Electronic Colloquium on Computational Complexity, Report No. 183 (2020)

Constant Depth Formula and Partial Function Versions of MCSP are Hard

Rahul Ilango
Massachusetts Institute of Technology, 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 $2^{\Omega_d(n)}$ 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 $\{0, 1, \star\}^*$) 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 collection of lower bound statements used to prove the correctness of our reductions *cannot* be (P/poly)-recognizable.

2012 ACM Subject Classification Theory of computation → Circuit complexity; Theory of computation → Problems, reductions and completeness

Keywords and phrases Minimum Circuit Size Problem, NP hardness, Circuit Lower Bounds, Natural Proofs Barrier, Constant Depth Formulas, Minimum Formula Size Problem, Exponential Time Hypothesis

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

Acknowledgements 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 aspects in an earlier version of the paper and because their extremely detailed comments improved the paper significantly.

ISSN 1433-8092

Contents

1 Introduction

1.1 Background and Motivation

1.1.1 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 $s \in \mathbb{N}$.

Kabanets and Cai [22] 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 $\mathsf{EXP} \neq \mathsf{ZPP}$ [29].
- If MCSP with a fixed size parameter $s = \mathsf{poly}(n)$ does not have circuits of size $\tilde{O}(N)$, then $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ [28].
- If $\mathsf{MCSP} \in \mathsf{P}$, then there are no one-way functions [22, 12].
- 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) [31].
- There is an equivalence between learning a circuit class $\mathcal{C}$ and the problem of "approximately minimizing" $\mathcal{C}$-circuits [8].
- If a certain approximation to MCSP is NP-hard, then there is a "worst-case to average-case" reduction for NP [15].

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

1.1.2 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 [33] for a history of 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 [22].

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.

[1] [4] 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.

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 [9, 34, 3, 10, 23] 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).

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 $(\mathcal{C})$-$\mathsf{MCSP}^\star$ which we define now.[2]

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

- **Given:** the 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 $\mathsf{MCSP}^\star$ is NP-complete [11, 1]. 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 $(\mathcal{C})$-$\mathsf{MCSP}^\star$. Second, reduce $(\mathcal{C})$-$\mathsf{MCSP}^\star$ to $(\mathcal{C})$-MCSP.

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

1.2 Results and Discussion

1.2.1 $(\mathcal{C})$-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 Turing reductions when $\mathcal{C}$ is the class, denoted AC^0_d, of depth-d *formulas* with AND/OR gates

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

of unbounded fan-in.

▶ **Theorem 1** (also Theorem 22). *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 1.3. 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 [27] proved that (DNF)-MCSP is NP-hard in the 1970s, and Hirahara, Oliveira, and Santhanam [16] recently showed that ($\mathsf{DNF} \circ \mathsf{XOR}$)-$\mathsf{MCSP}$ 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 [27] and [16], though their work forms our base case.

Another related work is the innovative paper of Buchfuhrer and Umans [7], 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 [3] 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 [22, 29], no such barriers are known for randomized reductions.

Indeed, some recent results show that for close variants of MCSP, like an oracle variant [17] and a multi-output variant [19], 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, additively. 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 $2^{n^{\Omega(1/d)}}$ [14].

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

▶ **Theorem 2** (Proved in Section 9). *For all $d \geq 2$ there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f) \geq 2^{\Omega_d(n)}$.*

Our proof works by "lifting" the $2^{\Omega(n)}$ seperation the parity function gives in the $d = 2$ case to higher depths at a low cost. We sketch the proof of the main technique used here in Section 1.3.2.

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 [30]. We defer discussion about this to Section 1.4.

1.2.2 $(\mathcal{C})$-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).

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** (also Theorem 11). *Assume ETH holds. Then there is no deterministic algorithm for solving MCSP* that runs 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 algorithm for deciding whether T can be computed by a monotone read once formula that runs in time $N^{o(\log\log N)}$.*

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 [25] 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 1.3.

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 [11, 1], if the same were known for MCSP*, then the breakthrough separation EXP $\neq$ ZPP would follow from an argument by Murray and Williams [29].

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 [5], 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 [36] 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.[3]

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 1.3) 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

[3] Subsequent to this work, the author was able to prove that (Formula)-MCSP is not in P under ETH by giving a reduction from (Formula)-MCSP* to (Formula)-MCSP.

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.

1.3 Proof Ideas

1.3.1 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. $\mathsf{L}_d^{\mathsf{OR}}(f)$ and $\mathsf{L}_d^{\mathsf{AND}}(f)$ 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 $\mathsf{L}_d^{\mathsf{OR}}$ to L_d, so it suffices to prove NP hardness for $\mathsf{L}_d^{\mathsf{OR}}$.
2. Show that when $d = 2$ it is NP-hard to compute $\mathsf{L}_d^{\mathsf{OR}}$ within any constant factor (this part was already known).
3. Show that when $d \geq 3$ one can compute a small approximation of $\mathsf{L}_{d-1}^{\mathsf{OR}}$ using an oracle that computes a small approximation of $\mathsf{L}_d^{\mathsf{OR}}$. 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 [7]. 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 [13] is easiest for us to use) in order to give a quasipolynomial time reduction from computing $\mathsf{L}_d^{\mathsf{OR}}$ 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 also show that one can avoid "switching lemma" type techniques in the proof 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 $\mathsf{L}_d^{\mathsf{OR}}$ 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 [10] and Allender et al. [3] and was subsequently improved by Khot and Saket [23]. 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 $\mathsf{L}_d^{\mathsf{OR}}$ and $\mathsf{L}_{d-1}^{\mathsf{OR}}$ 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 figure out how 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.

▶ **Theorem 4** (Informal version of Theorem 5)**.** *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. We state the theorem formally and give a sketch of the proof in Section 1.3.2. 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. The full proof of Theorem 4 can be found in Section 4.

Theorem 4 implies that, when g is chosen carefully, 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 $\mathsf{L}_{d-1}^{\mathsf{AND}}(g) - \mathsf{L}_d^{\mathsf{OR}}(g)$ be small. Uniformly random functions (with the right parameters) are expensive, but when $d = 3$, the quantity $\mathsf{L}_{d-1}^{\mathsf{AND}}(g) - \mathsf{L}_d^{\mathsf{OR}}(g)$ 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 [26]. 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 $\mathsf{L}_{d-1}^{\mathsf{AND}}(g) - \mathsf{L}_d^{\mathsf{OR}}(g)$ is, since as g gets more expensive $\mathsf{L}_{d-1}^{\mathsf{AND}}(g) - \mathsf{L}_d^{\mathsf{OR}}(g)$ 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 $\mathsf{L}_{d-1}^{\mathsf{AND}}(g) - \mathsf{L}_d^{\mathsf{OR}}(g)$.
- *How does the approximation error propagate as we go to higher and higher depths?* Because our method for computing $\mathsf{L}_{d-1}^{\mathsf{AND}}(f)$ 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 $\mathsf{L}_d^{\mathsf{OR}}$ to within a factor of $(1 + \alpha)$.

1.3.2 Proof Sketch: Main Constant Depth Formula Lower Bound

In this subsection we sketch the proof of Theorem 4, which we previously stated informally. The full proof of Theorem 4 can be found in Section 4.

Before giving the formal statement, we introduce some notation. A *non-deterministic formula* φ with n-inputs and m non-deterministic inputs is just a (standard) formula ψ with $(n+m)$-inputs with its last m inputs designated as "non-deterministic" inputs. φ evaluated at an input $x \in \{0,1\}^n$ equals $\bigvee_{y \in \{0,1\}^m} \psi(x, y)$. The size of φ is the same as the size of ψ: the number of leaves in the underlying binary tree. We use the notation $\mathsf{L}_{\mathsf{ND}}(f)$ to denote the minimum size of any non-deterministic formula with n (regular) inputs and n non-deterministic inputs for computing f. In this paper we will only consider non-deterministic formulas that have the same number of regular and non-deterministic inputs.

If $0 \leq \epsilon \leq 1$, we say a function $g : \{0,1\}^n \to \{0,1\}$ is an ϵ *one-sided approximation* of $f : \{0,1\}^n \to \{0,1\}$ if $g^{-1}(1) \subseteq f^{-1}(1)$ and $|g^{-1}(1)| \geq \epsilon |f^{-1}(1)|$. We let $\mathsf{L}_{\mathsf{ND},\epsilon}(f)$ denote minimum of $\mathsf{L}_{\mathsf{ND}}(g)$ among all g that are ϵ one-sided approximations of f.

We now give the formal statement of Theorem 4. The proof of this theorem can be found in Section 4.

▶ **Theorem 5.** *Let $d \geq 3$. Let $\gamma = \frac{1}{10^4}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant function, and let $g : \{0,1\}^m \to \{0,1\}$ be a non-constant function with $m \geq n$ that satisfies*

$$\min\{2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g), \mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g)\} \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f).$$

Then

$$\mathsf{L}_d^{\mathsf{OR}}(f(x) \wedge g(y)) \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f).$$

Our approach is a proof by contradiction. Suppose the hypotheses of the theorem hold and that there is an $\mathsf{OR} \circ \mathsf{AC}_{d-1}^0$ formula φ for computing $f(x) \wedge g(y)$ with less than $\mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$ leaves.

We begin by writing $\varphi = \bigvee_{i\in[t]} \varphi_i$ where each φ_i is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula. The key idea of our proof is to view each φ_i as a *non-deterministic* formula with y being its regular input and x being its non-deterministic input. In particular, for each $i \in [t]$ let $S_i \subseteq \{0,1\}^m$ be the subset of inputs accepted non-deterministically by φ_i. In other words

$$S_i = \{y : \exists x \text{ such that } \varphi_i(x,y) = 1\}.$$

Since $\varphi = \bigvee_{i\in[t]} \varphi_i$ computes $f(x) \wedge g(y)$ and f is not constant, it follows that the union of the S_i sets is precisely $g^{-1}(1)$. However using the assumption that φ has less than $\mathsf{L}^{\mathsf{OR}}_d(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ leaves, we show something stronger must occur: the sets $S_1, \dots, S_t$ must cover $g^{-1}(1)$ *redundantly*. Formally, we mean that for each element $y^1 \in g^{-1}(1)$, there exists some $i \neq j$ such that $y^1 \in S_i$ and $y^1 \in S_j$. Intuitively this represents a redundancy that we will exploit to contradict our assumptions.

Before we continue, we try to give some intuition for why the sets $S_1, \dots, S_t$ must form a redundant cover of $g^{-1}(1)$. Suppose that there was some $y^1 \in g^{-1}(1)$ such that $y^1 \in S_1$ but $y^1 \notin S_2 \cup \dots \cup S_t$. By the definition of the sets S_i this implies that $\varphi_i(x, y^1) = 0$ for all x and all $i \geq 2$. Since φ computes $f(x) \wedge g(y)$ and $g(y^1) = 1$ this means that

$$f(x) = f(x) \wedge g(y^1) = \varphi(x, y^1) = \bigvee_{i\in[t]} \varphi_i(x, y^1) = \varphi_1(x, y^1)$$

so we can conclude that φ_1 can be used to compute f (by setting $y = y^1$). This implies that φ_1 has at least $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ many x-leaves since φ_1 is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula. This means that φ also has at least $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ many x-leaves. On the other hand, φ must have $\mathsf{L}^{\mathsf{OR}}_d(g)$ many y-leaves because we can make φ compute g by setting x to a YES instance of f. Hence, we can conclude φ has at least $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_d(g)$ many leaves which is a contradiction. This completes the intuition for why $S_1, \dots, S_t$ form a redundant cover of $g^{-1}(1)$.

We ultimately exploit this redundancy in order to produce a non-deterministic .73 one-sided approximation to g whose complexity is too small. The idea is as follows. Consider partitioning $[t]$ into two subsets L and R uniformly at random, and consider the non-deterministic formulas $\psi_L = \bigvee_{i\in L} \varphi_i$ and $\psi_R = \bigvee_{i\in R} \varphi_i$ where we view the x-input non-deterministically and y as the true input. Because φ computes $f(x) \wedge g(y)$, we can conclude that ψ_L and ψ_R each compute one-sided non-deterministic approximations for g. Moreover, the redundancy of the cover implies that *in expectation* they form a .75 one-sided approximation of g. This is because each element of $g^{-1}(1)$ is contained in at least two sets in the list $S_1, \dots, S_t$, so ψ_L and ψ_R each get at least "two chances" to get a subformula φ_i that non-deterministically accepts any given YES instance of g.

Now we would like to conclude that ψ_L and ψ_R are both .75 one-sided approximations of g and hence yield a contradiction because $|\psi_L| + |\psi_R| = |\varphi|$ (because L and R are a partition) and $|\varphi| \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_d(g)$ and we assumed that $2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g) \geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_d(g)$. However, we cannot conclude this since we only get that ψ_L and ψ_R are each .75 one-sided approximations *in expectation*. It could be the case that each time ψ_L is a .75 one-sided approximation that ψ_R is not and vice versa.

We get around this by proving that the random variables $|\psi_L^{-1}(1)|$ and $|\psi_R^{-1}(1)|$ concentrate around their expectation. We argue this concentration must occur as a consequence of the fact that $S_1, \dots, S_t$ redundantly covers $g^{-1}(1)$. In particular, we use redundancy to show that each set S_i has small cardinality. Consequently, the smallness of the S_i sets can be used to bound the variance of the random variables $|\psi_L^{-1}(1)|$ and $|\psi_R^{-1}(1)|$, which in turn implies by the second moment method that there is a choice of L and R such that ψ_L and ψ_R both form non-deterministic .73 one-sided approximations for g, which we use to show that ψ_L and ψ_R witness a contradiction to the assumption that $2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g) \geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_d(g)$.

We finish our sketch by giving the intuition for why the each of the sets $S_1, \ldots, S_t$ must have small cardinality. Fix some $j \in [t]$. The redundancy of the cover implies that the union of all the S_i sets excluding S_j still covers $g^{-1}(1)$. This means that $\bigvee_{i \in [t] \setminus \{j\}} \varphi_i$ is a non-deterministic formula for g. On the other hand, we know that φ_j is a $\frac{|S_j|}{|g^{-1}(1)|}$ one-sided approximation of g. Thus, because we assumed that $|\varphi| < \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g)$ and a hypothesis of the theorem is that $\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g) \geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g)$, we can conclude that it must be the case that $|S_j| \leq \gamma |g^{-1}(1)|$. The reasoning is that otherwise we would get that $\bigvee_{i \in [t] \setminus \{j\}} \varphi_i$ computes g non-deterministically and φ_j computes a γ one-sided approximation non-deterministically and that combined they have size at most $|\varphi| < \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g)$.

1.3.3 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 [25], 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 [35]. However, we do highlight the specific way we use the gate elimination technique, since it will be relevant to our discussion in Section 1.4 regarding 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 implicitly 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 [18] to show a non-trivial search-to-decision reduction for (Formula)-MCSP. In fact, functions with many optimal formulas, like the OR_n function, precisely correspond to the hard instances for the algorithm in [18].

1.4 Connections with Constructivity and the Natural Proofs Barrier

There are close connections between MCSP and Razborov and Rudich's Natural Proofs barrier [30]. In this subsection, 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.[4] 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 $\mathcal{M}$ could be made sufficiently "algorithmic," then 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 $\mathcal{S}$ of lower bound statements. In particular, we consider sets $\mathcal{S}$ 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 $2^{(n-k)^{\Omega(1/d)}}$ [14]. A natural choice of the collection of lower bound statements associated with the switching lemma is

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

[4] 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.

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 P/poly.

With this in mind, we say a collection of lower bound statements $\mathcal{S}$ against a circuit class $\mathcal{C}$ is (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 (P/poly)-recognizable collections of lower bound statements (at least under the usual type of reductions: many-one, deterministic, polynomial-time). This is interesting because many lower bound methods we know, like Håstad's switching lemma, yield collections of lower bound statements that are (P/poly)-recognizable.

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

Finally, we note that a collection of lower bound statements being (P/poly)-recognizable is closely related to Razborov and Rudich's notion of (P/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 (P/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-$(s+2^{O_d(n)})$ 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

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

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 [26]. 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 $(\mathsf{P/poly})$-recognizable, it is possible that there is a variation $\mathcal{S}'$ of $\mathcal{S}$ that is $(\mathsf{P/poly})$-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 [32] 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 $(\mathsf{P/poly})$-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 one can indeed carry out a version of the logic above in order to argue that $\mathcal{S}$ is hard.

If the reader is curious, the proof of correctness for our randomized quasipolynomial time Turing reduction implies that following collection of lower bound statements against $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formulas is hard for coNP under randomized quasipolynomial time Turing reductions:

$$\{(T,s) : T \text{ is the truth table of the function } f(x) \wedge g(y) \text{ where } f:\{0,1\}^n \to \{0,1\} \text{ and } g:\{0,1\}^m \to \{0,1\} \text{ are non-constant functions satisfying } m \geq n \text{ and } s \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f) \text{ and } \min\{2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g), \mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g)\} \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)\}.$$

where $\gamma = 10^{-4}$ and the notation $\mathsf{L}_{\mathsf{ND},\cdot}$ is defined in Section 2.1.

1.5 Open Questions

Perhaps the most tantalizing open question is whether one can show that MCSP is not in P under ETH. We discussed a potential approach to doing this at the end of Section 1.2.2.

