extending this result to the case of depth-3 AND/OR formulas was open. We show that determining the minimum size of a depth-d formula computing a given Boolean function is NP-hard under quasipolynomial-time randomized reductions for all constant $d \geq 2$. Our approach is based on a method to "lift" depth-d formula lower bounds to depth-$(d+1)$. This method also implies the existence of a function with a $2^{\Omega_d(n^{1/5})}$ additive gap between its depth-d and depth-$(d+1)$ formula complexity.

We also make progress in the case of general, unrestricted circuits. We show that the version of MCSP where the input is a partial function (represented by a string in $\{0,1,?\}^*$) is not in P under the Exponential Time Hypothesis (ETH).

Intriguingly, we formulate a notion of lower bound statements being (P/poly)-recognizable that is closely related to Razborov and Rudich's definition of being (P/poly)-constructive. We show that unless there are subexponential-sized circuits computing SAT, the lower bound statements used to prove the correctness of our reductions *cannot* be (P/poly)-recognizable.

***Keywords*-Minimum Circuit Size Problem; MCSP; NP-hardness; Constant Depth Formulas; Natural Proofs Barrier**

I. Organization

This extended abstract is organized as follows.

- Section II: We discuss general background on MCSP and then proceed to describe more specific motivation and prior work related to our results.
- Section III: We state our results and some of the broader takeaways from our theorems.
- Section IV: We describe the main ideas and intuition behind our results.
- Section V: We conclude by talking about a connection between designing reductions to MCSP and (a variant of) the constructivity condition in the Natural Proofs barrier.

II. Background and Motivation

A. General Background

The Minimum Circuit Size Problem, abbreviated MCSP, requires one to determine whether a given Boolean function $f : \{0,1\}^n \to \{0,1\}$ (represented by its truth table, a binary string of length $N = 2^n$) is computable by circuits of size at most a given parameter $s \in \mathbb{N}$. Throughout this work, we adopt this n and N notation, where n is the number of inputs to f and N is the length of the truth table of f.

Kabanets and Cai [1] initiated the "modern" study of MCSP and recent work has uncovered deep connections between MCSP and a growing number of areas including cryptography, learning theory, pseudorandomness and average-case complexity.

Giving an exhaustive review of these results is beyond our scope. However, we informally state some highlights and recommend an excellent survey by Allender [2] for a detailed overview.

- If MCSP is NP-hard under polynomial time many-one reductions, then EXP $\neq$ ZPP [3].
- If MCSP with a fixed size parameter $s = \text{poly}(n)$ does not have circuits of size $\tilde{O}(N)$, then NP $\not\subseteq$ P/poly [4].
- If MCSP $\in$ P, then there are no one-way functions [1], [5].
- If a certain "universality conjecture" is true, then the existence of one-way functions is equivalent to zero-error average-case hardness of MCSP (under a certain setting of parameters) [6].
- There is an equivalence between learning a circuit class $\mathcal{C}$ and the problem of "approximately minimizing" $\mathcal{C}$-circuits [7].
- If a certain approximation to MCSP is NP-hard, then there is a "worst-case to average-case" reduction for NP [8].

Moreover, all but one of these results have been proved within the past five years!

B. Specific Background and Motivation

While it is easy to see that MCSP is in NP, it is a longstanding open question whether MCSP is NP-hard. Indeed, there is work dating back to the 1950s attempting to establish the intractability of MCSP (see [9] for a history of

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00047

this early work), and Levin is said[1] to have initially delayed publishing his results on the theory of NP-completeness in hopes of also showing MCSP is NP-complete. Nearly a half-century later, the question of whether MCSP is NP-complete remains wide open.

One intuition for why it is difficult to prove hardness for MCSP is that producing a NO instance of MCSP corresponds to producing a function with a certain circuit complexity lower bound, a notoriously difficult task even when the desired lower bound is quite small. Kabanets and Cai formalized this intuition to show that any "natural" polynomial-time reduction from SAT to MCSP would imply breakthrough circuit lower bounds [1].

We describe two potential ways researchers hope to "sidestep" having to prove strong lower bounds while still giving compelling evidence that MCSP is intractable. The first is to strengthen the assumption under which we are trying to show that MCSP is intractable. Roughly speaking, the Kabanets and Cai result suggests that proving $\mathsf{MCSP} \notin \mathsf{P}$ under the assumption that $\mathsf{P} \neq \mathsf{NP}$ likely requires breakthrough circuit lower bounds.

However, it is not clear whether a similar barrier exists to proving that, say, the Exponential Time Hypothesis (ETH) implies that $\mathsf{MCSP} \notin \mathsf{P}$. In particular, we certainly know of functions that require circuits of size cn for small constants c, and even brute-forcing over all circuits of size n requires about $n!$ time, which is superpolynomial in $N = 2^n$. Thus, it is conceivable that one could prove that $\mathsf{MCSP} \notin \mathsf{P}$ under ETH by showing that the brute-force algorithm for MCSP is nearly optimal when $s = O(n)$, since this is a regime where we already have lower bounds. Indeed, we view this as a tantalizing possibility.

Another approach to sidestep having to prove breakthrough circuit lower bounds is to consider the circuit minimization task for restricted classes of circuits $\mathcal{C}$ that we already have strong lower bounds against, like AC^0. To formalize this, let $\mathcal{C}$ be some class of circuits, and let $(\mathcal{C})$-MCSP be the task of determining whether a given truth table is computed by some $\mathcal{C}$-circuit of size at most a given parameter.

Despite our relatively good understanding of circuit classes like AC^0, progress on proving hardness for $(\mathcal{C})$-MCSP has been somewhat elusive. In 1979, Masek showed that (DNF)-MCSP is NP-hard. A series of subsequent results [11]–[15] simplified Masek's proof and showed near-optimal hardness of approximation for (DNF)-MCSP. However, it was only recently, in 2018, that hardness was proved for a class $\mathcal{C}$ beyond DNFs: Hirahara, Oliveira, and Santhanam [16] showed that $(\mathcal{C})$-MCSP is NP-hard when $\mathcal{C}$ is the class of $\mathsf{DNF} \circ \mathsf{XOR}$ circuits (that is, DNFs that are allowed to have XOR gates at its leaves).

