Addition Is Exponentially Harder Than Counting for Shallow Monotone Circuits*

Xi Chen†
Columbia University
New York, NY 10027 U.S.A.
xichen@cs.columbia.edu

Igor C. Oliveira‡
Charles University in Prague
Prague, Czech Republic
igorcarb@karlin.mff.cuni.cz

Rocco A. Servedio§
Columbia University
New York, NY 10027 U.S.A.
rocco@cs.columbia.edu

ABSTRACT

Let $\mathrm{Add}_{k,N}$ denote the Boolean function which takes as input k strings of N bits each, representing k numbers $a^{(1)}, \ldots, a^{(k)}$ in $\{0, 1, \ldots, 2^N - 1\}$, and outputs 1 if and only if $a^{(1)} + \cdots + a^{(k)} \geq 2^N$. Let $\mathsf{MAJ}_{t,n}$ denote a *monotone unweighted threshold gate*, i.e., the Boolean function which takes as input a single string $x \in \{0,1\}^n$ and outputs 1 if and only if $x_1 + \cdots + x_n \geq t$. The function $\mathrm{Add}_{k,N}$ may be viewed as a monotone function that performs addition, and $\mathsf{MAJ}_{t,n}$ may be viewed as a monotone gate that performs counting. We refer to circuits that are composed of MAJ gates as *monotone majority circuits*.

The main result of this paper is an exponential lower bound on the size of bounded-depth monotone majority circuits that compute $\mathrm{Add}_{k,N}$. More precisely, we show that for any constant $d \geq 2$, any depth-d monotone majority circuit that computes $\mathrm{Add}_{d,N}$ must have size $2^{\Omega(N^{1/d})}$. As $\mathrm{Add}_{k,N}$ can be computed by a single monotone *weighted* threshold gate (that uses exponentially large weights), our lower bound implies that constant-depth monotone majority circuits require exponential size to simulate monotone weighted threshold gates. This answers a question posed by Goldmann and Karpinski (STOC'93) and recently restated by Håstad (2010, 2014). We also show that our lower bound is essentially best possible, by constructing a depth-d, size-$2^{O(N^{1/d})}$ monotone majority circuit for $\mathrm{Add}_{d,N}$.

As a corollary of our lower bound, we significantly strengthen a classical theorem in circuit complexity due to Ajtai and Gurevich (JACM'87). They exhibited a monotone function that is in AC^0 but requires super-polynomial size for any constant-depth monotone circuit composed of unbounded fan-in AND and OR gates. We describe a monotone function that is in depth-3 AC^0 but requires *exponential* size monotone circuits of any constant depth, even if the circuits are composed of MAJ gates.

*The full version of the paper is available at [11].
†Supported in part by NSF grants CCF-1149257 and CCF-1423100.
‡Supported in part by CNPq grant 200252/2015-1.
§Supported in part by NSF grants CCF-1319788 and CCF-1420349.

STOC'17, Montreal, Canada

DOI: 10.1145/3055399.3055425

CCS CONCEPTS

• **Theory of computation → Circuit complexity**; *Algorithm design techniques*;

KEYWORDS

Circuit complexity, monotone circuit, unweighted threshold gate, weighted threshold function.

ACM Reference format:
Xi Chen, Igor C. Oliveira, and Rocco A. Servedio. 2017. Addition Is Exponentially Harder Than Counting for Shallow Monotone Circuits. In *Proceedings of 49th Annual ACM SIGACT Symposium on the Theory of Computing, Montreal, Canada, June 2017 (STOC'17),* 14 pages.
DOI: 10.1145/3055399.3055425

1 INTRODUCTION

> *"And you do Addition?" the White Queen asked.*
> *"What's one and one and one and one and one and one and one and one and one and one?"*
> *"I don't know," said Alice. "I lost count."*
> *"She can't do Addition," the Red Queen interrupted.*
>
> — Lewis Carroll, *Through the Looking Glass*

Threshold functions and threshold circuits. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is called a *weighted threshold function* (also known as a halfspace, weighted majority, weighted threshold gate or linear threshold function) if there exist integers $w_1, \ldots, w_n$ and t such that $f(x) = 1$ if and only if we have $w_1 x_1 + \cdots + w_n x_n \geq t$. The parameters $w_1, \ldots, w_n$ are called *weights*. We say that a threshold function f is *unweighted* if $|w_i| = 1$ for every $i \in \{1, \ldots, n\}$, and that it is *monotone* if every weight is non-negative. (Thus, a monotone unweighted threshold function is precisely a $\mathsf{MAJ}_{t,n}$ function as described in the abstract.) Threshold functions and their generalizations have been investigated for decades (see e.g. [12, 23, 24]) and arise in diverse areas including social choice theory [33], circuit complexity [6], structural complexity [7], learning theory [14], neural networks [26], cryptography [25], and many others.

In this work, we consider Boolean circuits that are composed of gates that compute threshold functions (i.e., *threshold gates*). (We refer to [21] as an extensive reference on Boolean functions and circuit complexity.) While individual threshold gates may appear relatively simple, Boolean circuits composed of these gates (i.e., *threshold circuits*) remain poorly understood despite intensive study. For instance, it is a notorious and long-standing open problem in complexity theory to prove the existence of a function in NP that cannot be computed by a depth-2 circuit with polynomially many

weighted threshold gates. This difficulty can be explained in part by the surprising computational power of bounded-depth threshold circuits, both in theory and practice. On the theory side, such circuits can efficiently implement all the basic arithmetic operations (see e.g., Table 1 in [29]) and can also simulate (in quasipolynomial size and depth 3) AND / OR / MOD_m Boolean circuits of much larger depth [2, 35]. On a more practical level, constant-depth networks of (continuous analogues of) threshold gates play a fundamental role in recent successful deep learning frameworks (see e.g. [28]).