There are also several intriguing open questions related to our (AC_d^0)-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 direction is better hardness of approximation for (AC_d^0)-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, what about improving the complexity gap result in Theorem 2? Can one give a multiplicative gap instead of an additive one? What about the case of circuits? Can one use our lower bound techniques to prove other interesting results?

2 Preliminaries

For a natural number n, we let $[n]$ denote the set $\{1,\ldots,n\}$. If E is some event, then we let $\mathbb{1}_E$ denote the indicator random variable that equals 1 if E occurs and 0 if E does not occur.

Big Oh Notation. We use the standard "big oh" notation O, o, Ω, ω with the convention that n will always be the parameter that is going to infinity. When there are multiple parameters, we use subscripts to denote parameters being held constant. For example $o_\delta(1)$ indicates a function that goes to zero as n goes to infinity and δ is held constant.

Binary Strings. For a binary string x, we let $\mathsf{wt}(x)$ denote the *weight* of x, that is the number of ones in x. Unless otherwise specified, if x is a binary string, then x_i denotes the ith bit of x.

Partial Functions. For us, *partial functions* will refer to functions of the form $\gamma : \{0,1\}^n \to \{0,1,\star\}$ for some n. We say a total function $f : \{0,1\}^n \to \{0,1\}$ *agrees* with γ if $f(x) = \gamma(x)$ for all x with $\gamma(x) \in \{0,1\}$. Similarly, a circuit (or formula) C computes a partial function γ if $C(x) = \gamma(x)$ for all x with $\gamma(x) \in \{0,1\}$.

Multiplicative Approximations. When $\alpha \geq 0$, we say a function $\mathcal{O}$ computes a $(1+\alpha)$ *multiplicative approximation* to a real-valued function f if for all inputs x

$$f(x) \leq \mathcal{O}(x) \leq (1+\alpha)f(x)$$

Textbook Background: Complexity Theory and Boolean Functions. We will make use of basic complexity theoretic notions such as P, NP, and various types of reductions that are explained, for example, in Arora and Barak's excellent textbook [6]. We will also assume knowledge of basic circuit lower bound techniques such as gate elimination that are described in Wegner's text [35].

The Exponential Time Hypothesis. The Exponential Time Hypothesis (abbreviated ETH) was first formulated by Impagliazzo, Paturi, and Zane [20, 21] and has been extremely useful for proving conditional lower bounds on various problems (see [24] 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.

Circuits. We use the usual model of general circuits with NOT gates and fan-in two AND and OR gates. The *size* of a circuit C, denoted $|C|$, is the number of AND and OR gates in the circuit.

2.1 Background on Formulas

A formula φ on n-inputs consists of a rooted binary tree whose leaves are labelled by elements of the set $\{0, 1, x_1, \neg x_1, \ldots, x_n, \neg x_n\}$ and whose internal nodes are labelled by either AND or OR. The *size* of a formula φ, denoted $|\varphi|$, is the number of leaves in its underlying binary tree.

Constant Depth Formulas. For each integer $d \geq 2$, we let AC^0_d denote the class of depth-d formulas. That is, formulas that are allowed to use AND and OR gates of unbounded fan-in, but whose underlying tree has depth at most d. The size of a constant depth formula is again the number of leaves in its underlying tree. We let $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$ and $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ denote the classes of depth-d formulas with an AND and OR top/output gate respectively.

For a function f, we let $\mathsf{L}_d(f)$ denote the size of the smallest depth-d formula computing f. Similarly, we let $\mathsf{L}_d^{\mathsf{AND}}(f)$ and $\mathsf{L}_d^{\mathsf{OR}}(f)$ denote the size of the smallest depth-d formula for computing f that has an AND top gate and OR top gate respectively.

Direct Sums and DeMorgan's Law. We will make heavy use of the following two elementary results about direct sums and negations of functions.

▶ **Proposition 6** (Direct Sum Theorem for Formulas). *Let $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$ be non-constant functions and let $F_\vee : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be given by $F_\vee(x, y) = f(x) \vee g(y)$. Then both of the following hold:*

- $\mathsf{L}_d^{\mathsf{OR}}(F_\vee) = \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d^{\mathsf{OR}}(g)$ *and*
- $\mathsf{L}_d^{\mathsf{AND}}(F_\vee) \geq \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{AND}}(g)$.

Similarly, if $F_\wedge(x, y) = f(x) \wedge g(y)$, then we have

- $\mathsf{L}_d^{\mathsf{OR}}(F_\wedge(x, y)) \geq \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d^{\mathsf{OR}}(g)$ *and*
- $\mathsf{L}_d^{\mathsf{AND}}(F_\wedge(x, y)) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{AND}}(g)$.

Proof. To demonstrate how these are proved, we show why $\mathsf{L}_d^{\mathsf{AND}}(F_\vee) \geq \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{AND}}(g)$. The other lower bounds can be proved similarly, and the upper bounds are easy to see.

Let φ be a $\mathsf{AND} \circ \mathsf{AC}^0_{d-1}$ formula computing $F_\vee$. Since f is not constant there exists an x_1 such that $f(x_1) = 0$. Thus, if we set all the x leaves in φ to x_1 and eliminate the resulting constant leaves using gate elimination, we obtain a formula φ' for computing g

whose size is at most the number of y leaves in φ. Thus, the number of y leaves in φ is at least $\mathsf{L}_d^{\mathsf{AND}}(g)$. Similarly, the number of x-leaves in φ must be at least $\mathsf{L}_d^{\mathsf{AND}}(f)$. Hence, we have that $|\varphi| \geq \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{AND}}(g)$. ◀

The next proposition is a consequence of DeMorgan's Laws.

▶ **Proposition 7** (DeMorgan's Laws).

$$\mathsf{L}_d^{\mathsf{OR}}(\neg f) = \mathsf{L}_d^{\mathsf{AND}}(f)$$

and

$$\mathsf{L}_d^{\mathsf{AND}}(\neg f) = \mathsf{L}_d^{\mathsf{OR}}(f).$$

Finally, we can combine the above two propositions to characterize the complexity of the direct sum of a function with its negation.

▶ **Proposition 8.** *Let f be a function. Let $F_\vee(x, y) = f(x) \vee \neg f(y)$. Let $F_\wedge(x, y) = f(x) \wedge \neg f(y)$. All of the following quantities equal $\mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f)$*

- $\mathsf{L}_d(F_\wedge)$,
- $\mathsf{L}_d(F_\vee)$,
- $\mathsf{L}_d^{\mathsf{AND}}(F_\wedge)$, *and*
- $\mathsf{L}_d^{\mathsf{OR}}(F_\vee)$.

Proof. We just prove that

$$\mathsf{L}_d(F_\wedge) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f).$$

The other proofs are similar. Using the direct sum rules in Proposition 6 and DeMorgan's laws as in Proposition 7 we get that

$$\mathsf{L}_d^{\mathsf{AND}}(F_\wedge) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{AND}}(\neg f) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f).$$

On the other hand, the direct sum rules and DeMorgan's laws also imply that

$$\mathsf{L}_d^{\mathsf{OR}}(F_\wedge) \geq \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d^{\mathsf{OR}}(\neg f) = \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d^{\mathsf{AND}}(f).$$

Together, these imply that

$$\mathsf{L}_d(F_\wedge) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f)$$

as desired. ◀

Non-deterministic formulas and one-sided approximations. A non-deterministic formula φ with n-inputs and m non-deterministic inputs is just a (normal) formula ψ on $(n+m)$-inputs with the last m-inputs being designated as "non-deterministic" inputs. The value of φ on input $x \in \{0,1\}^n$ equals

$$\varphi(x) = \bigvee_{y \in \{0,1\}^m} \psi(x, y).$$

The size of φ, denoted $|\varphi|$ is just the size of ψ.

For our purposes, we will only be interested in non-deterministic formulas that have the same number of regular and non-deterministic inputs. Indeed, for a function $f : \{0,1\}^n \to \{0,1\}$, we let $\mathsf{L}_{\mathsf{ND}}(f)$ denote the size of the smallest non-deterministic formula for computing f with n non-deterministic inputs.

We will also make use of simple bounds on the number of non-deterministic formulas with n regular inputs and n non-deterministic inputs.

▶ **Proposition 9** (Bound on the number of non-deterministic formulas). *The number of functions computed by non-deterministic formulas of size at most s with n-inputs and n non-deterministic inputs is at most*

$$2^{s\log(100n)}.$$

Proof. It suffices to count the number of non-deterministic formulas of size *exactly* s since if a function can be computed by a formula of size less than s, it can clearly also be computed by a formula of size exactly s by adding in gates that do not do anything.

The number of binary trees with s leaves is at most 4^{s+1} by bounds on the Catalan number. Each of the $s-1$ internal nodes can be labeled by either an AND or OR gate, so this gives 2^{s-1} possibilities. Finally the leaf nodes can each be labelled one of $4n+2$ possibilities (either one of the $2n$ variables, the negation of one of the $2n$ variables, or a constant $0,1$). This gives $(4n+2)^s$ possibilities.

In total, this gives us a bound of

$$4^{s+1}2^{s-1}(4n+2)^s = 2^{3s+1}2^{s\log(4n+2)} \leq 2^{4s}2^{s\log(6n)} = 2^{s\log(2^4)+s\log(6n)} \leq 2^{s\log(100n)}$$

where we use that s and n are both at least one. ◀

Finally, if $0 \leq \epsilon \leq 1$, we say a function $g : \{0,1\}^n \to \{0,1\}$ computes an ϵ *one-sided approximation* of a function $f : \{0,1\}^n \to \{0,1\}$ if both of the following conditions hold

- $g^{-1}(1) \subseteq f^{-1}(1)$, and
- $|g^{-1}(1)| \geq \epsilon \cdot |f^{-1}(1)|$.

We let $\mathsf{L}_{\mathsf{ND},\epsilon}(f)$ denote the minimum of $\mathsf{L}_{\mathsf{ND}}(g)$ for all functions g computing an ϵ one-sided approximation of f.

Read Once Formulas. A *read once formula* is a formula where each input variable occurs in at most one leaf. A *monotone read once formula* is a read once formula that reads each input variable positively (i.e., it does not use any negations).

2.2 Versions of MCSP

In this paper, we will mainly consider three versions of MCSP.

MCSP. The *Minimum Circuit Size Problem*, MCSP, is defined as follows:

- **Given:** the truth table $T \in \{0,1\}^{2^n}$ of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an integer size parameter s.
- **Decide:** Does there exists a circuit of size at most s that computes f?

MCSP for $\mathcal{C}$-circuits: ($\mathcal{C}$)-MCSP. The *Minimum $\mathcal{C}$-Circuit Size Problem*, ($\mathcal{C}$)-MCSP, is defined as follows:

- **Given:** the truth table $T \in \{0,1\}^{2^n}$ of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ and an integer size parameter s.
- **Decide:** Does there exists a $\mathcal{C}$-circuit of size at most s that computes f?

MCSP for partial functions: MCSP*. The *Minimum Circuit Size Problem for Partial Functions*, MCSP*, is defined as follows:

- **Given:** the truth table $T \in \{0,1,\star\}^{2^n}$ of a partial Boolean function $\gamma : \{0,1\}^n \to \{0,1,\star\}$ and an integer size parameter s.
- **Decide:** Does there exists a circuit of size at most s that computes γ?

3 ETH Hardness for MCSP*

We will prove hardness for MCSP* by giving a reduction from the $2n \times 2n$ *Bipartite Permutation Independent Set* problem. This problem was introduced by Lokshtanov, Marx, and Saurabh who proved hardness for it under ETH [25]. $2n \times 2n$ Bipartite Permutation Independent Set is defined as follows:

- **Given:** An undirected graph G over the vertex set $[2n] \times [2n]$ where every edge is between the sets of vertices $J_1 = \{(j,k) : j,k \in [n]\}$ and $J_2 = \{(n+j, n+k) : j,k \in [n]\}$.
- **Decide:** Does there exist a permutation $\pi : [2n] \to [2n]$ such that the set

$$\{(1, \pi(1)), \ldots, (2n, \pi(2n))\}$$

is both a subset of $J_1 \cup J_2$ and an independent set of G?

The following definition is equivalent and is easier for us to work with, so it is the one we use throughout the paper.

- **Given:** A directed graph G on the vertex set $[n] \times [n]$ with an edge set E.
- **Decide:** Does there exist a permutation $\pi : [2n] \to [2n]$ such that all of the following are true:
  - $\pi([n]) = [n]$,
  - $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$, and
  - if $((j,k),(j',k')) \in E$, then either $\pi(j) \neq k$ or $\pi(j'+n) \neq k'+n$.

If ETH is true, then this problem cannot be solved much faster than brute forcing over all (roughly $2^{n \log n}$) permutations.

▶ **Theorem 10** (Lokshtanov, Marx, and Saurabh [25])**.** *$2n \times 2n$ Bipartite Permutation Independent Set cannot be solved in deterministic time $2^{o(n \log n)}$ unless ETH fails.*

We prove hardness for MCSP* by giving a reduction from $2n \times 2n$ Bipartite Permutation Independent Set.

▶ **Theorem 11.** MCSP* *cannot be solved in deterministic time $N^{o(\log \log N)}$ on truth tables of length-N assuming ETH. In particular, detecting whether a truth table $T \in \{0,1,\star\}^{2^n}$ can be computed by a monotone read once formula cannot be solved in deterministic time $N^{o(\log \log N)}$ assuming ETH where $n = \log N$.*

Proof. We give a reduction from $2n \times 2n$ Bipartite Permutation Independent Set to MCSP* that runs in deterministic $2^{O(n)}$ time.

Reduction

Before we describe the reduction, we introduce some notation. For an $i \in [n]$, we let $e_i \in \{0,1\}^n$ denote the indicator vector with a one in the ith entry and zeroes everywhere else. Similarly, we let $\overline{e_i} \in \{0,1\}^n$ denote the complementary vector, with a zero in the ith entry and ones everywhere else.

The reduction R works as follows. Given an instance of $2n \times 2n$ Bipartite Permutation Independent Set defined by a directed graph $G = ([n] \times [n], E)$, the reduction outputs the truth table of the partial function $\gamma : \{0,1\}^{2n} \times \{0,1\}^{2n} \times \{0,1\}^{2n} \to \{0,1,\star\}$ given by

$\gamma(x, y, z) =$

$$\begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1, \ldots, x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1}, \ldots, x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ 1 & \text{, if } \exists\ ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}.$$

Running time

It is easy to see that γ is well-defined and that the truth table of γ can be output in time $2^{O(n)}$ given G.

Correctness

We prove the correctness of this reduction in stages, by showing each of the following are equivalent:

1. $\mathsf{MCSP}^\star(\gamma, 6n - 1) = 1$
2. γ can be computed by a read once formula
3. there exists a permutation $\pi : [2n] \to [2n]$ such that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ
4. there exists a permutation $\pi : [2n] \to [2n]$ that satisfies the instance of $2n \times 2n$ Bipartite Permutation Independent Set given by G.

The remainder of the proof is dedicated to proving the equivalences (1) $\iff$ (2), (2) $\iff$ (3), and (3) $\iff$ (4).

(1) $\iff$ (2)

We need to show that $\mathsf{MCSP}^\star(\gamma, 6n - 1) = 1$ if and only if γ can be computed by a read once formula.

This reverse direction is obvious (note that size for circuits equals the number of gates, but size for formulas equals the number of leaves).

The forward direction follows from γ depending on all of its $6n$ distinct input variables. It depends on all its y and z input variables because

$$\gamma(x, y, z) = \bigvee_{i\in[2n]} (y_i \wedge z_i)$$

when $x = 0^{2n}$. It depends on all its x input variables because when $z = 1^{2n}$

$$\gamma(x, y, z) = \bigvee_{i\in[2n]} (x_i \vee y_i).$$

(2) $\iff$ (3)

We need to show that γ can be computed by a read once formula if and only if there exists a permutation $\pi : [2n] \to [2n]$ such that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ.

The reverse direction is obvious. The forward direction follows from the following lemma, whose proof we defer to the end of the section.

▶ **Lemma 12.** *Suppose φ is a read once formula that computes a partial function $\gamma : \{0,1\}^{2n} \times \{0,1\}^{2n} \times \{0,1\}^{2n}$ satisfying*

$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & , \textit{ if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & , \textit{ if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & , \textit{ if } z = 1^{2n} \\ 0 & , \textit{ if } z = 0^{2n} \end{cases}.$$

Then there exists a permutation $\pi : [2n] \to [2n]$ such that $\varphi(x,y,z)$ equals, as a formula, $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$.

Note that our γ actually satisfies more constraints imposed on it than the ones stated in this lemma. For example, we specified $\gamma(x,y,z) = \mathsf{OR}_n(x_1,\ldots,x_n)$ when $(y,z) = (0^{2n}, 1^n0^n)$. But these extra constraints are not needed to prove the lemma.

(3) $\iff$ (4)

We need to show that there exists a permutation $\pi : [2n] \to [2n]$ such that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ if and only if there exists a permutation $\pi : [2n] \to [2n]$ that satisfies the instance of $2n \times 2n$ Bipartite Permutation Independent Set given by G.

The proof of this equivalence is long because there are many conditions to check. We give the full proof below, however, we remark that it essentially amounts to carefully plugging in definitions.