[1] [10] cites a personal communication from Levin regarding this, and some discussion can be found on Levin's website: https://www.cs.bu.edu/fac/lnd/research/hard.htm.

Before we go on to state our results, we give a quick review of how NP-hardness is proved for (DNF)-MCSP and $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP. In particular, both results are proved using a two part strategy that involves an intermediate problem $(\mathcal{C})$-MCSP* which we define now.[2]

Roughly speaking, $(\mathcal{C})$-MCSP* is the analogue of $(\mathcal{C})$-MCSP for partial truth tables. Formally, $(\mathcal{C})$-MCSP* is defined as follows

- **Given:** the partial truth table $T \in \{0,1,\star\}^{2^n}$ of an n-input partial function $\gamma : \{0,1\}^n \to \{0,1,\star\}$ and a size parameter $s \in \mathbb{N}$
- **Determine:** whether there is a $\mathcal{C}$-circuit of size at most s that computes γ on all its $\{0,1\}$-valued inputs.

We stress that the truth table T here is of length $N = 2^n$ and the function f is not represented by the set of $\{0,1\}$-valued input/output pairs $\{(x, f(x)) : f(x) \in \{0,1\}\}$, which could be exponentially more concise. Indeed, it is known that the input/output pair representation version of MCSP* is NP-complete [17], [18]. However, this result makes use of the succinctness of the input representation, and the instances that the reduction produces can be solved by brute force in time $\mathsf{poly}(N)$.

The two part strategy used to prove hardness for (DNF)-MCSP and $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP is then as follows: First, reduce an NP-hard problem to $(\mathcal{C})$-MCSP*. Second, reduce $(\mathcal{C})$-MCSP* to MCSP*.

Thus, the starting point of this work was to aim to prove hardness for $(\mathcal{C})$-MCSP* and $(\mathcal{C})$-MCSP for as expressive classes of circuits $\mathcal{C}$ as possible.

III. Results and Discussion

A. $(\mathcal{C})$-MCSP *is Hard when* $\mathcal{C}$ *is Constant Depth Formulas*

Our first result shows that $(\mathcal{C})$-MCSP is NP-hard under randomized quasipolynomial time reductions when $\mathcal{C}$ is the class, denoted AC^0_d, of depth-d *formulas* with NOT gates and AND/OR gates of unbounded fan-in.

Theorem 1. *Let $d \geq 2$. Given oracle access to (AC^0_d)-MCSP, one can compute SAT in randomized quasipolynomial time.*

We discuss some of the ideas behind our proof in Section IV. In a few sentences, our reduction works by induction on d. The $d = 2$ case is given by the previously known hardness of (DNF)-MCSP. For the inductive step, our main technical contribution is to prove a novel way to "lift" depth-d lower bounds to depth-$(d+1)$ lower bounds. We use this technique to estimate the depth-d complexity of a function using an oracle that computes the depth-$(d+1)$ complexity of functions.

Comparison to Previous Work. As we mentioned earlier, Masek [19] proved that (DNF)-MCSP is NP-hard in the

[2] Actually, Masek's original reduction was a direct reduction from Circuit-SAT, but later improvements used this framework.

1970s, and Hirahara, Oliveira, and Santhanam [16] recently showed that $(\mathsf{DNF} \circ \mathsf{XOR})$-MCSP is NP-hard.

One way the jump from DNF and $\mathsf{DNF} \circ \mathsf{XOR}$ to AC^0_3 is significant is that both DNF and $\mathsf{DNF} \circ \mathsf{XOR}$ circuits can be written as $\mathsf{OR} \circ \mathcal{D}$ for a circuit class $\mathcal{D}$ that is not functionally complete (i.e., not every function can be computed by a circuit in $\mathcal{D}$). In the case of DNFs and $\mathsf{DNF} \circ \mathsf{XOR}$ circuits, $\mathcal{D}$ contains functions corresponding to subcubes and affine subspaces respectively. On the other hand, AC^0_3 includes the class of $\mathsf{OR} \circ \mathsf{CNF}$ formulas and CNFs are functionally complete. This makes it more involved to prove lower bounds for AC^0_3. For example, it is still a major open question to prove explicit, strongly exponential lower bounds against AC^0_3. This reduced understanding is our rationale for why the depth-3 case was elusive. Indeed, this difference is manifest in our results as our method for "lifting" the existing depth-2 result requires significantly different ideas than the ones in [19] and [16], though their work forms our base case.

Another related work is the innovative paper of Buchfuhrer and Umans [20], who showed that the $\Sigma_2 P$ variant of (AC^0_d)-MCSP is $\Sigma_2 P$-hard. In particular, they consider the problem where given an AC^0_d formula φ and a size parameter s, one must output whether there is a AC^0_d formula of size at most s that computes the same function as φ. As we will describe later in this section, one of the first steps in our reduction is actually the same as in Buchfuhrer and Umans: to show that we can restrict to the case where the final output gate is assumed to be OR.

After this, however, our proof strategy diverges significantly. In a sense, this divergence is expected since the different input representations give the two problems a very different character. One consequence of this difference, as Buchfuhrer and Umans note in their paper, is that while the succinctness of the input representation in the $\Sigma_2 P$ version allows one to get by with clever applications of "weak" lower bounds, the full truth table representation used in MCSP and (AC^0_d)-MCSP means that proving NP-hardness through "the use of weak lower bounds is not even an option, under a complexity assumption."

Finally, perhaps the most direct prior work is by Allender, Hellerstein, McCabe, Pitassi, and Saks [13] who extended the cryptographic hardness results for MCSP to show cryptographic hardness for computing (AC^0_d)-MCSP when d is sufficiently large.

Using randomness to prove hardness for MCSP-type problems. While there is significant evidence that proving MCSP is NP-hard under deterministic reductions is beyond the reach of current techniques [1], [3], no such barriers are known for randomized reductions.

Indeed, some recent results show that for close variants of MCSP, like an oracle variant [21] and a multi-output variant [22], one can prove the problem is NP-hard using randomized reductions.

We view our reduction as a further demonstration of how one can use randomness in proving hardness for MCSP-related problems. Intriguingly, our result seems to use randomness in a more subtle way than the aforementioned results. In particular, while the aforementioned results use randomness to sample uniformly random functions, we use randomness to sample functions with specific properties that uniformly random functions do not have. These properties are crucial to our analysis.

Application: Large Gaps in Complexity Between Depths. A reasonable question is whether our method used in the reduction for "lifting" depth-d lower bounds to depth-$(d+1)$ formula lower bounds can be applied to prove new lower bounds.

Indeed, we give such an application. One can ask how far apart can the depth-d and depth-$(d+1)$ formula complexity of a function be. In our notation, this corresponds to asking how large can one make the quantity $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f)$.

Using existing depth hierarchy theorems for AC^0, there exist explicit functions for which this gap is at least $2^{n^{\Omega(1/d)}}$ [23].

Using our techniques, we are able to improve the dependence on d significantly.

Theorem 2. *For all $d \geq 2$ there exists a function f such that $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f) \geq 2^{\Omega_d(n^{1/5})}$.*

Our proof works by "lifting" the $d = 3$ case in the known depth hierarchy theorems to higher depths at a low cost.

We note, however, that our method comes with some drawbacks. First, the lower bound is existential and does not exhibit an explicit function witnessing this separation. Second, while there is a large additive gap $\mathsf{L}_{d-1}(f)$ and $\mathsf{L}_d(f)$, there is only a constant factor multiplicative gap between the two quantities, and lastly, (related to the previous point) it only gives a gap for formulas and not circuits.

Despite these drawbacks, we find Theorem 2 to be especially interesting because it does not yet seem possible to prove such a result using the usual AC^0 lower bound approaches. An intriguing question is how well this lower bound fits into the Natural Proofs framework of Razborov and Rudich [24]. We defer discussion about this to Section V.

Open Questions. There are several intriguing open questions related to our (AC^0_d)-MCSP result. Can one prove that minimizing constant depth *circuits* is NP-hard? Our proof techniques heavily rely on the underlying model being formulas.

Another interesting question is to prove stronger hardness of approximation for (AC^0_d)-MCSP. Our results only yield hardness for small constant factor approximations. One should be able to do significantly better.

One can also try to look beyond constant depth AND/OR formulas. What if one is allowed to use, say, $\oplus$ gates?

Finally, can one improve the gap between $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f)$ in Theorem 2? Can one give a multiplicative gap instead of an additive one? What about the case of circuits?

B. $(\mathcal{C})$-MCSP* is Hard for General Circuits

As we mentioned earlier, hardness for $(\mathcal{C})$-MCSP* has been an important intermediate step towards proving hardness for $(\mathcal{C})$-MCSP in previous results. This naturally motivates the search for the most expressive class $\mathcal{C}$ where we can show that $(\mathcal{C})$-MCSP* is hard. Perhaps surprisingly, we are able to show hardness even in the case of general circuits, but in order to do this we strengthen our assumption to the Exponential Time Hypothesis (ETH).

ETH was first formulated by Impagliazzo, Paturi, and Zane [25], [26] and has been extremely useful for proving conditional lower bounds on various problems (see [27] for a survey). It is somewhat technical to define ETH formally, but, roughly speaking, it is a slight strengthening of the statement that 3-SAT cannot be solved deterministically in $2^{o(n)}$ time.

To formalize our result, let MCSP* denote the the problem of $(\mathcal{C})$-MCSP* where $\mathcal{C}$ is the class of general circuits: that is circuits with fan-in two AND and OR gates as well as NOT gates where the size of a circuit is the number of AND and OR gates in the circuit. We establish that MCSP* is not in P assuming ETH.

Theorem 3. *Assume ETH holds. Then there is no deterministic algorithm solving* MCSP* *in time* $N^{o(\log \log N)}$. *Moreover, given the truth table of a partial function* $T \in \{0,1,\star\}^N$, *there is no deterministic* $N^{o(\log \log N)}$ *time algorithm for deciding whether* T *can be computed by a monotone read once formula.*

We prove this theorem by giving a reduction from a problem with known ETH hardness ($2n \times 2n$ Bipartite Permutation Independent Set) to MCSP*. Lokshtanov, Marx, and Saurabh [28] showed that, under ETH, $2n \times 2n$ Bipartite Permutation Independent Set cannot be solved in deterministic time $2^{o(n \log n)}$. We discuss the basic idea behind our proof in Section IV.

Input Representation and Closeness of MCSP* **to** MCSP**.** We again stress that the partial function input to MCSP* is represented as a string in $\{0,1,\star\}^{2^n}$ and not as a (possibly exponentially more concise) list of input/output pairs where the partial function is defined. To highlight this difference, we note that while the input/output pair representation variant of MCSP* is already known to be NP-complete under deterministic many-one reductions [17], [18], if the same were known for MCSP*, then the breakthrough separation EXP $\neq$ ZPP would follow from an argument by Murray and Williams [3].

Implications for Read Once Formulas. Theorem 3 establishes that under ETH the brute force algorithm for detecting whether a partial function can be computed by a monotone read once formula is nearly optimal, since there are roughly $N^{\log \log N}$ such read once formulas. This is in sharp contrast to the case when one is given a *total* function f as input: in that case, one can decide if f is computable by a monotone read once formula in time $\mathsf{poly}(n)$ given oracle access to the truth table of the function [29], an exponential gap!

Algorithmic Implications. Currently, the best known algorithm for solving MFSP on a truth table of length N and with a size parameter s is the brute force algorithm that runs in time $Ns2^{O(s \log n)}$. There have been some efforts [30] hoping to reduce the exponential dependence from $s \log n$ to s. Theorem 3 suggests that the exponential $s \log n$ dependence may be necessary when the input is a partial truth table, at least in the regime where $s = O(n)$.

Open Question: Extension to MCSP**?** A natural question is whether this result can be extended to show that MCSP $\notin$ P under ETH. We already know reductions from $(\mathcal{C})$-MCSP* to $(\mathcal{C})$-MCSP for the classes DNF and DNF $\circ$ XOR, so perhaps one can also reduce MCSP* to MCSP.