We start with the forward direction. Suppose that $\pi : [2n] \to [2n]$ is a permutation such that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ. We will show that π satisfies the constraints required in $2n \times 2n$ Bipartite Permutation Independent Set. That is, all the following hold

1. $\pi([n]) = [n]$,
2. $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$, and
3. if $((j,k),(j',k')) \in E$, then either $\pi(j) \neq k$ or $\pi(j'+n) \neq k'+n$

The proof that (1) and (2) hold are similar, so we just prove (1). We need to show that if $i \in [n]$, then $\pi(i) \in [n]$. This follows from the following series of equalities when setting $(x,y,z) = (e_i0^n, 0^{2n}, 1^n0^n)$

$$\begin{aligned} 1 &= \mathsf{OR}_n(x_1,\ldots,x_n) \\ &= \gamma(x,y,z) \\ &= \bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i) \\ &= \mathbb{1}_{\pi(i)\in[n]} \end{aligned}$$

where the justifications for these equalities are (in order):

- since $x = e_i0^n$ and $i \in [n]$,
- from the definition of γ when $(x,y,z) = (e_i0^n, 0^{2n}, 1^n0^n)$,
- since $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ, and
- since $(x,y,z) = (e_i0^n, 0^{2n}, 1^n0^n)$

This completes our justification that (1) and (2) hold.

For (3), suppose that $((j,k),(j',k')) \in E$. We need to show that either $\pi(j) \neq k$ or $\pi(j'+n) \neq k'+n$. This follows from the following series of equalities when setting

$(x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'})$

$$
\begin{aligned}
1 &= \gamma(x,y,z) \\
&= \bigvee_{i\in[2n]} ((x_{\pi(i)} \vee y_i) \wedge z_i) \\
&= x_{\pi(j)} \vee x_{\pi(j'+n)} \\
&= \mathbb{1}_{\pi(j)\notin\{k,k'+n\}} \vee \mathbb{1}_{\pi(j'+n)\notin\{k,k'+n\}} \\
&= \mathbb{1}_{\pi(j)\neq k} \vee \mathbb{1}_{\pi(j'+n)\neq k'+n}
\end{aligned}
$$

where the justifications for these equalities are (in order):

- from the definition of γ when $(x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'})$ and $((j,k),(j',k')) \in E$,
- since $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ,
- since $(y,z) = (0^{2n}, e_j e_{j'})$,
- since $x = \overline{e_k e_{k'}}$, and
- since we have already shown that (1) and (2) must hold (i.e, that $\pi([n]) = [n]$ and $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\})$.

This completes our proof of the forward direction.

Now we show the reverse direction. Suppose $\pi : [2n] \to [2n]$ satisfies the constraints in G. In other words, all of the following are true:

- $\pi([n]) = [n]$
- $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$
- if $((j,k),(j',k')) \in E$, then either $\pi(j) \neq k$ or $\pi(j'+n) \neq k'+n$

We will show that $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$ computes γ. In other words, we need to check the following seven cases:

$$\bigvee_{i\in[2n]} ((x_{\pi(i)} \vee y_i) \wedge z_i) =$$

$$
\begin{cases}
\bigvee_{i\in[2n]} (y_i \wedge z_i) & \text{, if } x = 0^{2n} \quad (1)\\
\bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \quad (2)\\
\bigvee_{i\in[2n]} (x_i \vee y_i) & \text{, if } z = 1^{2n} \quad (3)\\
0 & \text{, if } z = 0^{2n} \quad (4)\\
\mathsf{OR}_n(x_1,\ldots,x_n) & \text{, if } z = 1^n 0^n \text{ and } y = 0^{2n} \quad (5)\\
\mathsf{OR}_n(x_{n+1},\ldots,x_{2n}) & \text{, if } z = 0^n 1^n \text{ and } y = 0^{2n} \quad (6)\\
1 & \text{, if } \exists\ ((j,k),(j',k')) \in E \text{ with } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \quad (7)
\end{cases}
$$

The proof in cases (1) - (4) are easy to see. The proof in cases (5) and (6) follow from the fact that $\pi([n]) = [n]$ and $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$.

Lastly, we must check case (7). Suppose that $((j,k),(j',k')) \in E$. When $(x,y,z) =$

$(\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'})$, we have that

$$\begin{aligned}
\bigvee_{i\in[2n]} ((x_{\pi(i)} \vee y_i) \wedge z_i) &= x_{\pi(j)} \vee x_{\pi(j'+n)} \\
&= \mathbb{1}_{\pi(j)\notin\{k,k'+n\}} \vee \mathbb{1}_{\pi(j'+n)\notin\{k,k'+n\}} \\
&= \mathbb{1}_{\pi(j)\neq k} \vee \mathbb{1}_{\pi(j'+n)\neq k'+n} \\
&= 1
\end{aligned}$$

where the justification for each equality is (in order):

- since $y = 0^{2n}$ and $z = e_j e_{j'}$,
- since $x = \overline{e_k e_{k'}}$,
- since $\pi([n]) = [n]$ and $\pi(\{n+i : i \in [n]\}) = \{n+i : i \in [n]\}$, and
- since π satisfies all the constraints of G, we know that for $((j,k),(j',k')) \in E$ either $\pi(j) \neq k$ or $\pi(j'+n) \neq k'+n$

This completes the reverse direction. ◀

We now give the proof of Lemma 12. In this proof, it will be important to distinguish between when two formulas are equal as functions (i.e., they compute the same function) and when they are equal as formulas (i.e., they are isomorphic as labeled binary trees up to the commutativity of AND and OR gates). We will try to be explicit about this by prefacing equalities by "as functions" or "as formulas."

▶ **Lemma 12.** *Suppose φ is a read once formula that computes a partial function $\gamma : \{0,1\}^{2n} \times \{0,1\}^{2n} \times \{0,1\}^{2n}$ satisfying*

$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & , \text{ if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & , \text{ if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & , \text{ if } z = 1^{2n} \\ 0 & , \text{ if } z = 0^{2n} \end{cases}.$$

Then there exists a permutation $\pi : [2n] \to [2n]$ such that $\varphi(x,y,z)$ equals, as a formula, $\bigvee_{i\in[2n]}((x_{\pi(i)} \vee y_i) \wedge z_i)$.

Proof of Lemma 12. We begin by proving three claims about the structure of φ. In Claim 13, we show that φ is a monotone read once formula with $6n$ leaves, and thus $6n-1$ gates. Then, in Claim 14 we show that φ must have $4n-1$ OR gates, and finally, Claim 15 shows that each z variable leaf feeds into an AND gates.

▷ Claim 13. φ reads each x, y, and z input variable exactly once, and it reads each x, y, and z variable positively (i.e. it uses no negated input variables).

Proof. φ is a read once formula so each input variable can be used at most once, so to show that φ reads each input variable exactly once we just need to show that γ depends on every input.

Regarding positivity, in our model of formulas, negations are pushed to the leaf level, so only the monotone gates AND and OR can be used (no NOT gates). Thus, if the read once formula φ read the negated version of an input variable, then its output would have to be monotone in the value of that negated variable.

Now, when $x = 0^{2n}$, $\gamma(x,y,z) = \bigvee_{i\in 2n}(y_i \wedge z_i)$, so γ depends on all its y and z variables. Moreover, the output of $\bigvee_{i\in 2n}(y_i \wedge z_i)$ is monotone in all the y and z variables, so we know that each y and z input cannot be read negatively.

A similar argument can be made for the x variables, by setting $z = 1^{2n}$, in which case $\gamma(x, y, z) = \bigvee_{i \in 2n}(x_i \vee y_i)$. ◁

▷ Claim 14. φ has at least $4n - 1$ OR gates.

Proof. By setting $z = 1^{2n}$ and applying a standard gate elimination argument, one can eliminate gates in φ to obtain a read once formula ψ for computing $\bigvee_{i \in [2n]}(x_i \vee y_i)$ with $4n$ leaves and $4n - 1$ gates. It is easy to see that all $4n - 1$ of the gates in ψ must be OR gates. As a result, these $4n - 1$ OR gates must also be in φ. ◁

▷ Claim 15. For each $i \in [2n]$, the z_i leaf in φ feeds into an AND gate.

Proof. Fix some $i \in [2n]$. From Claim 13, we know that z_i is read exactly once, positively in the formula φ. If, for contradiction, the z_i leaf fed into an OR gate, then by setting $z_i = 1$ and applying a standard gate elimination argument, we could obtain a formula ψ with $6s - 2$ leaves for computing $\gamma(x, y, z)$ when $z_i = 1$.

This is a contradiction because $\gamma(x, y, z)$ depends on $6n-1$ of its inputs even when $z_i = 1$. In particular, $\gamma(x, y, 1^{2n}) = \bigvee_{j \in [2n]}(x_j \vee y_j)$, so it depends on all $4n$ of its x and y inputs. And $\gamma(0^{2n}, y, z) = \bigvee_{j \in [2n]}(y_i \wedge z_i)$ so it depends on the remaining $2n - 1$ of its z inputs. ◁

Now, we introduce some important subformulas of φ. For each $i \in [2n]$, let φ_i be the subformula of φ such that $z_i \wedge \varphi_i$ is a subformula of φ. Crucially, Claim 16 shows that $\varphi_1, \ldots, \varphi_{2n}$ all do not read any z inputs.

▷ Claim 16. For each $i \in [2n]$, the formula φ_i does not read any z input leaf.

Proof. Since $z_i \wedge \varphi_i$ is a subformula of φ and φ is a read once formula, we know that no z_i leaf occurs in φ_i.

Next, consider some $i' \in [n] \setminus \{i\}$. For contradiction, suppose φ_i read the $z_{i'}$ input. Then the output of the read once formula φ could not depend on the input $z_{i'}$ when $z_i = 0$ (since the read once property implies that the only time φ reads the input $z_{i'}$ is in the subformula $z_i \wedge \varphi_i(x, y, z)$, which always evaluates to zero when $z_i = 0$). But when $x = 0^{2n}$ and $z_i = 0$, $\varphi(x, y, z) = \bigvee_{j \in [2n]}(y_j \wedge z_j)$, so the output of φ does still depend on $z_{i'}$ when $z_i = 0$, giving us a contradiction. ◁

The key consequence of Claim 16 is that it means the subformulas $\varphi_1 \wedge z_1, \ldots, \varphi_{2n} \wedge z_{2n}$ are all disjoint subformulas of φ (since none of the φ_i can read a z variable). This implies that φ contains $2n$ AND gates. Since we already knew that there were $4n - 1$ OR gates in φ (by Claim 14) and $6n - 1$ gates total (by Claim 13), this means the only AND gates in φ are the $2n$ AND gates at the top of the subformulas $\varphi_1 \wedge z_1, \ldots, \varphi_{2n} \wedge z_{2n}$. Using this, along with the knowledge from Claim 13 that φ reads every input positively, we get that as a formula,

$$\varphi = (\bigvee_{w \in I} w) \vee (\bigvee_{i \in [2n]} (z_i \wedge \varphi_i(x, y, z)))$$

where I is some subset of the x and y input variables (i.e., $I \subseteq \{x_1, \ldots, x_{2n}, y_1, \ldots, y_{2n}\}$).

In fact, I must actually be empty!

▷ Claim 17. $I = \emptyset$.

Proof. When $z = 0^{2n}$, we have that

$$0 = \varphi(x, y, z) = (\bigvee_{w \in I} w) \vee \bigvee_{i \in [2n]} (z_i \wedge \varphi_i(x, y, z)) = \bigvee_{w \in I} w.$$

◁

So now, we know that, as a formula, we have that

$$\varphi = \bigvee_{i \in [2n]} (z_i \wedge \varphi_i(x, y, z)).$$

Next, we use the fact that φ_i can only use OR gates (since all the AND gates in φ are already accounted for). In particular, this, combined with the fact that φ is a monotone read once formula (by Claim 13), implies there exists pairwise disjoint subsets $I_1, \ldots, I_{2n}$ of $\{x_1, \ldots, x_{2n}, y_1, \ldots, y_{2n}\}$ such that, as a formula,

$$\varphi = \bigvee_{i \in [2n]} (z \wedge (\bigvee_{w \in I_i} w)).$$

Therefore, when $x = 0^{2n}$, we have that, as functions,

$$\bigvee_{i \in [2n]} (y_i \wedge z_i) = \gamma(x, y, z) = \varphi(x, y, z) = \bigvee_{i \in [2n]} (z_i \wedge (\bigvee_{w \in I_i} w)).$$

From this equality, it is easy to see that we must have $y_i \in I_i$ for all $i \in [2n]$.

As a result, we can conclude that, as a formula,

$$\varphi = \bigvee_{i \in [2n]} (z_i \wedge (y_i \vee \bigvee_{w \in J_i} w))$$

where $J_1, \ldots, J_{2n}$ are pairwise disjoint subsets of $\{x_1, \ldots, x_{2n}\}$.

Finally, when $x = 1^{2n}$, we have that, as a function,

$$\bigvee_{i \in [2n]} (z_i \wedge (y_i \vee \bigvee_{w \in J_i} w)) = \varphi(x, y, z) = \gamma(x, y, z) = \bigvee_{i \in [2n]} z_i.$$

From this we can conclude that there is a permutation $\pi : [2n] \to [2n]$ such that, as a formula,

$$\varphi = \bigvee_{i \in [2n]} (z_i \wedge (y_i \vee x_{\pi(i)}))$$

which is what we desired to show. ◀

4 Main Lower Bound for Constant Depth Formulas: From Depth d to $d + 1$

In this section we prove our main constant depth formula lower bound.

▶ **Theorem 5.** *Let $d \geq 3$. Let $\gamma = \frac{1}{10^4}$. Let $f : \{0,1\}^n \to \{0,1\}$ be a non-constant function, and let $g : \{0,1\}^m \to \{0,1\}$ be a non-constant function with $m \geq n$ that satisfies*

$$\min\{2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g), \mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g)\} \geq \mathsf{L}^{\mathsf{OR}}_d(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f).$$

Then

$$\mathsf{L}^{\mathsf{OR}}_d(f(x) \wedge g(y)) \geq \mathsf{L}^{\mathsf{OR}}_d(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f).$$

Proof. For convenience, let $F : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be given by $F(x, y) = f(x) \wedge g(y)$.

For contradiction, suppose there is the a $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula φ for computing F of size less than $\mathsf{L}^{\mathsf{OR}}_d(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f)$. We assume without loss of generality that φ alternates between

OR and AND gates at each level, and thus we can write $\varphi = \bigvee_{i\in[t]} \varphi_i$ where each φ_i is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula.

For each $i \in [t]$, let the set $S_i \subseteq \{0,1\}^m$ denote the set of y-inputs φ_i accepts when using the x-inputs non-deterministically. In other words,

$$S_i = \{y \in \{0,1\}^m : \bigvee_{x\in\{0,1\}^n} \varphi_i(x,y) = 1\}.$$

Since φ computes $F(x,y) = f(x) \wedge g(y)$, it is not too hard to see that the union of the S_i sets is exactly the set of YES instances of g.

▷ Claim 18. $\bigcup_{i\in[t]} S_i = g^{-1}(1)$.

Proof. First, we show that $\bigcup_{i\in[t]} S_i \subseteq g^{-1}(1)$. If $y \in S_i$ for some $i \in [t]$, then there exists some x such that $\varphi_i(x,y) = 1$. Thus we have that

$$f(x) \wedge g(y) = F(x,y) = \varphi(x,y) = \bigvee_{i\in[t]} \varphi_i(x,y) = 1$$

so $g(y) = 1$, so $y \in g^{-1}(1)$.

For the other direction, suppose that $y \in g^{-1}(1)$. Since f is not constant, there exists some x such that $f(x) = 1$. Then

$$1 = f(x) \wedge g(y) = F(x,y) = \varphi(x,y) = \bigvee_{i\in[t]} \varphi_i(x,y)$$

so there exists some $i \in [t]$ such that $\varphi_i(x,y) = 1$ so $y \in S_i$. ◁

However, an even stronger claim is true. Not only do the sets $S_1, \ldots, S_t$ cover $g^{-1}(1)$, but they must actually cover $g^{-1}(1)$ in a "redundant" way, which we make formal in the following claim.

▷ Claim 19. Each $y \in g^{-1}(1)$ is an element of at least two distinct sets in the list $S_1, \ldots, S_t$.

Proof. For contradiction, suppose not. Since we know that $g^{-1}(1) = \bigcup_{i\in[t]} S_i$ from Claim 18, it follows that there exists some $y_1 \in g^{-1}(1)$ such that y_1 is in exactly one of the sets in the list $S_1, \ldots, S_t$.

Without loss of generality, assume that y_1 is only in the set S_1. By definition, this means that $\varphi_i(x, y_1) = 0$ for all $i \geq 2$ and all $x \in \{0,1\}^n$. As a result, we have the following equality for all $x \in \{0,1\}^n$

$$f(x) = f(x) \wedge 1 = f(x) \wedge g(y_1) = F(x,y_1) = \bigvee_{i\in[t]} \varphi_i(x,y_1) = \varphi_1(x,y_1).$$

Hence, φ_1 can be made into an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula for f by fixing its y-inputs to y_1. This implies that φ_1 has at least $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ many x-leaves.

Clearly, this means that φ also has at least $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ many x-leaves. On the other hand, since f is non-constant, there exists an x_1 such that $f(x_1) = 1$. Thus, if we set the x-inputs of φ to be x_1, we have that $\varphi(x_1, y)$ computes $g(y)$. Hence, g has at least $\mathsf{L}^{\mathsf{OR}}_{d}(g)$ many y-leaves.