In our opinion, however, the most promising approach is to skip MCSP* entirely and extend our techniques to apply to MCSP directly. In particular, our MCSP* hardness result can be viewed in a more general framework that we describe now. Let $f : \{0,1\}^n \to \{0,1\}$ be a function whose optimal circuits have size exactly s. Let $F : \{0,1\}^n \times \{0,1\}^k \to \{0,1\}$. We say that F is a *simple extension* of f if

- F depends on all its inputs,
- F can be computed by a circuit of size $s + k$, and
- there exists a $y_0 \in \{0,1\}^k$ such that for all $x \in \{0,1\}^n$ we have $F(x, y_0) = f(x)$.

Essentially, the definition of a simple extension of an optimal f-circuit is made so that we can apply a "reverse gate elimination" argument (we describe what this is in Section IV) to argue that any optimal circuit for F is obtained by taking an optimal circuit for f and "uneliminating" (i.e. adding) gates "in a specific way."

From our definition, it is easy to see that one can compute whether F is a simple extension of f using an oracle to MCSP. Thus, if one can show hardness for deciding whether F is a simple extension of f, then one has established hardness for MCSP.

Indeed, our approach to proving hardness for MCSP* essentially shows that deciding whether a *partial* function F is a simple extension of OR_n (the OR function on n bits) cannot be solved in time $N^{o(\log \log N)}$ under ETH.

We believe that one might be able to prove a similar hardness result for MCSP by letting f be a function other than OR_n. Indeed the difficultly with using $f = \mathsf{OR}_n$ to try to prove hardness for MCSP is that the set of optimal OR_n circuits is so well structured that it is easy to decide whether any total function F is a simple extension of $f = \mathsf{OR}_n$. This difficultly is manifest in any function f whose optimal circuits are read once formulas.

Thus, the missing component in extending our results to MCSP is finding some function f whose optimal circuits we can characterize but are also sufficiently complex. Since we can make do with linear-sized optimal circuits, we see no immediate reason why existing techniques cannot yield such an f.

IV. Proof Ideas

A. Hardness for (AC^0_d)-MCSP.

Before we begin, we introduce some notation. The *size* of a formula φ is denoted by $|\varphi|$ and equals the number of leaves in the binary tree underlying φ. Given a Boolean function f, $\mathsf{L}_d(f)$ denotes the size of the smallest depth-d formula computing f. $\mathsf{L}^{\mathsf{OR}}_d(f)$ and $\mathsf{L}^{\mathsf{AND}}_d(f)$ denote the size of the smallest depth-d formula whose output/top gate is an OR or AND gate respectively.

Three Step Overview. At a high-level, our strategy for proving the NP-hardness of computing $\mathsf{L}_d(\cdot)$ breaks into three parts.

1) Show that for all $d \geq 2$ one can reduce computing $\mathsf{L}^{\mathsf{OR}}_d$ to L_d, so it suffices to prove NP hardness for $\mathsf{L}^{\mathsf{OR}}_d$.
2) Show that when $d = 2$ it is NP-hard to compute $\mathsf{L}^{\mathsf{OR}}_d$ within any constant factor (this part was already known).
3) Show that when $d \geq 3$ one can compute a small approximation of $\mathsf{L}^{\mathsf{OR}}_{d-1}$ using an oracle that computes a small approximation of $\mathsf{L}^{\mathsf{OR}}_d$. Conclude that L_d is NP-hard to compute for all $d \geq 2$.

We now describe each of these steps in order.

Step 1: Restrict to a Top OR Gate. The idea in Step (1) to restrict the top gate of the formula is also used in the aforementioned result of Buchfuhrer and Umans [20]. However, the method they use to restrict the top gate can blow up the size of the corresponding truth table exponentially. We modify their approach using existing depth hierarchy theorems for AC^0 (the statement of the depth-hierarchy theorem in [31] is easiest for us to use) in order to give a quasipolynomial time reduction from computing $\mathsf{L}^{\mathsf{OR}}_d$ to L_d.

We note that this is the only part of our proof that makes use of classical "switching lemma style" lower bound techniques. This dependence, however, is not strictly necessarily: we show that one can avoid "switching lemma" type techniques altogether at the cost of losing some hardness of approximation.

At a high-level, the key idea for how to prove step (1) is to take the direct sum of f with a function g that is much easier to compute with a top OR gate than a top AND gate in order to force any optimal depth-d formula for computing the direct sum to use a top OR gate.

Step 2: $d = 2$ Base Case. In step (2), we use the NP-hardness of computing $\mathsf{L}^{\mathsf{OR}}_d$ to any constant factor when $d = 2$ as the base case of our inductive approach. This result (actually a stronger version) was first proved in the work of Feldman [14] and Allender et al. [13] and was subsequently improved by Khot and Saket [15]. There is a technicality in that these results use a slightly different size measure for DNFs: the number of terms in a DNF rather than the number of leaves. However, we show that there is an easy reduction between computing the two size measures for DNFs.

Step 3: $d \geq 3$ Inductive Argument. Finally, Step (3)'s connection between computing $\mathsf{L}^{\mathsf{OR}}_d$ and $\mathsf{L}^{\mathsf{OR}}_{d-1}$ is the heart of our reduction and required several new ideas. Since the goal in this step is to be able to compute $\mathsf{L}^{\mathsf{OR}}_{d-1}(f)$ for some function f using an oracle to $\mathsf{L}^{\mathsf{OR}}_d$, a natural approach is to construct some function F such that any optimal $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula for F must "contain" an optimal $\mathsf{OR} \circ \mathsf{AC}^0_{d-2}$ formula for f "within" it. Our original hope was to be able to force such a situation using a "switching lemma style" argument, but we were not able to make this approach to work.

Instead, we take an approach based on direct sums. Our proof of step (3) begins with an observation that, while trivial, was an important perspective switch (at least for the author): DeMorgan's laws imply that $\mathsf{L}^{\mathsf{OR}}_{d-1}(f) = \mathsf{L}^{\mathsf{AND}}_{d-1}(\neg f)$ for all functions f. Thus, if we want to compute $\mathsf{L}^{\mathsf{OR}}_{d-1}(f)$ given an oracle to L_d for any function f, it suffices to show how to compute $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ using an oracle to L_d for any function f.

The natural approach mentioned above then becomes to try constructing a function F such that any optimal $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula for F contains an optimal $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula for f within it. A reasonable candidate for F is the direct sum of f with another function g, that is $F(x, y) = f(x) \wedge g(y)$.