Summing the bound on the x-leaves and the y-leaves, we get that

$$|\varphi| \geq \mathsf{L}^{\mathsf{OR}}_{d}(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f)$$

which contradicts our supposition that $|\varphi| < \mathsf{L}^{\mathsf{OR}}_{d}(g) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f)$. ◁

We can use this "redundancy" to show that each of the S_i sets must be "small." This is roughly because the redundancy implies that even if you remove any one of the φ_i from φ, what remains can be used to make a non-deterministic formula for g and thus, still has most of the "cost" of computing g within it.

▷ Claim 20. For all $i \in [t]$, we have $|S_i| \leq \gamma \cdot |g^{-1}(1)|$.

Proof. For contradiction, suppose that $|S_i| > \gamma \cdot |g^{-1}(1)|$ for some $i \in [t]$. This implies that, viewing the x-inputs to φ_i non-deterministically, φ_i yields a non-deterministic one-sided γ-approximation of g, so

$$|\varphi_i| \geq \mathsf{L}_{\mathsf{ND},\gamma}(g).$$

On the other hand, since $\bigcup_{j\in[t]} S_j = g^{-1}(1)$ from Claim 18 and since each element of $g^{-1}(1)$ is contained in two sets in the list $S_1, \ldots, S_t$ by Claim 19, we know that

$$\bigcup_{j\in[t]\setminus\{i\}} S_j = g^{-1}(1).$$

From the definition of $S_1, \ldots, S_t$, this implies that

$$\bigvee_{j\in[t]\setminus\{i\}} \varphi_j$$

is a non-deterministic formula for g, viewing the x-inputs non-deterministically. Hence,

$$\sum_{j\in[t]\setminus\{i\}} |\varphi_j| \geq \mathsf{L}_{\mathsf{ND}}(g).$$

Thus, putting these two bounds together, we have that

$$|\varphi| = |\varphi_i| + \sum_{j\in[t]\setminus\{i\}} |\varphi_j| \geq \mathsf{L}_{\mathsf{ND},\gamma}(g) + \mathsf{L}_{\mathsf{ND}}(g).$$

However, an assumption in the theorem statement is that $\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g) \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$, so we have that

$$|\varphi| \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$$

which contradicts our supposition that $|\varphi| < \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$. ◁

We can then use the fact that the sets $S_1, \ldots, S_t$ have small cardinality and the fact that they form a "redundant" cover of $g^{-1}(1)$ in order to argue that we can partition the list of sets $S_1, \ldots, S_t$ into two disjoint lists that each covers a significant portion of $g^{-1}(1)$. This is made formal in the following claim.

▷ Claim 21. There exist disjoint subsets $L, R \subseteq [t]$ such that for all $T \in \{L, R\}$,

$$|\bigcup_{i\in T} S_i| \geq .73|g^{-1}(1)|.$$

Before proving Claim 21, we show how we can finish the proof using the claim. Let L and R be sets satisfying the claim. For $T \in \{L, R\}$, define the $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formula φ_T given by

$$\varphi_T = \bigvee_{i\in T} \varphi_i.$$

Since for each $T \in \{L, R\}$, we have that

$$|\bigcup_{i \in T} S_i| \geq .73|g^{-1}(1)|,$$

we know that φ_T is a non-deterministic .73-one-sided approximation for g. Hence for all $T \in \{L, R\}$, we have that $|\varphi_T| \geq \mathsf{L}_{\mathsf{ND},.73}(g)$.

Since L and R are disjoint, we have that

$$|\varphi| \geq |\varphi_L| + |\varphi_R| \geq 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g) \geq \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$$

which contradicts our supposition that $|\varphi| < \mathsf{L}_d^{\mathsf{OR}}(g) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$.

It remains to prove Claim 21.

Proof of Claim 21. We prove this using the probabilistic method. For each element $i \in [t]$, flip an independent, unbiased coin to decide whether i should be placed in L or in R. We will argue that this yields a disjoint L and R pair with the desired properties with positive probability using the second moment method.

We will now show that

$$\Pr_L[|\bigcup_{i \in L} S_i| \geq .73|g^{-1}(1)|] \geq \frac{2}{3}.$$

Assuming this is true, we know by symmetry that

$$\Pr_R[|\bigcup_{i \in R} S_i| \geq .73|g^{-1}(1)|] \geq \frac{2}{3}$$

and so by a union bound it follows that

$$\Pr_{L,R}[|\bigcup_{i \in L} S_i| \geq .73|g^{-1}(1)| \text{ AND } |\bigcup_{i \in R} S_i| \geq .73|g^{-1}(1)|] > 0$$

which is what we desired to prove (note that L and R are disjoint by construction).

Hence, it suffices to prove that

$$\Pr_L[|\bigcup_{i \in L} S_i| \geq .73|g^{-1}(1)|] \geq \frac{2}{3}.$$

For simplicity, let X denote the random variable $|\bigcup_{i \in L} S_i|$ and for each $y \in g^{-1}(1)$, let X_y denote the indicator random variable for the event that $y \in \bigcup_{i \in L} S_i$. Then using linearity of expectation we have that

$$\begin{aligned}
\mathbb{E}[X] &= \mathbb{E}[\sum_{y \in g^{-1}(1)} X_y] \\
&= \sum_{y \in g^{-1}(1)} \mathbb{E}[X_y] \\
&= \sum_{y \in g^{-1}(1)} (1 - 2^{-|\{i \in [t] : y \in S_i\}|}) \\
&\geq \sum_{y \in g^{-1}(1)} (1 - 2^{-2}) \\
&= \frac{3}{4}|g^{-1}(1)|.
\end{aligned}$$

where the inequality follows from the fact that each $y \in g^{-1}(1)$ lies in two at least two distinct sets in the list $S_1, \ldots, S_t$ as proved in Claim 19.

Thus, Chebyshev's inequality the implies that

$$\Pr[X \le .73|g^{-1}(1)|] \le \Pr[|X - \mathbb{E}[X]| \ge .02|g^{-1}(1)|] \le \frac{\mathsf{Var}[X]}{(.02|g^{-1}(1)|)^2}$$

Thus, if we could show that $\frac{\mathsf{Var}[X]}{(.02|g^{-1}(1)|)^2} \le \frac{1}{3}$, then we would have that

$$\Pr[X \le .73|g^{-1}(1)|] \le \frac{1}{3}$$

as desired.

We now show that $\frac{\mathsf{Var}[X]}{(.02|g^{-1}(1)|)^2} \le \frac{1}{3}$, or equivalently, that

$$\mathsf{Var}[X] \le \frac{4}{3 \cdot 10^4}|g^{-1}(1)|^2.$$

Using the fact that $X = \sum_{y \in g^{-1}(1)} X_y$, we have that

$$\mathsf{Var}[X] = \sum_{y,y' \in g^{-1}(1)} \mathsf{Cov}[X_y, X_{y'}]$$

Now fix some $y \in g^{-1}(1)$, and we will bound $\sum_{y' \in g^{-1}(1)} \mathsf{Cov}[X_y, X_{y'}]$. Let $D_y = \{y' : \exists i \in [t]$ such that $\{y, y'\} \subseteq S_i\}$. Note that if $y' \notin D_y$, then y' and y never appear in any set S_i together, and hence X_y and X'_y are independent random variables. Thus,

$$\sum_{y' \in g^{-1}(1)} \mathsf{Cov}[X_y, X_{y'}] = \sum_{y' \in D_y} \mathsf{Cov}[X_y, X_{y'}].$$

Since $|S_i| \le \gamma|g^{-1}(1)|$ for all $i \in [t]$ by Claim 20, it follows that

$$|\{i \in [t] : y \in S_i\}| \ge \frac{|D_y|}{\gamma|g^{-1}(1)|}$$

which implies that

$$\mathbb{E}[X_y] \ge 1 - 2^{-\frac{|D_y|}{\gamma|g^{-1}(1)|}}.$$

Hence,

$$\begin{aligned}
\sum_{y' \in D_y} \mathsf{Cov}[X_y, X_{y'}] &= \sum_{y' \in D_y} \mathsf{Cov}[X_y, X_{y'}] \\
&= \sum_{y' \in D_y} \mathbb{E}[X_y X_{y'}] - \mathbb{E}[X_y]\,\mathbb{E}[X_{y'}] \\
&\le \sum_{y' \in D_y} \mathbb{E}[X_{y'}] - \mathbb{E}[X_y]\,\mathbb{E}[X_{y'}] \\
&\le \sum_{y' \in D_y} \mathbb{E}[X_{y'}] - (1 - 2^{-\frac{|D_y|}{\gamma|g^{-1}(1)|}})\,\mathbb{E}[X_{y'}] \\
&\le |D_y| 2^{-\frac{|D_y|}{\gamma|g^{-1}(1)|}} \\
&\le \frac{\gamma|g^{-1}(1)|}{\ln 2} 2^{-\frac{1}{\ln 2}} \\
&\le \gamma|g^{-1}(1)|
\end{aligned}$$

where the second to last inequality follows from some calculus.

Hence, we have that

$$\mathsf{Var}[X] = \sum_{y,y' \in g^{-1}(1)} \mathsf{Cov}[X_y, X_{y'}] \leq \gamma |g^{-1}(1)|^2 \leq \frac{4}{3 \cdot 10^4} |g^{-1}(1)|^2$$

since $\gamma = \frac{1}{10^4}$.

◁

◀

5 (AC^0_d)-MCSP is NP-hard

We use the lower bound technique in Theorem 5 to prove hardness for constant depth formula minimization.

▶ **Theorem 22.** *Let $d \geq 2$ be an integer. Then there exists an $\alpha_d > 0$ such that computing $\mathsf{L}_d(\cdot)$ up to a factor of $(1 + \alpha_d)$ is* NP*-complete under randomized quasipolynomial Turing reductions.*

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

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

Each of these parts correspond to the following three theorems (in order).

▶ **Theorem 23.** *Let $d \geq 2$ be an integer. Let $\alpha \geq 0$. Given access to an oracle $\mathcal{O}$ that computes an $(1+\alpha)$ multiplicative approximation to L_d and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}^{\mathsf{OR}}_d(f)$ and $\mathsf{L}^{\mathsf{AND}}_d(f)$ up to a factor of $(1 + \alpha)^2$ in deterministic quasipolynomial time.*

▶ **Corollary 24** (Easy corollary of Khot and Saket [23])**.** *Given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, determining $\mathsf{L}^{\mathsf{OR}}_2(f)$ up to a factor of $n^{1-\epsilon}$ is* NP*-hard under quasipolynomial time Turing reductions for arbitrarily small $\epsilon > 0$.*

We note that [23] actually proves the NP-hardness of $\mathsf{L}^{\mathsf{OR}}_2$ when the size of a DNF is the number of *terms* in the DNF rather than the number of leaves. However, there is an easy reduction between computing these two size measures, which we show in Section 7.

▶ **Theorem 25.** *Let $d \geq 3$. Let $0 < \alpha < 10^{-7}$. Given access to an oracle $\mathcal{O}$ that computes $\mathsf{L}^{\mathsf{OR}}_d$ up to a factor of $(1+\alpha)$ and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}^{\mathsf{OR}}_{d-1}(f)$ up to a $(1 + O(\alpha))$ factor in randomized quasipolynomial time.*

In the next three sections, we prove these theorems in reverse order. We finish this section by showing that these three parts together imply Theorem 22.

Proof of Theorem 22. The reduction from computing $\mathsf{L}_d^{\mathsf{OR}}$ to computing L_d in Theorem 23 implies that it suffices to show that that for each $d \geq 2$ there exists some $\alpha_d > 0$ such that computing $\mathsf{L}_d^{\mathsf{OR}}(f)$ up to a factor of $(1+\alpha_d)$ is NP-hard under randomized quasipolynomial Turing reductions.

We show this is indeed the case by induction on d. The base case of $d = 2$ is provided by Corollary 24. Next suppose $d \geq 3$ and that computing $\mathsf{L}_{d-1}^{\mathsf{OR}}(f)$ up to a factor of $(1+\alpha_{d-1})$ is NP-hard under randomized quasipolynomial Turing reductions. Then Theorem 25 implies that there exists an $\alpha_d > 0$ such that computing $\mathsf{L}_d^{\mathsf{OR}}(f)$ up to a factor of $(1+\alpha_d)$ is NP-hard under quasipolynomial time randomized Turing reductions. ◀

6 Approximating $\mathsf{L}_{d-1}^{\mathsf{OR}}(f)$ Using $\mathsf{L}_d^{\mathsf{OR}}(\cdot)$

In this section, we prove Theorem 25.

▶ **Theorem 25.** *Let $d \geq 3$. Let $0 < \alpha < 10^{-7}$. Given access to an oracle $\mathcal{O}$ that computes $\mathsf{L}_d^{\mathsf{OR}}$ up to a factor of $(1+\alpha)$ and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}_{d-1}^{\mathsf{OR}}(f)$ up to a $(1+O(\alpha))$ factor in randomized quasipolynomial time.*

Before proving Theorem 25, we state the following lemma that will be an important ingredient in our proof. This lemma essentially shows that we can sample functions whose CNF complexity is within a certain range and whose non-deterministic formula complexity is very close to its CNF complexity.

▶ **Lemma 26.** *Let $\gamma = 10^{-4}$. Let $0 < \delta < \frac{\gamma}{16}$ be a parameter such that $\frac{1}{\delta} \in \mathbb{N}$. Let n and t be positive integers satisfying $n^{\frac{8}{\delta}} \leq t \leq 2^n$. Then there exists a distribution $\mathcal{D}_{n,t,\delta}$ of Boolean functions with $(n + n^{2/\delta})$-inputs samplable in time quasipolynomial in 2^n such that if $g \leftarrow \mathcal{D}_{n,t,\delta}$, then with probability $1 - o_\delta(1)$ all of the following hold*

1. $(1-4\delta)tn^2 \leq \mathsf{L}_{\mathsf{ND}}(g) \leq \mathsf{L}_2^{\mathsf{AND}}(g) \leq (1+4\delta)tn^2$,
2. $\min\{\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g)\} \geq (1+\frac{\gamma}{2})tn^2$.

In one sentence, Lemma 26 is proved using a counting argument. We defer the prove of Lemma 26 to the end of this section.

Assuming Lemma 26 is true, we can prove Theorem 25.

Proof of Theorem 25. Assume that the oracle $\mathcal{O}$ satisfies

$$\mathsf{L}_d^{\mathsf{OR}}(g) \leq \mathcal{O}(g) \leq (1+\alpha) \cdot \mathsf{L}_d^{\mathsf{OR}}(g)$$

for all functions g.

Next, we note it suffices to show that one can compute $\mathsf{L}_{d-1}^{\mathsf{AND}}(f)$ up to a $(1+O(\alpha))$ factor in quasipolynomial time since, as mentioned in Proposition 7, DeMorgan's laws imply that $\mathsf{L}_{d-1}^{\mathsf{OR}}(f) = \mathsf{L}_{d-1}^{\mathsf{AND}}(\neg f)$.

Let $0 < \delta < \frac{\gamma}{16}$ with $\frac{1}{\delta} \in \mathbb{N}$ be some sufficiently small constant depending on α.

Algorithm for the reduction.

Given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, our algorithm for computing an approximation to $\mathsf{L}_{d-1}^{\mathsf{AND}}(f)$ is as follows. First, using brute force, iterate through all $\mathsf{AND} \circ \mathsf{AC}_{d-2}^0$ formulas of size $n^{1024/\delta}$, see if any of them compute f, and output the size of the smallest one computing f if one does.

Otherwise, for each $i \in [2^{2n}]$ and for each positive integer t satisfying $n^{8/\delta} \leq t \leq 2^n$, sample $g_{i,t} \leftarrow \mathcal{D}_{n,t,\delta}$, and set

$$b_{i,t} = \begin{cases} 1 & \text{, if } \mathcal{O}(f(x) \wedge g_{i,t}(y)) \geq (1 + \frac{\gamma}{16})tn^2 \\ 0 & \text{, otherwise.} \end{cases}$$

Finally, after we have finished computing $b_{i,t}$ for all $i \in [2^{2n}]$ and all $n^{8/\delta} \leq t \leq 2^n$, set

$$t^\star = \max_t \{t : \text{for at least half of } i \in [2^{2n}],\ b_{i,t} = 1\},$$

let $i^\star$ be a random element of $[2^{2n}]$ and output

$$\mathcal{O}(f(x) \wedge g_{i^\star,t^\star}(y)) - t^\star \cdot n^2.$$

This completes our description of the algorithm.

Running Time.

Next, we check that this algorithm runs in quasipolynomial time. By Proposition 9, the number of formulas of size at most $n^{\frac{1024}{\delta}}$ with n-inputs is bounded by

$$2^{n^{\frac{1024}{\delta}} \log(100n)}$$

and is thus quasipolynomial in $N = 2^n$. Thus, we can iterate through all $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formulas of size at most $n^{\frac{1024}{\delta}}$ by iterating through all the unrestricted formulas of size $n^{\frac{1024}{\delta}}$ and checking whether each unrestricted formula is an $\mathsf{AND} \circ \mathsf{AC}^0_{d-2}$ formula (by turning repeated gates into a single gate with larger fan-in). Thus, the brute-force part of the algorithm runs in quasipolynomial time.

For the remaining part of the algorithm, it is easy to see it runs in quasipolynomial time as long as the truth table of each $g_{i,t}$ is quasipolynomial in the length of the truth table of f. Since from Lemma 26 we know that $g_{i,t}$ takes $n + n^{2/\delta}$ inputs, it follows that the length of the truth table of each $g_{i,t}$ is $2^{n+n^{2/\delta}}$ which is quasipolynomial in 2^n, as desired. This completes our analysis of the running time of the algorithm.

Correctness.

We now prove that the algorithm outputs a $(1+O(\alpha))$ approximation to $\mathsf{L}^{\mathsf{AND}}_{d-1}$ with probability at least $2/3$ when n is sufficiently large. Clearly, brute-force stage of the algorithm ensures that the algorithm outputs the $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ exactly when $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq n^{\frac{1024}{\delta}}$. Thus, for the rest of the analysis we can assume that $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \geq n^{\frac{1024}{\delta}}$.

Conditioning on a likely event.

To begin, we will condition on an event that occurs with probability at least two thirds, which we describe next. For any $i \in [2^{2n}]$ and any t satisfying $n^{8/\delta} \leq t \leq 2^n$, we say that $g_{i,t}$ is *good* if it satisfies all the conditions at the end of Lemma 26, that is, if the following two statements are true:

1. $(1 - 4\delta)tn^2 \leq \mathsf{L}_{\mathsf{ND}}(g_{i,t}) \leq \mathsf{L}^{\mathsf{AND}}_2(g_{i,t}) \leq (1 + 4\delta)tn^2$, and
2. $\min\{\mathsf{L}_{\mathsf{ND}}(g_{i,t}) + \mathsf{L}_{\mathsf{ND},\gamma}(g_{i,t}), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g_{i,t})\} \geq (1 + \frac{\gamma}{2})tn^2$.

We will condition on the event E that for each fixed t we have that $g_{i,t}$ is good for at least 90% of the $i \in [2^{2n}]$ and $g_{i^\star,t^\star}$ is good. We show that this event occurs with high probability.

▷ Claim 27. E occurs with probability at least $2/3$.

Proof. We do this by a union bound argument.

Fix some $t \in [2^n]$ satisfying $n^{8/\delta} \le t \le 2^n$. We bound the probability that $g_{i,t}$ is good for less than a .9 fraction of the $i \in [2^{2n}]$. Lemma 26 implies that for each fixed i that $g_{i,t}$ is good with probability $1 - o_\delta(1)$. Thus, since each $g_{i,t}$ is sampled independently, we get by a Chernoff bound that

$$Pr[\sum_{i \in [2^{2n}]} \mathbb{1}_{g_{i,t}} \le .9 \cdot 2^{2n}] \le e^{-\Omega_\delta(2^{2n})}.$$

Thus, union bounding over all $t \in [2^n]$, we get that for each fixed t, $g_{i,t}$ is good for 90% of all i with probability at least

$$1 - o_\delta(1) + 2^n \cdot e^{-\Omega_\delta(2^{2n})} = 1 - o_\delta(1).$$

This event also implies that $g_{i^\star,t^\star}$ is good with probability at least 90% since $i^\star$ is chosen at random. Hence, we have that E occurs with probability at least $2/3$ by choosing δ sufficiently small. ◁

For the remainder of the proof, we assume that E occurs.

Lower bounding $t^\star$.

Next, we work to lower bound the value of $t^\star$.

▷ Claim 28. If $g_{i,t}$ is good and $\frac{\gamma}{8}tn^2 \le \mathsf{L}_{d-1}^{\mathsf{AND}}(f) \le \frac{\gamma}{4}tn^2$, then $b_{i,t} = 1$.

Proof of Claim. We wish to use the lower bound that

$$\mathsf{L}_d^{\mathsf{OR}}(f(x) \wedge g_{i,t}(y)) \ge \mathsf{L}_d^{\mathsf{OR}}(g_{i,t}) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f)$$

that is given in Theorem 5. If we could use this lower bound, then we would have that

$$\begin{aligned} \mathcal{O}(f(x) \wedge g_{i,t}(y)) &\ge \mathsf{L}_d^{\mathsf{OR}}(f(x) \wedge g_{i,t}(y)) \\ &\ge \mathsf{L}_d^{\mathsf{OR}}(g_{i,t}) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f) \\ &\ge (1-4\delta)tn^2 + \frac{\gamma}{8}tn^2 \\ &\ge (1+\frac{\gamma}{16})tn^2 \end{aligned}$$

where the first inequality comes from $\mathcal{O}$ being a multiplication approximation of $\mathsf{L}_d^{\mathsf{OR}}$, the second inequality comes the lower bound in Theorem 5, the third inequality comes from the fact $g_{i,t}$ is good and the hypothesis of the claim, and the last inequality comes from setting δ so that $4 \cdot \delta \le \frac{\gamma}{16}$. Thus, since $\mathcal{O}(f(x) \wedge g_{i,t}(y)) \ge (1 + \frac{\gamma}{16})tn^2$, we know that $b_{i,t} = 1$ (by definition) and the claim is proved.

Hence, to prove the claim, we just need to check that the hypotheses in Theorem 5 hold. That is, we need to check that f and g are not constant functions and that

$$\min\{\mathsf{L}_{\mathsf{ND}}(g_{i,t}) + \mathsf{L}_{\mathsf{ND},\gamma}(g_{i,t}), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g_{i,t})\} \ge \mathsf{L}_d^{\mathsf{OR}}(g_{i,t}) + \mathsf{L}_{d-1}^{\mathsf{AND}}(f).$$

Since, after the brute force stage of the algorithm, we know that $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \geq n^{\frac{1024}{\delta}}$, it follows that f is not a constant function. Similarly, since $g_{i,t}$ is good, we know that $\mathsf{L}_{\mathsf{ND}}(g_{i,t}) \geq (1-4\delta)tn^2$, so g is not constant either.

For the last condition, we have that

$$\mathsf{L}^{\mathsf{OR}}_{d}(g_{i,t}) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq (1+4\cdot\delta)tn^2 + \frac{\gamma}{4}tn^2 \leq \min\{\mathsf{L}_{\mathsf{ND}}(g_{i,t}) + \mathsf{L}_{\mathsf{ND},\gamma}(g_{i,t}), 2\cdot \mathsf{L}_{\mathsf{ND},.73}(g_{i,t})\}$$

where the first inequality comes from property (1) of $g_{i,t}$ being good and the assumption in the claim on $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ and the last inequality comes from property (2) of $g_{i,t}$ being good and setting δ so that $4\delta \leq \gamma/4$. ◁

We use Claim 28 to show that $t^\star$ exists and to lower bound $t^\star$ in terms of $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$. In particular, since we know that

$$n^{\frac{1024}{\delta}} \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq n2^n$$

(where the lower bound comes from the brute force stage of the algorithm and the upper bound is the trivial CNF upper bound), it follows that when n is sufficiently large that there exists an integer t satisfying both that

$$n^{8/\delta} \leq t \leq 2^n$$

and that

$$\frac{\gamma}{8}tn^2 \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq \frac{\gamma}{4}tn^2.$$

Hence, using Claim 28 and the fact that E occurs, we get that $t^\star$ exists and $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq \frac{\gamma}{4}t^\star n^2$ when n is sufficiently large.

Upper bounding $t^\star$.

On the other hand the following claim implies that $t^\star$ cannot be too large.

▷ Claim 29. If for some i $g_{i,t}$ is good and $b_{i,t} = 1$ and n is sufficiently large, then $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \geq (\frac{\gamma}{16} - 5\alpha)tn^2$.

Proof of Claim. Since $b_{i,t} = 1$, we have that

$$(1+\frac{\gamma}{16})tn^2 \leq \mathcal{O}(f(x) \wedge g_{i,t}(y)) \leq (1+\alpha)\mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g_{i,t}(y)).$$

On the other hand,

$$\mathsf{L}^{\mathsf{OR}}_{d}(f(x)\wedge g_{i,t}(y)) \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f(x)\wedge g_{i,t}(y)) \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{AND}}_{d-1}(g_{i,t}) \leq (1+4\delta)tn^2 + \mathsf{L}^{\mathsf{AND}}_{d-1}(f)$$

where the last inequality comes from property (1) of $g_{i,t}$ being good (note $d \geq 3$). Putting these two bounds together, we get that

$$\begin{aligned}
\mathsf{L}^{\mathsf{AND}}_{d-1}(f) &\geq \frac{1}{(1+\alpha)}(1+\frac{\gamma}{16})tn^2 - (1+4\delta)tn^2 \\
&\geq (1-2\alpha)(1+\frac{\gamma}{16})tn^2 - (1+4\delta)tn^2 \\
&\geq (1+\frac{\gamma}{16} - 4\alpha)tn^2 - (1+4\delta)tn^2 \\
&\geq (\frac{\gamma}{16} - 4\alpha - 4\delta)tn^2 \\
&\geq (\frac{\gamma}{16} - 5\alpha)tn^2
\end{aligned}$$

where the first inequality comes from $\frac{1}{1+\alpha} \geq 1 - 2\alpha$ when $\alpha \leq 1$, the second inequality comes from $\gamma < 1$, and the last inequality comes from assuming that $4\delta \leq \alpha$. $\triangleleft$

Conditioned on the event E occurring, Claim 29 implies that

$$\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \geq (\frac{\gamma}{16} - 5\alpha)n^2 t^\star$$

when n is sufficiently large.

Putting the bounds on $t^\star$ together.

Putting our bounds together, we have that

$$(\frac{\gamma}{16} - 5\alpha)n^2 t^\star \leq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq \frac{\gamma}{4} t^\star n^2$$

when n is sufficiently large and E occurs. Using these inequalities, we can prove the correctness of our algorithm's output. First, we show the upper bound. We have

$$\begin{aligned}
\mathcal{O}(f(x) \wedge g_{i^\star,t^\star}(y)) - t^\star n^2 &\leq (1+\alpha)\mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g_{i^\star,t^\star}(y)) - t^\star n^2 \\
&\leq (1+\alpha)[\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{AND}}_{d-1}(g_{i^\star,t^\star})] - t^\star n^2 \\
&\leq (1+\alpha)[\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + (1+4\delta)t^\star n^2] - t^\star n^2 \\
&\leq (1+\alpha)\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + (1+2\alpha+8\delta)t^\star n^2 - t^\star n^2 \\
&\leq (1+\alpha)\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + (2\alpha+8\delta)t^\star n^2 \\
&\leq (1+\alpha)\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \frac{2\alpha+8\delta}{\frac{\gamma}{16} - 5\alpha}\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \\
&\leq (1+\alpha)\mathsf{L}^{\mathsf{AND}}_{d-1}(f) + O(\alpha)\cdot \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \\
&\leq (1+O(\alpha))\mathsf{L}^{\mathsf{AND}}_{d-1}(f)
\end{aligned}$$

where the third inequality comes from $g_{i^\star,t^\star}$ being good, the sixth inequality comes from the lower bound on $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$, and the seventh inequality comes from setting δ sufficiently small and since $\alpha < \gamma/10^3$.

Next, we argue the lower bound on the output. For this we will again make use of Theorem 5 in order to obtain the lower bound

$$\mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g_{i^\star,t^\star}(y)) \geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g_{i^\star,t^\star}).$$

To do this, we must check that the two hypothesis of Theorem 5 hold. In particular, we know that f is not a constant function (since the brute force stage ensures $\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \geq n^{1024/\delta}$) and $g_{i^\star,t^\star}$ is not constant (because it is good) and we have that

$$\mathsf{L}^{\mathsf{OR}}_{d}(g_{i^\star,t^\star}) + \mathsf{L}^{\mathsf{AND}}_{d-1}(f) \leq (1+4\cdot\delta)t^\star n^2 + \frac{\gamma}{4}t^\star n^2 \leq \min\{\mathsf{L}_{\mathsf{ND}}(g_{i^\star,t^\star}) + \mathsf{L}_{\mathsf{ND},\gamma}(g_{i^\star,t^\star}), 2\cdot\mathsf{L}_{\mathsf{ND},.73}(g_{i^\star,t^\star})\}$$

using that $g_{i^\star,t^\star}$ is good, the inequality on $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$ and setting δ sufficiently small. This means we can indeed apply Theorem 5. We make use of it to derive our lower bound

$$\begin{aligned}
\mathcal{O}(f(x) \wedge g_{i^\star,t^\star}(y)) - t^\star n^2 &\geq \mathsf{L}^{\mathsf{OR}}_{d}(f(x) \wedge g_{i^\star,t^\star}(y)) - t^\star n^2 \\
&\geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + \mathsf{L}^{\mathsf{OR}}_{d}(g_{i^\star,t^\star}) - t^\star n^2 \\
&\geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) + (1-4\delta)t^\star n^2 - t^\star n^2 \\
&\geq \mathsf{L}^{\mathsf{AND}}_{d-1}(f) - 4\delta t^\star n^2 \\
&\geq (1 - \frac{4\delta}{\frac{\gamma}{16} + 5\alpha})\mathsf{L}^{\mathsf{AND}}_{d-1}(f) \\
&\geq (1-2\alpha)\mathsf{L}^{\mathsf{AND}}_{d-1}(f).
\end{aligned}$$

where the second inequality comes from Theorem 5, the third inequality comes from $g_{i^\star,t^\star}$ being good, and the last inequality comes from setting δ sufficiently small.

Hence, we have the algorithm outputs $(1+O(\alpha))$ approximation of $\mathsf{L}^{\mathsf{AND}}_{d-1}(f)$, as desired. ◀

Next, we prove Lemma 26. We note that the functions we use in the proof of this lemma are taken from Lupanov's construction of asymptotically optimal depth-3 formulas [26]. In particular, one can view our functions as the functions computed by the depth-2 subformulas in Lupanov's depth-3 formulas.

▶ **Lemma 26.** *Let $\gamma = 10^{-4}$. Let $0 < \delta < \frac{\gamma}{16}$ be a parameter such that $\frac{1}{\delta} \in \mathbb{N}$. Let n and t be positive integers satisfying $n^{\frac{8}{\delta}} \le t \le 2^n$. Then there exists a distribution $\mathcal{D}_{n,t,\delta}$ of Boolean functions with $(n + n^{2/\delta})$-inputs samplable in time quasipolynomial in 2^n such that if $g \leftarrow \mathcal{D}_{n,t,\delta}$, then with probability $1 - o_\delta(1)$ all of the following hold*

1. $(1-4\delta)tn^2 \le \mathsf{L}_{\mathsf{ND}}(g) \le \mathsf{L}_2^{\mathsf{AND}}(g) \le (1+4\delta)tn^2$,
2. $\min\{\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g), 2\cdot \mathsf{L}_{\mathsf{ND},.73}(g)\} \ge (1+\frac{\gamma}{2})tn^2$.

Proof. Fix some positive integers n and t satisfying $n^{\frac{8}{\delta}} \le t \le 2^n$. Set $m = n^{\frac{2}{\delta}}$. Note that $t \ge m^4$.

Defining the distribution.

Our distribution $\mathcal{D}_{n,t,\delta}$ on Boolean functions will be as follows. For each $y \in [t]$, sample $Z_y \subseteq [m]$ to be a random subset of $[m]$ where each element of $[m]$ is placed in Z_y independently with probability $m^{\delta-1}$. The Boolean function output by the distribution is the function

$$g : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$$

where $g(y,z) = 1$ if and only if all of the following hold:

- $\mathsf{wt}(z) = 1$ (recall, $\mathsf{wt}(z)$ denotes the number of ones in z),
- $y \in [t]$ (We interpret y as an element of $[2^n]$ in the natural way. So, $y \in [t]$ if and only if the binary integer represented by y is at most $t-1$. Note that $t \le 2^n$.), and
- the jth bit of z is one for some $j \in Z_y$.

This completes our description of the distribution $\mathcal{D}_{n,t,\delta}$. It is easy to see that one can sample a function from $\mathcal{D}_{n,t,\delta}$ in time $2^{O(m\cdot n)}$ which is quasipolynomial in 2^n.

Union bounding against a bad event.

We now establish that a function g sampled from $\mathcal{D}_{n,t,\delta}$ has the desired properties with high probability. To begin, we consider a high probability event involving $\sum_{y\in[t]} |Z_y|$. Since $\sum_{y\in[t]} |Z_y|$ is the sum of $m \cdot t$ independent Bernoulli random variables with probabilty $m^{\delta-1}$ of being one and $m \cdot t \cdot m^{\delta-1} = n^2 t$, Chernoff bounds imply that

$$tn^2(1-\delta) \le \sum_{y\in[t]} |Z_y| \le tn^2(1+\delta)$$

with probability at least $1-o(1)$. Thus, we can union bound over this $o(1)$ failure probability and assume for the remainder of this proof that when n is sufficiently large we have that

$$tn^2(1-\delta) \le \sum_{y\in[t]} |Z_y| \le tn^2(1+\delta).$$

Upper bounding the complexity of g.

Next, we establish the upper bound $\mathsf{L}_2^{\mathsf{AND}}(g) \leq (1+4\delta)n^2 t$. Observe that we can compute g as follows:

$$g(y,z) = \mathbb{1}_{\mathsf{wt}(z)=1} \wedge \mathbb{1}_{y\in[t]} \wedge \bigwedge_{\tilde{y}\in[t]} (\mathbb{1}_{y\neq\tilde{y}} \vee (\bigvee_{j\in Z_{\tilde{y}}} z_j))$$

where z_j denotes the jth bit of y.