One can gain some intuition for the complexity of F by examining the following family of formulas for computing $f(x) \wedge g(y)$. Suppose φ and ψ are $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formulas for computing f and g respectively. Then we can expand $\varphi = \bigvee_{i \in [t_f]} \varphi_i$ where each φ_i is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula and t_f is the top fan-in of φ. Similarly, write $\psi = \bigvee_{j \in [t_g]} \psi_j$.

Observe that, by distributivity, we can then compute F as

$$\bigvee_{i \in [t_f], j \in [t_g]} (\varphi_i(x) \wedge \psi_j(y)).$$

This yields a formula for computing f of size

$$|\varphi| \cdot t_g + |\psi| \cdot t_f.$$

Hence, if computing g is significantly more expensive than computing f and g has an optimal formula with top fan-in $t_g = 1$, then the optimal formula for F within this family is plausibly obtained by picking a formula φ for computing f that has top fan-in $t_f = 1$ (i.e. φ is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula computing f). In this case, we would have our desired property that optimal formulas for F contain an optimal $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula for f within them. Our main lower bound is a partial formalization of this intuition. We

state this result informally here and point the reader to the full version for a formal statement.

Theorem 4 (Informal)**.** *Let f be a boolean function, and let g be a function that is "expensive" to compute compared to f. Then*

$$\begin{aligned}\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g) &\leq \mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g(y)) \\ &\leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{AND}}_{d-1}(g).\end{aligned}$$

The proof of Theorem 4 is, in our opinion, our most interesting proof. Unfortunately, even stating Theorem 4 formally requires more definitions than should be given in an extended abstract. Roughly speaking, however, g is "expensive" compared to f if computing even a weak one-sided approximation of g using *non-deterministic* formulas is more expensive than computing f exactly with $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formulas.

One can view Theorem 4 as saying that the family we described above for computing F is nearly optimal when computing g is "expensive" compared to f and when g has an near-optimal $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula with top fan-in one (i.e. a $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula).

Indeed, Theorem 4 implies that the quantity

$$\mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g(y)) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$$

gives an additive approximation to $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ with error bounded by $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$. This is how our reduction estimates $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$.

While we do not describe the details of our reduction here, there are three important details (phrased as questions) we would like to highlight about getting the reduction to work:

- *How do we get our hands on such g?* We need g to satisfy two properties: be expensive relative to f and have the quantity $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$ be small. Uniformly random functions (with the right parameters) are expensive, but when $d = 3$, the quantity $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$ is *not* small for such uniformly random g. We get around this by selecting our g to be drawn randomly from a set of functions that roughly corresponds to the subfunctions computed by CNF subformulas in Lupanov's construction of near optimal depth-3 formulas for random functions [32]. In this way, we get functions that are essentially optimally computed by CNFs but also have properties expected of random functions.
- *Without knowing the complexity of f, how can we know that g is expensive compared to f?* In our reduction we have to balance how expensive g is with how large $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$ is, since as g gets more expensive $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$ also gets larger. Thus, in some sense we need to know the complexity of f in order to ensure the approximation error we get is small. The idea we use is to successively iterate through all the possibilities for the complexity of f from high to low, and only output an estimate for f the first time the estimate significantly exceeds the error bound $\mathsf{L}^{\mathsf{AND}}_{d-1}(g) - \mathsf{L}^{\mathsf{OR}}_{d}(g)$.
- *How does the approximation error propagate as we go to higher and higher depths?* Because our method for computing $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ involves some additive error, we must be careful that at each depth we prove enough hardness of approximation in order to imply hardness for the next depth. Indeed, we show that for each $d \geq 3$ there is an $\alpha > 0$ such that it is NP-hard to approximate $\mathsf{L}^{\mathsf{OR}}_{d}$ to within a factor of $(1 + \alpha)$.

B. Hardness for $\mathsf{MCSP}^\star$

The heart of our hardness proof for $\mathsf{MCSP}^\star$ is the trivial lower bound for computing OR_n (the OR function on n bits). One can easily characterize what the optimal circuits for OR_n look like: all optimal circuits for OR_n are given by taking a rooted binary tree with exactly n-leaves, labelling the internal nodes by fan-in two OR gates, and labelling each leaf node with an input variable in the set $\{x_1, \ldots, x_n\}$ bijectively. This last part is crucial for us, since it implies there are at least $n!$ many optimal circuits for computing OR_n. It also suggests that one might be able to associate optimal circuits for OR_n with permutations.

Indeed this is the approach we take. Our starting point is the $2n \times 2n$ Bipartite Permutation Independent Set problem defined by Lokshtanov, Marx, and Saurabh [28], who showed that, under ETH, one cannot solve $2n \times 2n$ Bipartite Permutation Independent Set much faster than brute forcing over all $n!$ permutations, specifically not as fast as $2^{o(n \log n)}$. For our high-level description, all the reader needs to know about $2n \times 2n$ Bipartite Permutation Independent Set is that it

- asks whether there is a permutation $\pi : [2n] \to [2n]$ satisfying certain properties, and
- it cannot be solved in time $2^{o(n \log n)}$ under ETH.

Our reduction works by showing that given some instance I of $2n \times 2n$ Bipartite Permutation Independent Set, one can construct a partial function $\gamma : \{0,1\}^{2n} \times \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1\}$ such that

$$\begin{aligned}&\text{there exists a permutation } \pi \text{ satisfying } I \\ \iff &\exists \pi \text{ so } \bigvee_{i \in [2n]} (z_i \wedge (y_i \vee x_{\pi(i)})) \text{ computes } \gamma(x, y, z) \\ \iff &\text{a monotone read once formula computes } \gamma \\ \iff &\mathsf{MCSP}^\star(\gamma, 6n - 1) = 1.\end{aligned}$$

We note that all the lower bound techniques used in our proof of correctness are classical and can, for example, be found in Wegner's text on Boolean functions [33]. However, we do highlight the specific way we use the gate elimination technique, since it will be relevant to our discussion in section V regrading the Natural Proofs framework.

"Reverse" Gate Elimination. One usually uses gate elimination to say that if some circuit C computes some

function f, then one can obtain a smaller circuit C' for computing a restriction $f' = f|_\sigma$ of f by applying various simplifications to C that eliminate gates in f. Reverse gate elimination is the same technique but with a "reverse perspective."

Suppose C is a circuit of size s for computing f and $f' = f|_\sigma$ is some restriction of f. Assume that gate elimination implies that one can eliminate k gates from C to obtain a circuit C' of size $s-k$ for f'. Then, equivalently, we have that the circuit C can be obtained by taking C' and "un-eliminating" (i.e. adding) gates to C' in a specific manner that is dual to the way gates are eliminated in gate elimination. Thus, if one knows what the circuits for f' of size $s-k$ look like (as is the case with circuits for OR_n of size $n-1$), one can constrain what circuits of size s for computing f look like.

We use this technique to argue that any circuit for computing γ has an optimal OR_n circuit "within it," which we can associate with a permutation.

We note that the "reverse gate elimination" technique was also used in [34] to show a non-trivial search-to-decision reduction for $(\mathsf{Formula})$-MCSP. In fact, functions with many optimal formulas, like the OR_n function, precisely correspond to the hard instances for the algorithm in [34].

V. Connections with Constructivity and the Natural Proofs Barrier

There are close connections between MCSP and Razborov and Rudich's Natural Proofs barrier [24]. In this section, we will focus on one specific connection between designing reductions to $(\mathcal{C})$-MCSP and a strengthening of the constructivity condition in the Natural Proofs barrier.[3] We begin by describing the connection informally, before going into more detail.

Intuition. Roughly speaking, Razborov and Rudich's celebrated Natural Proofs result shows that any "natural" lower bound against a circuit class $\mathcal{C}$ can be made "algorithmic" and that this algorithm can be used to defeat certain types of cryptography constructed within the circuit class $\mathcal{C}$. Since the general belief is that strong cryptography exists in even relatively weak looking circuit classes $\mathcal{C}$, Razborov and Rudich's result suggests it is unlikely that there are "natural proofs" showing strong lower bounds against many circuit classes.

The relevance of this to $(\mathcal{C})$-MCSP is as follows. Suppose one has a reduction R from SAT to $(\mathcal{C})$-MCSP. In the proof of correctness of this reduction, one must use some lower bound method $\mathcal{M}$ against $\mathcal{C}$-circuits. If this method $\mathcal{M}$ were "natural," then $\mathcal{M}$ could be made "algorithmic." But then we argue that one could plug the algorithmic version of $\mathcal{M}$ into the reduction R and obtain an efficient algorithm for SAT. Hence, if one believes that SAT does not have efficient algorithms, one should also believe that the lower bound method $\mathcal{M}$ cannot be made "algorithmic" (at least without making modifications to $\mathcal{M}$).

A More Formal Description. We now describe this idea in more detail. A "lower bound method" $\mathcal{M}$ is not a formal notion, so we instead look at collections of lower bound statements $\mathcal{S}$, whose elements are of the form (T, s) where T is a truth table and s is a lower bound on the complexity of T. For most lower bound methods $\mathcal{M}$, there is a natural choice of the lower bound statements $\mathcal{S}_\mathcal{M}$ that $\mathcal{M}$ "proves," although we note that whether a $\mathcal{M}$ "proves" a lower bound statement is not necessarily well-defined.

One example where it is easy to define $\mathcal{S}_\mathcal{M}$ is Håstad's switching lemma, which implies that if a function $f : \{0,1\}^n \to \{0,1\}$ cannot be made to compute a constant function by setting $n-k$ of its inputs to $0/1$-values, then f cannot be computed by a depth-d circuit of size $2^{(n-k)^{\Omega(1/d)}}$ [23]. A natural choice of the collection of lower bound statements associated with the switching lemma is

$$\mathcal{S}_\mathcal{M} = \{(T,s) : T \text{ is not constant on any subcube of dimension } k \text{ and } s < 2^{(n-k)^{\Omega(1/d)}}\}.$$

The connection to $(\mathcal{C})$-MCSP is as follows. Suppose one had a polynomial-time many-one reduction R from, say, SAT to $(\mathcal{C})$-MCSP. In the proof of correctness for this reduction, one must have some method for proving a collection of lower bound statements $\mathcal{S}$ such that if φ is unsatisfiable and (T, s) is output by the reduction, then the lower bound statement that the $\mathcal{C}$-complexity of T is greater than s is an element of $\mathcal{S}$, i.e. $(T, s) \in \mathcal{S}$. On the other hand if φ is satisfiable and the reduction outputs (T, s), then we know that the $\mathcal{C}$-complexity of T is at most s, so $(T, s) \notin \mathcal{S}$ because we require that $\mathcal{S}$ only contains correct lower bounds.

Hence, we can conclude that the reduction R actually also implies that recognizing elements of $\mathcal{S}$ is coNP-hard! In fact, it shows that even the promise problem of distinguishing the lower bounds contained in $\mathcal{S}$ from strings in the set of YES instances of $(\mathcal{C})$-MCSP

$$\{(T,s) : \text{the truth table } T \text{ has } \mathcal{C}\text{-circuits of size} \leq s\}$$

is coNP-hard. Thus, if one believes that, say, $\mathsf{coNP} \not\subseteq \mathsf{P/poly}$, it better not be the case that the language $\mathcal{S}$ can be computed in $\mathsf{P/poly}$.

With this in mind, we say a collection of lower bound statements $\mathcal{S}$ against a circuit class $\mathcal{C}$ is $(\mathsf{P/poly})$-*recognizable* if there exists a family of polynomial-sized circuits that accepts all elements of $\mathcal{S}$ and rejects all the YES instances of $(\mathcal{C})$-MCSP. The logic above demonstrates that, under widely believed complexity assumptions, one should not be able to prove hardness for $(\mathcal{C})$-MCSP using $(\mathsf{P/poly})$-recognizable collections of lower bound statements. This is

[3] To the author's knowledge, this connection was first observed in a conversation between the author and Rahul Santhanam, who kindly allowed for its inclusion here.

interesting because many lower bound methods we know, like Håstad's switching lemma, yield collections of lower bound statements that are $(\mathsf{P}/\mathsf{poly})$-recognizable.