The next two claims upper bound the complexity of this formula in pieces.

▷ Claim 30. $\mathsf{L}_2^{\mathsf{AND}}(\mathbb{1}_{\mathsf{wt}(y)=1}) \leq 2m^2$.

Proof. We can compute $\mathbb{1}_{\mathsf{wt}(z)=1}$ by checking if at least one bit of z is one and then checking if for each pair of bits that at least one of them is zero. That is,

$$\mathbb{1}_{\mathsf{wt}(z)=1} = (z_1 \vee \cdots \vee z_m) \wedge \bigwedge_{j\neq j'\in[m]} (\neg z_j \vee \neg z_{j'})$$

so $\mathsf{L}_2^{\mathsf{AND}}(\mathbb{1}_{\mathsf{wt}(y)=1}) \leq m + m^2/2 \leq 2m^2$. ◁

▷ Claim 31. $\mathsf{L}_2^{\mathsf{AND}}(\mathbb{1}_{y\in[t]}) \leq (t+1)n$

Proof. Pick the integer k so that $2^{k-1} < t \leq 2^k$. Then

$$\mathbb{1}_{y\in[t]} = \mathbb{1}_{y\in[2^k]} \wedge \bigwedge_{\tilde{y}\in[2^k]\setminus[t]} \mathbb{1}_{\tilde{y}\neq y}.$$

It is easy to see that $\mathsf{L}_2^{\mathsf{AND}}(\mathbb{1}_{y\in[2^k]}) \leq n$ (you just check that the first $n-k$ bits of y are zero), and since $2^k - t \leq 2t - t = t$, we get that

$$\mathsf{L}_2^{\mathsf{AND}}(\bigwedge_{\tilde{y}\in[2^k]\setminus[t]} \mathbb{1}_{\tilde{y}\neq y}) \leq |[2^k]\setminus[t]| \cdot n \leq tn.$$

◁

Putting these bounds together, we get that

$$\begin{aligned}
\mathsf{L}_2^{\mathsf{AND}}(g) &\leq 2m^2 + (t+1)n + t\cdot n + \sum_{\tilde{y}\in[t]} |Z_{\tilde{y}}| \\
&\leq 2m^2 + (t+1)n + t\cdot n + tn^2(1+\delta) \\
&\leq tn^2(1+4\delta)
\end{aligned}$$

when n is sufficiently large (note that n being sufficiently large can be absorbed into the $o_\delta(1)$ failure probability in the lemma statement) and where the second inequality comes from our previous assumption that

$$tn^2(1-\delta) \leq \sum_{y\in[t]} |Z_y| \leq tn^2(1+\delta).$$

Lower bounding the complexity of g.

It remains to prove the lower bounds in the lemma statement. To prove these lower bounds, we use the following claim.

▷ Claim 32. Let $0 < \epsilon \leq 1$. With probability $1-o_{\epsilon,\delta}(1)$, we have that $\mathsf{L}_{\mathsf{ND},\epsilon}(g) \geq \epsilon(1-4\delta)tn^2$.

Before we prove the claim, we show how we can use it to finish the proof of the lemma. In particular, the claim implies that with probability $1 - o(1)$ all of the following hold

- $\mathsf{L}_{\mathsf{ND}}(g) \geq (1-4\delta)tn^2$,
- $\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g) \geq (1+\gamma)(1-4\delta)tn^2$, and
- $2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g) \geq 2 \cdot (.73)(1-4\delta)tn^2$.

Thus, to prove the lemma we require that both of the following hold

- $(1+\gamma)(1-4\delta) \geq 1 + \frac{\gamma}{2}$, and
- $2 \cdot (.73)(1-4\delta) \geq 1 + \frac{\gamma}{2}$.

Hence, the lemma is true since $\delta \leq \gamma/16$.

It remains to prove the claim.

Proof of Claim. We prove this by a union bound argument. Fix any $h : \{0,1\}^{n+m} \to \{0,1\}$. We bound the probability that h is an ϵ-one-sided approximation for g. By construction, we have that $|g^{-1}(1)| = \sum_{y \in [t]} |Z_y|$. Since we have already union bounded against the possibility that $\sum_{y\in[t]} |Z_y| < (1-\delta)tn^2$, we know that h computes an ϵ one-sided approximation of g with probability zero if $|h^{-1}(1)| < \epsilon \cdot (1-\delta)tn^2$.

On the other hand, suppose that $|h^{-1}(1)| \geq \epsilon(1-\delta)tn^2$. Then, since each value of g is an independent Bernoulli random variable, whose probability of equalling one is at most $m^{\delta-1}$, we get that the probability g outputs one whenever h outputs one is at most

$$(m^{\delta-1})^{\epsilon(1-\delta)tn^2} = m^{-(1-\delta)\epsilon(1-\delta)tn^2} = 2^{-(1-\delta)\frac{2}{\delta}\epsilon(1-\delta)tn^2 \log n} = O(2^{-\frac{2}{\delta}(1-3\delta)\epsilon tn^2 \log n}).$$

In contrast, using Proposition 9, the number of functions computed by a non-deterministic formula size s with $m+n$ inputs and $m+n$ non-deterministic inputs is at most

$$2^{s \log(100(m+n))} \leq 2^{s \log(200m)} \leq 2^{\frac{2}{\delta} s \log(200n)}.$$

Thus, setting $s = \epsilon(1-4\delta)tn^2$ we get the number of functions computed by a non-determinstic formula of size s is bounded by

$$2^{\frac{2}{\delta}\epsilon(1-4\delta)tn^2 \log(200n)}.$$

Hence, the probability an ϵ-one-sided approximation of g can be computed by a non-deterministic formula of size at most $\epsilon(1-4\delta)tn^2$ is bounded above by

$$O(2^{-\frac{2}{\delta}(1-3\delta)tn^2 \log n}) \cdot 2^{\frac{2}{\delta}\epsilon(1-4\delta)tn^2 \log(200n)} = o_{\epsilon,\delta}(1).$$

◁

◀

7 NP Hardness of $\mathsf{L}_2^{\mathsf{OR}}$

After a long line of work that began with Masek [27], Khot and Saket [23] proved near optimal hardness of approximation for minimizing the number of terms in a DNF.

▶ **Theorem 33** (Khot and Saket [23]). *Given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, determining the minimum number of terms in a DNF for computing f up to a factor of $n^{1-\epsilon}$ is* NP *hard under quasipolynomial time Turing reductions for all $\epsilon > 0$.*

We will need a version of Khot and Saket's theorem that proves hardness of minimizing the number of leaves in a DNF (which is our size measure). This follows from an easy reduction.

▶ **Corollary 24** (Easy corollary of Khot and Saket [23]). *Given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, determining $\mathsf{L}_2^{\mathsf{OR}}(f)$ up to a factor of $n^{1-\epsilon}$ is* NP*-hard under quasipolynomial time Turing reductions for arbitrarily small $\epsilon > 0$.*

Proof. Let $\epsilon > 0$. We show that, given an oracle $\mathcal{O}$ that computes $\mathsf{L}_2^{\mathsf{OR}}$ up to a factor of $n^{1-\epsilon}$ and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute in polynomial time the minimum number of terms in any DNF for f up to a factor of $O(n^{1-\epsilon})$.

The algorithm is as follows. Given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, define $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ by

$$f'(x,y) = f(x) \wedge \bigwedge_{i\in[n]} y_i$$

where y_i index the bits of y. Output $\frac{\mathcal{O}(f')}{n}$.

It is easy to see that this is a polynomial time reduction, so it remains to argue for correctness. Let $q^\star$ be the minimum number of terms in a DNF required to compute f. It is easy to see that if f can be computed by a DNF $\varphi = \bigvee_{j\in[q^\star]} \varphi_i$ with $q^\star$ terms then f' can be computed by a DNF

$$\varphi' = \bigvee_{j\in[q]} [\varphi_i \wedge y_1 \cdots \wedge y_n]$$

with at most $2nq^\star$ leaves.

On the other hand, suppose that $\mathsf{L}_2^{\mathsf{OR}}(f') = s$ and $\varphi' = \bigvee_{i\in[q']} \varphi'_i$ is a DNF for f' with s leaves. By the optimality of φ', we know that each φ'_i must output one on at least one input. It follows that φ'_i uses at least n literals since it must include $y_1 \wedge \cdots \wedge y_n$ in order to only accept YES instances of f'. Hence, we have that $s \geq q'n$. Therefore, there exists a DNF for f with at most q' terms by setting the values of $y_1 = \cdots = y_n = 1$ in φ', so $q^\star \leq q' \leq s/n$.

Putting these two bounds together, we get that

$$q^\star \leq \frac{\mathsf{L}_2^{\mathsf{OR}}(f')}{n} \leq 2q^\star.$$

Therefore, we have that our output $\frac{\mathcal{O}(f')}{n}$ satisfies the following guarantee

$$q^\star \leq \frac{\mathsf{L}_2^{\mathsf{OR}}(f')}{n} \leq \frac{\mathcal{O}(f')}{n} \leq (2n)^{1-\epsilon}\frac{\mathsf{L}_2^{\mathsf{OR}}(f')}{n} \leq O(n^{1-\epsilon}q^\star),$$

as desired. ◀

8 OR-top to General Reduction

In this section we will prove the following theorem.

▶ **Theorem 23.** *Let $d \geq 2$ be an integer. Let $\alpha \geq 0$. Given access to an oracle $\mathcal{O}$ that computes an $(1+\alpha)$ multiplicative approximation to L_d and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}_d^{\mathsf{OR}}(f)$ and $\mathsf{L}_d^{\mathsf{AND}}(f)$ up to a factor of $(1+\alpha)^2$ in deterministic quasipolynomial time.*

In our proof we will make use of known depth hierarchy theorems for AC^0 formulas. Various versions of these hierarchy theorems suffice for our purposes. We cite the one in [13] since it is clearest from the theorem statement that the depth d upper bound is given by a read once formula.

It will be important to us that these results are "explicit." We say a function family $f_n : \{0,1\}^n \to \{0,1\}$ is *explicit* if there is a deterministic algorithm A_{f_n} that given the input 1^n outputs the truth table of f_n in time $2^{O(n)}$. We say a family of formulas φ_n that take n-inputs is *explicit* if there is a deterministic algorithm A that on input 1^n outputs φ_n in time $2^{O(n)}$.

▶ **Theorem 34** (Håstad, Rossman, Servedio and Tan [13]). *Let $d \geq 2$. There is an explicit function Sipser_d that can be computed by an explicit depth-d read once formula, but requires depth-$(d-1)$ formulas of size $2^{n^{\Omega(1/d)}}$ to compute.*

A consequence of this hierarchy theorem is that there exist explicit functions that are much easier to compute via a depth-d formula with a top OR gate compared to a top AND gate.

▶ **Corollary 35.** *Let $d \geq 2$. There exists an explicit function $g_n : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{L}_d^{\mathsf{OR}}(g_n) \leq n$ and $\mathsf{L}_d^{\mathsf{AND}}(g_n) \geq 2^{n^{\Omega(1/d)}}$.*

Proof. Fix $d \geq 2$. Our function $g_n : \{0,1\}^n \to \{0,1\}$ is defined as follows. By Theorem 34, there is an explicit function Sipser_{d+1} on n-inputs that is computed by an explicit depth-$(d+1)$ read once formula φ_n. Without loss of generality assume that the top gate of φ_n is an AND gate (if this is not the case, then use $\neg\mathsf{Sipser}_{d+1}$ instead of Sipser_{d+1}). Then we can write $\varphi_n = \bigwedge_{i \in [k]} \varphi_n^i$ where each $\varphi_n^1, \ldots, \varphi_n^k$ are $\mathsf{OR} \circ \mathsf{AC}^0_{d-1}$ formulas that are read once on pairwise disjoint inputs. Furthermore, $\sum_{i \in [k]} |\varphi_n^i| = |\varphi_n| = n$.

We then let $g_n : \{0,1\}^n \to \{0,1\}$ be the function computed by

$$g_n(x) = \bigvee_{i \in [k]} \varphi_n^i(x).$$

By construction, we have that $\mathsf{L}_d^{\mathsf{OR}}(g_n) \leq n$.

It remains to lower bound $\mathsf{L}_d^{\mathsf{AND}}(g_n)$. Since $\varphi_n^1, \ldots, \varphi_n^k$ use pairwise disjoint inputs, the direct sum rules in Proposition 6 imply that[5]

$$\mathsf{L}_d^{\mathsf{AND}}(g_n) \geq \sum_{i \in [k]} \mathsf{L}_d^{\mathsf{AND}}(\varphi_n^i).$$

On the other hand, since $\varphi_n = \bigwedge_{i \in [k]} \varphi_n^i$ computes Sipser_{d+1} we have that

$$\sum_{i \in [k]} \mathsf{L}_d^{\mathsf{AND}}(\varphi_n^i) \geq \mathsf{L}_d^{\mathsf{AND}}\Big(\bigwedge_{i \in [k]} \varphi_n^i\Big) \geq \mathsf{L}_d(\mathsf{Sipser}_{d+1}) \geq 2^{n^{\Omega(1/d)}}$$

where the last lower bound comes from Theorem 34. Hence, we can conclude that

$$\mathsf{L}_d^{\mathsf{AND}}(g_n) \geq 2^{n^{\Omega(1/d)}}$$

◀

Now we are ready to prove Theorem 23

▶ **Theorem 23.** *Let $d \geq 2$ be an integer. Let $\alpha \geq 0$. Given access to an oracle $\mathcal{O}$ that computes an $(1+\alpha)$ multiplicative approximation to L_d and given the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}_d^{\mathsf{OR}}(f)$ and $\mathsf{L}_d^{\mathsf{AND}}(f)$ up to a factor of $(1+\alpha)^2$ in deterministic quasipolynomial time.*

---

[5] Here we begin abusing notation by writing $\mathsf{L}_d^{\mathsf{AND}}(\varphi_n^i)$ to mean $\mathsf{L}_d^{\mathsf{AND}}(h_n^i)$ where h_n^i is the function computed by φ_n^i

Proof. By applying DeMorgan's laws as in Proposition 7, we know that $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d^{\mathsf{OR}}(\neg f)$, so it suffices to show how to compute $\mathsf{L}_d^{\mathsf{OR}}(f)$ in polynomial time given oracle access to L_d.

Let m be a parameter we set later. Let $g_m : \{0,1\}^m \to \{0,1\}$ be the explicit function given in Corollary 35 such that $\mathsf{L}_d^{\mathsf{OR}}(g_m) \leq m$ and $\mathsf{L}_d^{\mathsf{AND}}(g_m) \geq 2^{m^{\Omega(1/d)}}$.

Our algorithm for computing $\mathsf{L}_d^{\mathsf{OR}}(f)$ given oracle access to L_d will be as follows. First, using brute force, we iterate through all formulas of size at most $\frac{m}{\alpha}$ on n-inputs and output $\mathsf{L}_d^{\mathsf{OR}}(f)$ exactly if we find a formula computing f. Otherwise, we output $\mathcal{O}(f(x) \vee g_m(y))$. This completes our description of the algorithm.

Next we argue that this gives the desired output. Clearly, if $\mathsf{L}_d^{\mathsf{OR}}(f) \leq \frac{m}{\alpha}$, the output is correct. Thus we assume that $\mathsf{L}_d^{\mathsf{OR}}(f) > \frac{m}{\alpha}$. The idea is that the cost of using an top AND gate to compute g_m is so high that the any optimal circuit for $f(x) \vee g_m(y)$ must use a top OR gate regardless of what f is doing. Indeed, computing $f(x) \vee g_m(y)$ using a top OR gate, we get that

$$\mathsf{L}_d^{\mathsf{OR}}(f(x) \vee g_m(y)) = m + \mathsf{L}_d^{\mathsf{OR}}(f) \leq m + n2^n$$

where the equality comes from the direct sum rules in Proposition 6 and the inequality comes from the trivial DNF upper bound. On the other hand

$$\mathsf{L}_d^{\mathsf{AND}}(f(x) \vee g_m(y)) \geq \mathsf{L}_d^{\mathsf{AND}}(g_m) \geq 2^{m^{\Omega(1/d)}}$$

where the first inequality comes from the direct sum rules in Proposition 6 and the last inequality comes from our the properties of g_m.

We now set $m = n^{O_d(1)}$ such that

$$\mathsf{L}_d^{\mathsf{AND}}(f(x) \vee g_m(y)) \geq 2^{m^{\Omega(1/d)}} \geq 2^{n^2}.$$

We can then conclude that $\mathsf{L}_d^{\mathsf{OR}}(f(x) \vee g_m(y)) \leq m + n2^n$ and $\mathsf{L}_d^{\mathsf{AND}}(f(x) \vee g_m(y)) \geq 2^{n^2}$. Hence we have that

$$\mathsf{L}_d(f(x) \vee g_m(y)) = \mathsf{L}_d^{\mathsf{OR}}(f(x) \vee g_m(y)) = \mathsf{L}_d^{\mathsf{OR}}(f) + m$$

when n is sufficiently large. Since $\mathsf{L}_d^{\mathsf{OR}}(f) \geq \frac{m}{\alpha}$, we get that

$$\mathsf{L}_d^{\mathsf{OR}}(f) \leq \mathsf{L}_d(f(x) \vee g_m(y)) \leq (1+\alpha)\mathsf{L}_d^{\mathsf{OR}}(f).$$

Thus, we can conclude that $\mathcal{O}(f(x) \vee g_m(y))$ gives a $(1+\alpha)^2$ approximation of $\mathsf{L}_d^{\mathsf{OR}}(f)$, as desired.