One nice property of the definition of $(\mathsf{P}/\mathsf{poly})$-recognizability is monotonicity: if a set of lower bound statements $\mathcal{S}$ is $(\mathsf{P}/\mathsf{poly})$-recognizable, then all subsets of S are also $(\mathsf{P}/\mathsf{poly})$-recognizable. In the contrapositive, if a set $\mathcal{S}$ is not $(\mathsf{P}/\mathsf{poly})$-recognizable, then any set that contains $\mathcal{S}$ is also not $(\mathsf{P}/\mathsf{poly})$-recognizable. This is a consequence of the promise problem underlying the definition.

Finally, we note that a collection of lower bound statements being $(\mathsf{P}/\mathsf{poly})$-recognizable is closely related to Razborov and Rudich's notion of $(\mathsf{P}/\mathsf{poly})$-constructive. The main difference being that Razborov and Rudich's formalization is only concerned with lower bound statements where the size lower bound s is fixed to some particular (usually super-polynomial) value.

The Takeaway. Perhaps the most useful consequence of this connection is that it gives a helpful tool for designing reductions to $(\mathcal{C})$-MCSP, since it rules out many approaches that solely rely on easily recognizable lower bound statements. Indeed, our proof that $\mathsf{MCSP}^\star$ is not in P under ETH was inspired by our failure to rule out lower bounds obtained by gate elimination within this framework.

This connection may also give further motivation for proving hardness results for $(\mathcal{C})$-MCSP. Since the collection of lower bound statements used to prove hardness for $(\mathcal{C})$-MCSP (likely) cannot be $(\mathsf{P}/\mathsf{poly})$-recognizable, any proof requires considering lower bounds of a slightly different flavor than many existing lower bound techniques. One might hope that these different lower bound techniques might also be useful in understanding other questions about the class $\mathcal{C}$ and, optimistically, might be a step towards proving non-naturalizing lower bounds.

Indeed, our hardness result for (AC^0_d)-MCSP gives evidence for these two motivations. Using the novel lower bound techniques in our reduction, we prove our "large gaps in formula complexity between depths" result (Theorem 2). Previous techniques like random restrictions do not seem capable of achieving the parameters in Theorem 2 (since random restrictions typically establish lower bounds of the form $2^{n^{O(1/d)}}$ and our lower bound has a much better dependence on d).

Moreover, if we view Theorem 2 as separating the class of size-s depth-$(d+1)$ formulas from size-$(s+2^{O_d(n^{1/5})})$ depth-d formulas for some s, it is not clear to what extent this circuit class separation naturalizes in the sense of Razborov and Rudich's Natural Proofs Barrier. For one, our method only proves a lower bound on a specific class of functions obtained via a direct sum. This seems to violate the largeness condition of a natural proof, which roughly says that the lower bound method should apply to a significant fraction of functions. It is worth noting that (to the author's knowledge) it is open whether uniformly random functions $f : \{0,1\}^n \to \{0,1\}$ have a gap as large as

$$\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f) \geq 2^{n^{\Omega(1)}}$$

with high probability. Lupanov showed that

$$\mathsf{L}_d(f) = (1 + o(1))\mathsf{L}_{d+1}(f)$$

when $d \geq 3$ with high probability [32]. Second, it is not clear how to recognize the functions witnessing this lower bound in polynomial time given a truth table. This seems to violate the constructivity condition of a Natural Proof.

Of course, this does not mean that this separation does not naturalize, just that it does not obviously naturalize. Since results can naturalize in highly non-trivial ways (we mention an example in the next paragraph), it would be interesting to explore whether one can put this result in the framework of Natural Proofs. Either way, we view this result as a compelling example of the further insights that understanding $(\mathcal{C})$-MCSP could give.

Caveats. Even though a collection of lower bound statements $\mathcal{S}$ might not be $(\mathsf{P}/\mathsf{poly})$-recognizable, it is possible that there is a variation $\mathcal{S}'$ of $\mathcal{S}$ that is $(\mathsf{P}/\mathsf{poly})$-recognizable and still captures all the "interesting" lower bounds given by $\mathcal{S}$. A situation like this occurs in Razborov and Rudich's paper where they show how to modify Smolensky's [35] lower bound against $\mathsf{AC}^0[p]$ circuits to fit into the natural proofs framework, even though it is unclear whether Smolensky's original method is constructive.

That being said, if a collection of lower bound statements $\mathcal{S}$ is used to prove hardness for $(\mathcal{C})$-MCSP, then any $(\mathsf{P}/\mathsf{poly})$-recognizable modification $\mathcal{S}'$ (likely) loses the ability to prove hardness of $(\mathcal{C})$-MCSP, so it seems like some "interesting" lower bounds must be lost in this case.

Another caveat worth mentioning is that our logic above assumes that the reduction from SAT to $(\mathcal{C})$-MCSP is a deterministic many-one reduction. In contrast, one can imagine more exotic reductions, where it is not clear how to define the collection of lower bound statements $\mathcal{S}$ used to prove the correctness of a reduction. Nevertheless, we feel that our logic is broadly applicable. In the specific reductions we prove (one is a deterministic many-one reduction and one is a randomized quasipolynomial time Turing reduction), the definition of $\mathcal{S}$ does makes sense, and we can indeed carry out a version of the logic above in order to argue that $\mathcal{S}$ is hard.

ACKNOWLEDGMENT

I would like to give a special thanks to Rahul Santhanam for valuable discussions on this work. The origins of this paper can be traced to a fruitful visit to his research group. In addition, I'm grateful to Eric Allender, Shuichi Hirahara, Bruno Loff, Dylan McKay, Igor Oliveira, Ján Pich, Ninad Rajgopal, Michael Saks, and Ryan Williams for helpful perspectives and remarks on our results and techniques.

I also want to give a profuse thanks to the anonymous FOCS reviewers for their patience with the technical details in an earlier version of the paper and because their extremely detailed comments improved our presentation significantly.

This research was supported by an Akamai Presidential Fellowship and by NSF Grants CCF-1741615 and CCF-1909429.

REFERENCES

[1] V. Kabanets and J. Cai, "Circuit minimization problem," in *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, 2000, pp. 73–79.