Finally, we analyze the running time of this algorithm. The brute force stage of the algorithm takes time roughly

$$2^{O(\frac{m}{\alpha}\log n)} = 2^{n^{O(1)}}$$

and constructing the truth table for the oracle query can also be done in $2^{n^{O(1)}}$ time. Thus, the algorithm runs in time quasipolynomial in N, as desired. ◀

8.1 An alternate version avoiding the switching lemma.

Note to the reader: the remainder of this section is not strictly necessary to read and can safely be skipped.

One may ask how necessary "switching lemma" types of lower bounds (such as the one used to prove the depth hierarchy theorem we make use of in Theorem 34) to our reduction.

Indeed, Theorem 23 is the only place where we use such lower bounds. However, we can actually get by without using switching lemma style techniques, albeit with a loss in hardness of approximation. We show how to do this in the next proof, which only really makes use of direct sum rules and DeMorgan's laws.

▶ **Theorem 36.** *Let $d \geq 2$. Given access to an oracle computing L_d and the truth table of a function $f : \{0,1\}^n \to \{0,1\}$, one can compute $\mathsf{L}_d^{\mathsf{OR}}(f)$ and $\mathsf{L}_d^{\mathsf{AND}}(f)$ in polynomial time.*

Proof. By applying DeMorgan's laws as in Proposition 7, we know that $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d^{\mathsf{OR}}(\neg f)$, so it suffices just to show how to compute $\mathsf{L}_d^{\mathsf{OR}}(f)$ in polynomial time given oracle access to L_d.

Fix $d \geq 2$. We split into two cases. First, we consider the case that for all functions h that

$$\mathsf{L}_d^{\mathsf{OR}}(h) = \mathsf{L}_d(h).$$

(We actually know this case is false by Corollary 35, but we want to avoid using any switching lemma style results in this proof.) In this case, we can clearly get the desired algorithm for computing $\mathsf{L}_d^{\mathsf{OR}}(f)$ by just outputting $\mathsf{L}_d(f)$.

For the second case, we know that there exists a function $h : \{0,1\}^m \to \{0,1\}$ such that $\mathsf{L}_d^{\mathsf{OR}}(h) \neq \mathsf{L}_d(h)$. Then we must have that $\mathsf{L}_d^{\mathsf{OR}}(h) > \mathsf{L}_d^{\mathsf{AND}}(h)$.

Given a function $f : \{0,1\}^n \to \{0,1\}$, our algorithm for computing $\mathsf{L}_d^{\mathsf{OR}}(f)$ is simply to output

$$\begin{cases} \mathsf{L}_d(f) & \text{, if } \mathsf{L}_d(f(x) \wedge h(y)) \neq \mathsf{L}_d(f) + \mathsf{L}_d(h) \\ \mathsf{L}_d(f(x) \wedge \neg f(y)) - \mathsf{L}_d(f) & \text{, otherwise.} \end{cases}$$

It is easy to see that this algorithm runs in polynomial-time, so we just need to show that the algorithm produces the correct output. We will do this by proving two claims:

1. $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f)$ if and only if $\mathsf{L}_d(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h)$.
2. $\mathsf{L}_d^{\mathsf{max}}(f) = \mathsf{L}_d(f(x) \wedge \neg f(y)) - \mathsf{L}_d(f)$

where we define $\mathsf{L}_d^{\mathsf{max}}(f) = \max\{\mathsf{L}_d^{\mathsf{OR}}(f), \mathsf{L}_d^{\mathsf{AND}}(f)\}$

Assuming that (1) and (2) are true, we can prove the correctness of the algorithm as follows.

If $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f)$, then by (1) we have that $\mathsf{L}_d(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h)$, so the algorithm will output

$$\mathsf{L}_d(f(x) \wedge \neg f(y)) - \mathsf{L}_d(f) = \mathsf{L}_d^{\mathsf{max}}(f) = \mathsf{L}_d^{\mathsf{OR}}(f)$$

where the first equality comes from (2) and the last equality is because $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f)$.

On the other hand, if $\mathsf{L}_d^{\mathsf{AND}}(f) \neq \mathsf{L}_d(f)$, then by (1) we have that $\mathsf{L}_d(f(x) \wedge h(y)) \neq \mathsf{L}_d(f) + \mathsf{L}_d(h)$, so the algorithm outputs

$$\mathsf{L}_d(f) = \mathsf{L}_d^{\mathsf{OR}}(f)$$

where the equality comes from $\mathsf{L}_d^{\mathsf{AND}}(f) \neq \mathsf{L}_d(f)$.

Hence, to prove the correctness of the algorithm, it suffices to prove (1) and (2), which we show in the following claims.

▷ Claim 37. (1) is true. That is, $\mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f)$ if and only if $\mathsf{L}_d(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h)$.

Proof. We begin by establishing that $\mathsf{L}_d^{\mathsf{OR}}(f(x) \wedge h(y)) > \mathsf{L}_d(f) + \mathsf{L}_d(h)$. Indeed, we have that

$$\mathsf{L}_d^{\mathsf{OR}}(f(x) \wedge h(y)) \geq \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d^{\mathsf{OR}}(h) > \mathsf{L}_d^{\mathsf{OR}}(f) + \mathsf{L}_d(h) \geq \mathsf{L}_d(f) + \mathsf{L}_d(h)$$

where the first inequality comes from the direct sum rules in Proposition 6 and the second inequality comes from the assumption that $\mathsf{L}_d(h) \neq \mathsf{L}_d^{\mathsf{OR}}(h)$.

As a consequence, we have that

$$\mathsf{L}_d(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h) \iff \mathsf{L}_d^{\mathsf{AND}}(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h).$$

However, we know that

$$\begin{aligned} \mathsf{L}_d^{\mathsf{AND}}(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h) &\iff \mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f) \text{ and } \mathsf{L}_d^{\mathsf{AND}}(h) = \mathsf{L}_d(h) \\ &\iff \mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f) \end{aligned}$$

where the first equivalence comes from the direct sum rules in Proposition 6 and the second equivalence comes from the assumption that $\mathsf{L}_d(h) \neq \mathsf{L}_d^{\mathsf{OR}}(h)$.

Thus we have established

$$\mathsf{L}_d(f(x) \wedge h(y)) = \mathsf{L}_d(f) + \mathsf{L}_d(h) \iff \mathsf{L}_d^{\mathsf{AND}}(f) = \mathsf{L}_d(f)$$

as desired. ◁

▷ Claim 38. (2) is true. That is, $\mathsf{L}_d^{\mathsf{max}}(f) = \mathsf{L}_d(f(x) \wedge \neg f(y)) - \mathsf{L}_d(f)$.

Proof. From Proposition 8 we know that

$$\mathsf{L}_d(f(x) \wedge \neg f(y)) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f).$$

Hence, we get that

$$\mathsf{L}_d(f(x) \wedge \neg f(y)) - \mathsf{L}_d(f) = \mathsf{L}_d^{\mathsf{AND}}(f) + \mathsf{L}_d^{\mathsf{OR}}(f) - \mathsf{L}_d(f) = \mathsf{L}_d^{\mathsf{max}}(f)$$

as desired. ◁

◀

9 Gaps in Complexity Between Depths

In this section we prove Theorem 2.

▶ **Theorem 2** (Proved in Section 9). *For all $d \geq 2$ there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f) \geq 2^{\Omega_d(n)}$.*

The main idea here is to "lift" the $2^{\Omega(n)}$ additive gap known for the case of $d = 2$ to higher depths, using the lower bound method in Theorem 5. To do this, we will need a stronger version of Lemma 26 that shows the existence of "non-deterministically hard" truth tables of length polynomial in 2^n rather than quasipolynomial. This comes at the cost of having depth-3 near optimal formulas rather than depth-2, which is why we did not use them in our (AC_d^0)-MCSP hardness result.

Again the inspiration for our proof comes from Lupanov's nearly optimal depth-3 construction [26].

▶ **Lemma 39.** *Let n and t be integers where n is a power of two and $1 \le t \le 2^n/n$. Then there exists a distribution of functions that takes q-inputs where $n \le q \le O(n)$ such that if f is sampled from this distribution then with probability $1 - o(1)$ both of the following hold*

- $(1 - o(1))tn^{11} \le \mathsf{L}_{\mathsf{ND}}(f) \le \mathsf{L}_3^{\mathsf{AND}}(f) \le (1 + o(1))tn^{11}$*, and*
- $\min\{\mathsf{L}_{\mathsf{ND}}(f) + \mathsf{L}_{\mathsf{ND},\gamma}(f), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(f)\} \ge (1 + \gamma/4)tn^{11}$ *where* $\gamma = 10^{-4}$.

We defer the proof of Lemma 39 (which is essentially a counting argument) to the end of the section. We use this lemma to prove the desired gap result.

To start, we prove a weaker version of Theorem 2.

▶ **Theorem 40.** *Let $d \ge 2$. There exists a family of functions $f_n : \{0,1\}^{\Theta_d(n)} \to \{0,1\}$ such that $\mathsf{L}_d^{\mathsf{OR}}(f_n) - \mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) \ge 2^{\Omega_d(n)}$.*

Proof. We work by induction on d. Our inductive hypothesis is that there exists a family of functions $f_n : \{0,1\}^{\Theta_d(n)} \to \{0,1\}$ such that both of the following hold:

1. $\mathsf{L}_d^{\mathsf{OR}}(f_n) = 2^{\Omega_d(n)}$, and
2. $\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) = (1 - \Omega_d(1))\mathsf{L}_d^{\mathsf{OR}}(f_n)$.

Base Case.

For the base case of $d = 2$, we can let $f_n : \{0,1\}^n \to \{0,1\}$ be given by the parity function PARITY_n. It is a folklore result that

- $\mathsf{L}_2^{\mathsf{OR}}(\mathsf{PARITY}_n) = n2^n$ (using the fact that any subcube with more than one element must contain both YES and NO instances of PARITY_n), and
- $\mathsf{L}_3^{\mathsf{OR}}(\mathsf{PARITY}_n) \le 2^{O(\sqrt{n})}$ (by computing PARITY_n via a divide and conquer approach)

Thus, it is easy to see that PARITY_n satisfies the inductive hypothesis.

Inductive Step.

Now suppose that we have proved the theorem for some $d \ge 2$, and we want to prove the $d + 1$ case. We will construct a family of functions f_n satisfying the inductive hypothesis for depth $d + 1$.

Let $\neg h_n : \{0,1\}^{\Theta_d(n)} \to \{0,1\}$ denote the family of functions satisfying the inductive hypothesis for depth d. Combining the inductive hypothesis with DeMorgan's laws, we have that

1. $\mathsf{L}_d^{\mathsf{AND}}(h_n) = 2^{\Omega_d(n)}$, and
2. $\mathsf{L}_{d+1}^{\mathsf{AND}}(h_n) = (1 - \Omega_d(1))\mathsf{L}_d^{\mathsf{AND}}(h_n)$.

We now construct f_n (note it suffices to do this when n is sufficiently large). Fix some positive integer n. Let $m = \Theta_d(n)$ be the least power of two greater than the number of inputs h_n takes. Using condition (1) on h_n and the trivial CNF upper bound, we know that

$$2^{\Omega_d(m)} \le \mathsf{L}_d^{\mathsf{AND}}(h_n) \le m2^m.$$

Thus, when n is sufficiently large there must exist an integer t such that $1 \le t \le 2^m/m$ and such that

$$\frac{8}{\gamma}\mathsf{L}_d^{\mathsf{AND}}(h_n) \le tm^{11} \le \frac{16}{\gamma}\mathsf{L}_d^{\mathsf{AND}}(h_n)$$

where $\gamma = 10^{-4}$.

Then by Lemma 39, there exists a function $g : \{0,1\}^r \to \{0,1\}$ where $m \le r \le O_d(n)$ such that both of the following hold

- $(1 - o(1))tm^{11} \leq \mathsf{L}_{\mathsf{ND}}(g) \leq \mathsf{L}_3^{\mathsf{AND}}(g) \leq (1 + o(1))tm^{11}$, and
- $\min\{\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g)\} \geq (1 + \gamma/4)tm^{11}$.

Let $f_n : \{0,1\}^{\Theta_d(n)} \times \{0,1\}^r \to \{0,1\}$ be given by $f_n(x, y) = h_n(x) \wedge g(y)$. Note that f_n takes $\Theta_d(n) + r = \Theta_d(n)$ inputs, as desired.

One can check that f_n satisfies all of the hypotheses of Theorem 5 when n is sufficiently large. (The trickiest condition to verify is:

$$\min\{\mathsf{L}_{\mathsf{ND}}(g) + \mathsf{L}_{\mathsf{ND},\gamma}(g), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(g)\} \geq (1 + \gamma/4)tm^{11} \geq \mathsf{L}_{d+1}^{\mathsf{OR}}(g) + \mathsf{L}_d^{\mathsf{AND}}(h_n)$$

which follows from the hypotheses on g and the choice of t.) Using Theorem 5, we get the following lower bound on f_n

$$\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) \geq \mathsf{L}_d^{\mathsf{AND}}(h_n) + \mathsf{L}_{d+1}^{\mathsf{OR}}(g) \geq \mathsf{L}_d^{\mathsf{AND}}(h_n) + (1 - o(1))tm^{11}.$$

Since $\mathsf{L}_d^{\mathsf{AND}}(h_n) = 2^{\Omega_d(n)}$, this confirms condition (1) of the inductive hypothesis.

On the other hand, we can upper bound the complexity of f_n by

$$\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) \leq \mathsf{L}_d^{\mathsf{AND}}(f_n) \leq \mathsf{L}_d^{\mathsf{AND}}(h_n) + \mathsf{L}_d^{\mathsf{AND}}(g) \leq \mathsf{L}_d^{\mathsf{AND}}(h_n) + (1 + o(1))tm^{11} \leq O(\mathsf{L}_d^{\mathsf{AND}}(h_n))$$

where the last inequality comes from our choice of t.

This allows us to confirm condition (2):

$$\begin{aligned}
\mathsf{L}_{d+2}^{\mathsf{OR}}(f_n) &\leq \mathsf{L}_{d+1}^{\mathsf{AND}}(f_n) \\
&\leq \mathsf{L}_{d+1}^{\mathsf{AND}}(h_n) + \mathsf{L}_3^{\mathsf{AND}}(g) \\
&\leq \mathsf{L}_{d+1}^{\mathsf{AND}}(h_n) + (1 + o(1))tm^{11} \\
&\leq (1 - \Omega_d(1))\mathsf{L}_d^{\mathsf{AND}}(h_n) + (1 + o(1))tm^{11} \\
&\leq \mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) + o(tm^{11}) - \Omega_d(\mathsf{L}_d^{\mathsf{AND}}(h_n)) \\
&\leq \mathsf{L}_{d+1}^{\mathsf{OR}}(f_n) - \Omega_d(\mathsf{L}_d^{\mathsf{AND}}(h_n)) \\
&\leq (1 - \Omega_d(1))\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n).
\end{aligned}$$

where the last four equalities are justified (in order) by:

- condition (2) on h_n,
- the our lower bound on $\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n)$,
- our choice of t, and
- our upper bound on $\mathsf{L}_{d+1}^{\mathsf{OR}}(f_n)$.

◀

We can now prove the full theorem.

▶ **Theorem 2** (Proved in Section 9)**.** *For all $d \geq 2$ there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that $\mathsf{L}_d(f) - \mathsf{L}_{d+1}(f) \geq 2^{\Omega_d(n)}$.*

Proof of Theorem 2. Fix some d. Let $h_n : \{0,1\}^{\Theta_d(n)} \to \{0,1\}$ be the function guaranteed by Theorem 40 satisfying $\mathsf{L}_d^{\mathsf{OR}}(h_n) - \mathsf{L}_{d+1}^{\mathsf{OR}}(h_n) \geq 2^{\Omega_d(n)}$.

Let $M \subseteq \mathbb{N}$ be the set containing all the input lengths of the functions in the family h_n, that is,

$$M = \{m : \text{there is an } n \text{ such that } h_n \text{ takes } m \text{ inputs}\}.$$

Next, define the function $m^\star : \mathbb{N} \to \mathbb{N}$ by

$$m^\star(n) = \begin{cases} 0 & \text{, if } \{1, \ldots, \lfloor n/2 \rfloor\} \cap M = \emptyset \\ \max(\{1, \ldots, \lfloor n/2 \rfloor\} \cap M) & \text{, otherwise} \end{cases}$$

Since h_n takes $\Theta_d(n)$ inputs, we have that $m^\star(n) = \Omega(n)$.