[2] E. Allender, "The new complexity landscape around circuit minimization," in *Language and Automata Theory and Applications - 14th International Conference, LATA 2020, Milan, Italy, March 4-6, 2020, Proceedings*, vol. 12038, 2020, pp. 3–16.

[3] C. D. Murray and R. R. Williams, "On the (non) NP-hardness of computing circuit complexity," in *30th Conference on Computational Complexity, CCC 2015, June 17-19, 2015, Portland, Oregon, USA*, vol. 33, 2015, pp. 365–380.

[4] D. M. McKay, C. D. Murray, and R. R. Williams, "Weak lower bounds on resource-bounded compression imply strong separations of complexity classes," in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, 2019, pp. 1215–1225.

[5] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM J. Comput.*, vol. 28, no. 4, pp. 1364–1396, 1999.

[6] R. Santhanam, "Pseudorandomness and the minimum circuit size problem," in *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, vol. 151, 2020, pp. 68:1–68:26.

[7] M. L. Carmosino, R. Impagliazzo, V. Kabanets, and A. Kolokolova, "Learning algorithms from natural proofs," in *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, vol. 50, 2016, pp. 10:1–10:24.

[8] S. Hirahara, "Non-black-box worst-case to average-case reductions within NP," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 25, p. 138, 2018.

[9] B. A. Trakhtenbrot, "A survey of russian approaches to perebor (brute-force searches) algorithms," *IEEE Annals of the History of Computing*, vol. 6, no. 4, pp. 384–400, 1984.

[10] E. Allender, M. Koucký, D. Ronneburger, and S. Roy, "The pervasive reach of resource-bounded kolmogorov complexity in computational complexity theory," *Journal of Computer and System Sciences*, vol. 77, no. 1, pp. 14 – 40, 2011.

[11] S. L. A. Czort, "The complexity of minimizing disjunctive normal form formulas," Master's thesis, University of Aarhus, 1999.

[12] C. Umans, T. Villa, and A. L. Sangiovanni-Vincentelli, "Complexity of two-level logic minimization," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 25, no. 7, pp. 1230–1246, 2006.

[13] E. Allender, L. Hellerstein, P. McCabe, T. Pitassi, and M. E. Saks, "Minimizing DNF formulas and AC0d circuits given a truth table," in *21st Annual IEEE Conference on Computational Complexity (CCC 2006), 16-20 July 2006, Prague, Czech Republic*, 2006, pp. 237–251.

[14] V. Feldman, "Hardness of approximate two-level logic minimization and PAC learning with membership queries," in *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*, 2006, pp. 363–372.

[15] S. Khot and R. Saket, "Hardness of minimizing and learning DNF expressions," in *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, 2008, pp. 231–240.

[16] S. Hirahara, I. C. Oliveira, and R. Santhanam, "NP-hardness of minimum circuit size problem for OR-AND-MOD circuits," in *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, vol. 102, 2018, pp. 5:1–5:31.

[17] T. Hancock, T. Jiang, M. Li, and J. Tromp, "Lower bounds on learning decision lists and trees," *Inf. Comput.*, vol. 126, no. 2, p. 114–122, May 1996.

[18] M. Alekhnovich, M. Braverman, V. Feldman, A. R. Klivans, and T. Pitassi, "The complexity of properly learning simple concept classes," *J. Comput. Syst. Sci.*, vol. 74, no. 1, p. 16–34, Feb. 2008.

[19] W. J. Masek, "Some NP-complete set covering problems," 1979, unpublished Manuscript.

[20] D. Buchfuhrer and C. Umans, "The complexity of boolean formula minimization," *J. Comput. Syst. Sci.*, vol. 77, no. 1, pp. 142–153, 2011.

[21] R. Ilango, "Approaching MCSP from above and below: Hardness for a conditional variant and AC0[p]," in *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, vol. 151, 2020, pp. 34:1–34:26.

[22] R. Ilango, B. Loff, and I. C. Oliveira, "NP-hardness of circuit minimization for multi-output functions," in *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, vol. 169, 2020, pp. 22:1–22:36.

[23] J. Håstad, "Almost optimal lower bounds for small depth circuits," *Advances in Computing Research*, vol. 5, pp. 143–170, 1989.

[24] A. A. Razborov and S. Rudich, "Natural proofs," *J. Comput. Syst. Sci.*, vol. 55, no. 1, pp. 24–35, 1997.

[25] R. Impagliazzo and R. Paturi, "On the complexity of k-SAT," *J. Comput. Syst. Sci.*, vol. 62, no. 2, pp. 367–375, 2001.

[26] R. Impagliazzo, R. Paturi, and F. Zane, "Which problems have strongly exponential complexity?" *J. Comput. Syst. Sci.*, vol. 63, no. 4, pp. 512–530, 2001.

[27] D. Lokshtanov, D. Marx, and S. Saurabh, "Lower bounds based on the exponential time hypothesis," *Bull. EATCS*, vol. 105, pp. 41–72, 2011.

[28] ——, "Slightly superexponential parameterized problems," *SIAM J. Comput.*, vol. 47, no. 3, pp. 675–702, 2018.

[29] D. Angluin, L. Hellerstein, and M. Karpinski, "Learning read-once formulas with queries," *J. ACM*, vol. 40, no. 1, p. 185–210, Jan. 1993.

[30] R. Williams, Personal Communication.

[31] J. Håstad, B. Rossman, R. A. Servedio, and L. Tan, "An average-case depth hierarchy theorem for boolean circuits," *J. ACM*, vol. 64, no. 5, pp. 35:1–35:27, 2017.

[32] O. B. Lupanov, "On the realization of functions of logical algebra by formula of finite classes (formula of limited depth) in the basis &, v, -*," *Problemy Kibernetiki*, vol. 6, pp. 5–14, 1961.

[33] I. Wegener, *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

[34] R. Ilango, "Connecting perebor conjectures: Towards a search to decision reduction for minimizing formulas," in *35th Computational Complexity Conference, CCC 2020, July 28-31, 2020, Saarbrücken, Germany (Virtual Conference)*, vol. 169, 2020, pp. 31:1–31:35.

[35] R. Smolensky, "Algebraic methods in the theory of lower bounds for boolean circuit complexity," in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, 1987, pp. 77–82.