We now define $f_n : \{0,1\}^n \to \{0,1\}$ by

$$f_n(x) = \begin{cases} 0 & \text{, if } m_n = 0 \\ h_{m^\star(n)}(x_1, \dots, x_{m^\star(n)}) \wedge \neg h_{m^\star(n)}(x_{m^\star(n)+1}, \dots, x_{2m^\star(n)}) & \text{, otherwise} \end{cases}$$

Therefore, when n is sufficiently large, we have that

$$\begin{aligned} & \mathsf{L}_{d+1}(f_n) - \mathsf{L}_{d+2}(f_n) \\ \geq & \mathsf{L}_{d+1}(h_{m^\star(n)}(x_1, \dots, x_{m^\star(n)}) \wedge \neg h_{m^\star(n)}(x_{m^\star(n)+1}, \dots, x_{2m^\star(n)})) - \\ & - \mathsf{L}_{d+2}(h_{m^\star(n)}(x_1, \dots, x_{m^\star(n)}) \wedge \neg h_{m^\star(n)}(x_{m^\star(n)+1}, \dots, x_{2m^\star(n)})) \\ = & \mathsf{L}^{\mathsf{OR}}_{d+1}(h_{m^\star(n)}) + \mathsf{L}^{\mathsf{AND}}_{d+1}(h_{m^\star(n)}) - \mathsf{L}^{\mathsf{OR}}_{d+2}(h_{m^\star(n)}) - \mathsf{L}^{\mathsf{AND}}_{d+2}(h_{m^\star(n)}) \\ \geq & \mathsf{L}^{\mathsf{OR}}_{d+1}(h_{m^\star(n)}) - \mathsf{L}^{\mathsf{OR}}_{d+2}(h_{m^\star(n)}) \\ \geq & 2^{\Omega_d(m^\star(n))} \\ \geq & 2^{\Omega_d(n)} \end{aligned}$$

where justifications for these equalities/inequalities are (in order):

1. follows from the definition of f_n, n being sufficiently large, and M being non-empty
2. follows from the properties of direct sums of functions with their negations proved in Proposition 8
3. follows from the quantity $\mathsf{L}^{\mathsf{AND}}_{d+1}(H_{m^\star(n)}) - \mathsf{L}^{\mathsf{AND}}_{d+2}(h_{m^\star(n)})$ being non-negative
4. follows the work above on h_m
5. follows from $m^\star(n) = \Omega(n)$

◀

We end the section by proving Lemma 39.

▶ **Lemma 39.** *Let n and t be integers where n is a power of two and $1 \leq t \leq 2^n/n$. Then there exists a distribution of functions that takes q-inputs where $n \leq q \leq O(n)$ such that if f is sampled from this distribution then with probability $1 - o(1)$ both of the following hold*

- $(1 - o(1))tn^{11} \leq \mathsf{L}_{\mathsf{ND}}(f) \leq \mathsf{L}^{\mathsf{AND}}_3(f) \leq (1 + o(1))tn^{11}$, *and*
- $\min\{\mathsf{L}_{\mathsf{ND}}(f) + \mathsf{L}_{\mathsf{ND},\gamma}(f), 2 \cdot \mathsf{L}_{\mathsf{ND},.73}(f)\} \geq (1 + \gamma/4)tn^{11}$ *where* $\gamma = 10^{-4}$.

Proof. Set $m = 10 \log n$ and set ℓ to be an integer satisfying[6] $n^{1-1/\log(\log(n))} \leq 2^{2^\ell} \leq 4n^{1-1/\log(\log(n))}$. Since n is a power of two, we can partition $\{0,1\}^n$ into Hamming balls of radius one $B_1, \dots, B_{\frac{2^n}{n}}$ by the Hamming code. Let $c^1, \dots, c^{\frac{2^n}{n}} \in \{0,1\}^n$ be the centers of these balls.

We also define an encoding σ of the elements in the set $X = \bigcup_{i \in [t]} B_i$. In particular, let $\sigma : X \to [t] \times [n]$ be the bijection given by

$$\sigma(x) = (i, j) \text{ where } x = c^i \oplus e_j$$

where $e_j = 0^{j-1}10^{n-j-1}$.

[6] If n is small, it may not be possible to set ℓ in this way, but this possibility can just be absorbed into the $o(1)$ failure probability in the lemma statement.

Definition of f

We define the function $f : \{0,1\}^n \times \{0,1\}^m \times \{0,1\}^\ell$ as follows. For each $i \in [t]$, $j \in [n]$ and $y \in \{0,1\}^m$, let $g_{i,j,y} : \{0,1\}^\ell \to \{0,1\}$ be uniformly random function. Then we define f by

$$f(x,y,z) = \begin{cases} 0 & \text{, if } x \notin X \\ g_{i,j,y}(z) & \text{, if } x \in X \text{ and } \sigma(x) = (i,j) \end{cases}.$$

We make a few notes about f before we proceed. First, f takes $n + m + \ell = O(n)$ inputs. Next, let $I = X \times \{0,1\}^m \times \{0,1\}^\ell$. Note that f restricted to I is a uniformly random function, and that f is always zero outside of I. It will also be useful to know that

$$|I| = t \cdot n \cdot 2^m \cdot 2^\ell \geq tn^{11} \cdot (1 - 1/\log(\log(n)))\log(n).$$

Upper bounding the complexity of f

To begin, we prove an upper bound on the complexity of f.

▷ Claim 41.

$$\mathsf{L}_3^{\mathsf{AND}}(f) \leq (1 + o(1))tn^{11}$$

Proof. Lupanov observed that one can compute f via the following $\mathsf{AND} \circ \mathsf{OR} \circ \mathsf{AND}$ formula

$$(\bigvee_{i \in [t]} \mathbb{1}_{x \in B_i}) \wedge \bigwedge_{\substack{\tilde{g}:\{0,1\}^\ell \to \{0,1\}, \\ i \in [t]}} [\mathbb{1}_{x \notin B_i} \vee \tilde{g}(z) \vee \bigvee_{\tilde{y} \in \{0,1\}^m} [\mathbb{1}_{\tilde{y} = y} \wedge \bigwedge_{j \in [n] : g_{i,j,\tilde{y}} = \tilde{g}} (x_j = (c^i)_j)]]$$

where $(c^i)_j$ denotes the jth bit in c^i.

We upper bound the number of leaves in this formula. One can compute $\mathbb{1}_{x \in B_i}$ by checking if at least one bit of x differs from c^i and that for every pair of bits from y at least one agrees with the corresponding bit in c^i. Using this strategy, we get that

$$\mathsf{L}_2(\mathbb{1}_{x \in B_i}) = \mathsf{L}_2(\mathbb{1}_{x \notin B_i}) \leq 2n^2.$$

By the trivial DNF upper bound, we get that $\mathsf{L}_2^{\mathsf{OR}}(\tilde{g}) \leq \ell 2^\ell$. Finally,

$$\mathsf{L}_1^{\mathsf{AND}}(\mathbb{1}_{\tilde{y} = y} \wedge \bigwedge_{j \in [n] : g_{i,j,\tilde{y}}) = \tilde{g}} (x_j = (c^i)_j) \leq m + \sum_{j \in [n] : g_{i,j,\tilde{y}} = \tilde{g}} 1$$

Putting these all together, we get the upper bound

$$\begin{aligned} \mathsf{L}_3^{\mathsf{AND}}(f) &\leq 2tn^2 + t2^{2^\ell}(2n^2 + \ell 2^\ell + m2^m) + \sum_{\tilde{g}, i, \tilde{y}} \sum_{j \in [n] : g_{i,j,\tilde{y}} = \tilde{g}} 1 \\ &\leq 2tn^2 + t2^{2^\ell}(2n^2 + \ell 2^\ell + m2^m) + tn2^m \\ &\leq 2tn^2 + 4tn^{1 - 1/\log(\log(n))}(2n^2 + n + 10n^{10}\log n) + tn^{11} \\ &\leq (1 + o(1))tn^{11} \end{aligned}$$

◁

Lower bounding the complexity of f

We now argue the lower bounds on f. All of these lower bounds are proved via a counting argument. In particular, we will use that the number of nondeterministic formulas of size s with $(n+m+\ell)$-inputs and $(n+m+\ell)$ nondeterministic inputs is bounded by

$$2^{s\log(100(n+m+\ell))} \leq 2^{s\log(200n)}.$$

for sufficiently large n by Proposition 9.

▷ Claim 42. With probability $1-o(1)$,

$$\mathsf{L}_{\mathsf{ND}}(f) \geq (1-o(1))tn^{11}$$

Proof. We use a union bound argument. Since f is a uniformly random function on I, the probability any fixed function h equals f is at most

$$2^{-|I|} \leq 2^{-tn^{11}\cdot(1-1/\log(\log(n)))\log(n)}.$$

The claim follows by combining this probability bound with the $2^{s\log(200n)}$ bound on the number of non-deterministic formulas of size s. ◁

▷ Claim 43. With probability $1-o(1)$,

$$\mathsf{L}_{\mathsf{ND}}(f) + \mathsf{L}_{\mathsf{ND},\gamma}(f) \geq (1+\gamma/4)tn^{11}$$

Proof. In the previous claim, we proved that $\mathsf{L}_{\mathsf{ND}}(f) \geq (1-o(1))tn^{11}$. Thus, we now just need to lower bound $\mathsf{L}_{\mathsf{ND},\gamma}(f)$. We again work via a union bound argument.

The probability there exists any function h with $|h^{-1}(1)| < \gamma\frac{(1-1/\log(\log(n)))|I|}{2}$ that computes a γ one-sided approximation of f is $o(1)$. This is because f is a uniformly random function on I and is zero outside of I, so by a Chernoff bound, we have that f has at least $\frac{(1-1/\log(\log(n)))|I|}{2}$ YES inputs with probability $1-o(1)$.

On the other hand, if $|h^{-1}(1)| \geq \gamma\frac{(1-1/\log(\log(n)))|I|}{2}$, then the probability some fixed function h computes a γ one-sided approximation to f is at most

$$2^{-\gamma\frac{(1-1/\log(\log(n)))|I|}{2}} \leq 2^{-\gamma(1-1/\log(\log(n)))^2 tn^{11}\log(n)/2}$$

since h needs to have at least $\frac{\gamma(1-1/\log(\log(n)))|I|}{2}$ YES instances to have any hope of computing a γ one-sided approximation of f and all these YES instances of h must be YES instances of f.

By combining this probability bound with the $2^{s\log(200n)}$ bound on the number of non-deterministic formulas of size s and $(n+m+\ell)$-inputs, we get that $\mathsf{L}_{\mathsf{ND},\gamma}(f) \geq (\frac{\gamma}{2}-o(1))tn^{11}$ with probability $1-o(1)$. ◁

▷ Claim 44. With probability $1-o(1)$,

$$2\cdot\mathsf{L}_{\mathsf{ND},.73}(f) \geq (1+\gamma/4)tn^{11}$$

Proof. We again use a union bound. Fix some function $h:\{0,1\}^n\times\{0,1\}^m\times\{0,1\}^\ell$. We bound the probability that h computes a .73 one-sided approximation of f.

Set $k = |h^{-1}(1)|$. For h to be a .73 one-sided approximation of f, two events must occur:

1. $h^{-1}(1) \subseteq f^{-1}(1)$
2. $|f^{-1}(1)| \leq k/.73$

We bound the probability that events (1) and (2) both occur. Since f is a uniformly random function on I and zero elsewhere, the probability that event (1) occurs is at most 2^{-k}.

Next, we work to bound the probability that event (2) occurs given that event (1) occurs. Event (2) is equivalent to saying that $\sum_{(x,y,z)\in I}[\mathbb{1}_{f(x,y,z)=1}] \leq k/.73$. If event (1) occurs, then

$$\sum_{(x,y,z)\in I}[\mathbb{1}_{f(x,y,z)=1}] = k + \sum_{(x,y,z)\in I\setminus Y_h}[\mathbb{1}_{f(x,y,z)=1}].$$

Since $\sum_{(x,y,z)\in I\setminus Y_h}[\mathbb{1}_{f(x,y,z)=1}]$ is the sum of $|I|-k$ independent binomial random variables with expectation .5, it follows from a Chernoff bound that the probability that event (2) occurs given event (1) occurs is

$$\Pr[k + \sum_{x\in X\setminus Y_h}\mathbb{1}_{f(x)=1} \leq k/.73] \leq e^{-D(q||.5)\cdot(|I|-k)}$$

where D is the KL divergence function and

$$q = \frac{k(1/.73-1)}{|I|-k} = \frac{\alpha\cdot(1/.73-1)}{1-\alpha}$$

where $\alpha = k/|I|$. Note that when $q \geq 1$, this bound does not make sense, in which case we adopt the convention that $e^{-D(q||.5)} = 1$.

Hence, we have that the probability that h computes a .73 one-sided approximation of f is at most

$$2^{-\alpha\cdot|I|}\cdot e^{-D(\frac{\alpha\cdot(1/.73-1)}{1-\alpha}||.5)\cdot(1-\alpha)|I|}.$$

Using some calculus, we get that this quantity is at most $2^{-.501|I|}$, which is upper bounded by

$$2^{-.501t\cdot n^{11}\cdot(1-1/\log(\log(n)))\log(n)}.$$

Combining this upper bound on the probability that h computes a .73 one-sided approximation of f with the $2^{s\log(200n)}$ bound on the number of non-deterministic formulas of size s and $(n+m+\ell)$-inputs, we get that

$$\mathsf{L}_{\mathsf{ND},.73}(f) \geq (.501 - o(1))tn^{11}$$

with probability $1-o(1)$.

Therefore,

$$2\mathsf{L}_{\mathsf{ND},.73}(f) \geq (1.02 - o(1))tn^{11} \geq (1+\gamma/4)tn^{11}$$

with probability $1-o(1)$. ◁

Combining the last three claims with a union bound completes our proof of this lemma. ◀

References

1 Misha Alekhnovich, Mark Braverman, Vitaly Feldman, Adam R. Klivans, and Toniann Pitassi. The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34, February 2008.

2 Eric Allender. The new complexity landscape around circuit minimization. In *Language and Automata Theory and Applications - 14th International Conference (LATA)*, volume 12038, pages 3–16, 2020.

3 Eric Allender, Lisa Hellerstein, Paul McCabe, Toniann Pitassi, and Michael E. Saks. Minimizing DNF formulas and AC0d circuits given a truth table. In *21st Annual IEEE Conference on Computational Complexity (CCC)*, pages 237–251, 2006.

4 Eric Allender, Michal Koucký, Detlef Ronneburger, and Sambuddha Roy. The pervasive reach of resource-bounded kolmogorov complexity in computational complexity theory. *Journal of Computer and System Sciences*, 77(1):14 – 40, 2011.

5 Dana Angluin, Lisa Hellerstein, and Marek Karpinski. Learning read-once formulas with queries. *J. ACM*, 40(1):185–210, January 1993.

6 Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach.* 2009.

7 David Buchfuhrer and Christopher Umans. The complexity of boolean formula minimization. *J. Comput. Syst. Sci.*, 77(1):142–153, 2011.

8 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *31st Conference on Computational Complexity (CCC)*, volume 50, pages 10:1–10:24, 2016.

9 Sebastian Lukas Arne Czort. The complexity of minimizing disjunctive normal form formulas. Master's thesis, University of Aarhus, 1999.

10 Vitaly Feldman. Hardness of approximate two-level logic minimization and PAC learning with membership queries. In *38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 363–372, 2006.

11 Thomas Hancock, Tao Jiang, Ming Li, and John Tromp. Lower bounds on learning decision lists and trees. *Inf. Comput.*, 126(2):114–122, May 1996.

12 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

13 Johan Håstad, Benjamin Rossman, Rocco A. Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for boolean circuits. *J. ACM*, 64(5):35:1–35:27, 2017.

14 John Håstad. Almost optimal lower bounds for small depth circuits. *Advances in Computing Research*, 5:143–170, 1989.

15 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

16 Shuichi Hirahara, Igor Carboni Oliveira, and Rahul Santhanam. NP-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference (CCC)*, volume 102, pages 5:1–5:31, 2018.

17 Rahul Ilango. Approaching MCSP from above and below: Hardness for a conditional variant and AC0[p]. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 34:1–34:26, 2020.

18 Rahul Ilango. Connecting perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *35th Computational Complexity Conference (CCC)*, volume 169, pages 31:1–31:35, 2020.

19 Rahul Ilango, Bruno Loff, and Igor Carboni Oliveira. NP-hardness of circuit minimization for multi-output functions. In *35th Computational Complexity Conference (CCC)*, volume 169, pages 22:1–22:36, 2020.

20 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

21 Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

22 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *32nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 73–79, 2000.

23 Subhash Khot and Rishi Saket. Hardness of minimizing and learning DNF expressions. In *49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 231–240, 2008.

24 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Lower bounds based on the exponential time hypothesis. *Bull. EATCS*, 105:41–72, 2011.

25 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. *SIAM J. Comput.*, 47(3):675–702, 2018.

26 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*, 6:5–14, 1961.

27 William J. Masek. Some NP-complete set covering problems. Unpublished Manuscript, 1979.

28 Dylan M. McKay, Cody D. Murray, and R. Ryan Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *51st Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1215–1225, 2019.

29 Cody D. Murray and Richard Ryan Williams. On the (non) NP-hardness of computing circuit complexity. In *30th Conference on Computational Complexity (CCC)*, volume 33, pages 365–380, 2015.

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

31 Rahul Santhanam. Pseudorandomness and the minimum circuit size problem. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*, volume 151, pages 68:1–68:26, 2020.

32 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *19th Annual ACM Symposium on Theory of Computing (STOC)*, pages 77–82, 1987.

33 Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Annals of the History of Computing*, 6(4):384–400, 1984.

34 Christopher Umans, Tiziano Villa, and Alberto L. Sangiovanni-Vincentelli. Complexity of two-level logic minimization. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 25(7):1230–1246, 2006.

35 Ingo Wegener. *The Complexity of Boolean Functions*. Wiley-Teubner, 1987.

36 Ryan Williams. Personal Communication.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il