Despite our inability to establish strong lower bounds against threshold circuits, there have been some notable successes in understanding the relative power of weighted versus unweighted threshold gates and circuits. Siu and Bruck [30] were the first to show that any weighted threshold gate can be simulated by a polynomial-size, constant-depth circuit consisting of unweighted threshold gates (such circuits are also known as *majority circuits*). This result was improved by Goldmann, Håstad, and Razborov in [15], who showed (non-constructively) that weighted threshold gates can be computed by polynomial-size majority circuits of depth 2; in fact, [15] showed that any depth-d weighted threshold circuit can be simulated efficiently by a depth-$(d+1)$ majority circuit. Soon thereafter Goldmann and Karpinski [16] gave a constructive proof with better parameters for the size of the resulting majority circuits. Subsequent simplifications and improvements of these simulations were given by Hofmeister [20], and Amano and Maruoka [4].

Monotone functions and monotone circuits. In a different, and highly successful, strand of research on circuit complexity, a wide range of lower bounds have been obtained against various types of *monotone* Boolean circuits (i.e., circuits that consist of AND / OR gates only but no negations). A sequence of well-known results [3, 5, 27, 32] culminated in the existence of explicit monotone Boolean functions that can be computed by polynomial-size Boolean circuits but require *monotone* circuits of exponential size. Analogous results highlighting the limitations of monotone circuits are also known at the "low-complexity" end of the spectrum: in an important result, Ajtai and Gurevich [1] exhibited a monotone function in AC^0 (i.e. a constant-depth, polynomial-size AND / OR / NOT Boolean circuit) that requires *monotone* AC^0 circuits (composed of AND / OR gates) to have super-polynomial size. However, it should be noted that the Ajtai–Gurevich circuit lower bound against monotone AC^0 is quantitatively not very strong (at best a quasipolynomial $n^{\Omega(\log n)}$ lower bound; see discussion following the statement of the Ajtai–Gurevich theorem below). Other works have given alternative or simplified expositions of the Ajtai–Gurevich lower bound and of its consequences in formal logic (see [10] for the former and [31] for the latter). But prior to the results of this paper, stronger lower bounds against monotone AC^0 circuits for monotone functions in AC^0 remained elusive.

This work: Monotone weighted threshold functions versus constant-depth monotone majority circuits. As mentioned earlier, Goldmann and Karpinski obtained a constructive proof in [16] that weighted threshold gates can be simulated by polynomial-size, depth-2 majority circuits. They also observed that even if the weighted threshold gate is monotone, known simulations produce majority circuits that are inherently *non-monotone* (i.e., they contain majority gates with negative weights, or equivalently, negation gates), which led them to ask the question of whether an efficient monotone simulation is possible in constant depth.

Hofmeister [19] made some early progress on this question by showing that any monotone depth-2 majority circuit that computes the function $\mathrm{Add}_{2,N}$ from the abstract must have exponential size. To state the result more precisely, let us first clearly specify our notion of monotone majority circuits. A monotone majority circuit here is a directed acyclic graph that may have multiple edges (called wires). There is a single node with no outgoing wires, called the output gate. Nodes that have no incoming wires are called input nodes and are each labeled either $0, 1$ or x_i for some i; every other node is labeled a monotone unweighted threshold gate $\mathsf{MAJ}_{t,m}$ for some t, with m being its in-degree, which outputs 1 if and only if there are at least t 1's from its m input wires. We say that the *size* of a monotone unweighted threshold gate $\mathsf{MAJ}_{t,m}$ is m (or its in-degree), and the size of a monotone majority circuit is the sum of the sizes of its gates (or its number of wires).[1] Then Hofmeister [19] showed that every depth-2 monotone majority circuit for $\mathrm{Add}_{2,N}$ has size $2^{\Omega(\sqrt{N})}$.

As mentioned above, in subsequent work [20] and [4], several improvements were made on the Goldmann–Karpinski simulation, but neither is monotone, and no further progress was obtained on the lower bound side after Hofmeister's result [19] until the current work. The question of Goldmann and Karpinski was recently restated by Håstad [9, 17].

1.1 Our Results

Our main result shows that monotone weighted threshold gates cannot be simulated by subexponential-size monotone majority circuits of constant depth. This may be viewed as an extension of Hofmeister's depth-2 lower bound [19] to arbitrary constant depth (in fact we obtain super-polynomial size lower bounds for circuits of small super-constant depth; see discussions after Theorem 1.1). We thus answer the question posed by Goldmann and Karpinski [16] and by Håstad [9, 17].

Before giving a precise statement of our results, we define formally the family $\mathrm{Add}_{k,N}$ of Boolean functions as described in the abstract. Given $t \geq 1$, we let $[t]$ denote the set $\{1, \ldots, t\}$. For $k \geq 2$, $\mathrm{Add}_{k,N}$ maps $\{0,1\}^{k \times N}$ to $\{0,1\}$ as follows. Given

$$x = (x_{i,j})_{i \in [k], j \in [N]} \in \{0,1\}^{k \times N},$$

we define

$$\mathrm{Sum}_{k,N}(x) \stackrel{\text{def}}{=} \sum_{j=1}^{N} 2^{N-j} \cdot (x_{1,j} + \cdots + x_{k,j}) \quad \text{and}$$

$$\mathrm{Add}_{k,N}(x) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \mathrm{Sum}_{k,N}(x) \geq 2^N \\ 0 & \text{otherwise.} \end{cases}$$

It is helpful to think of the input $x = (x_{i,j})$ as a k-row, N-column, and 0/1-valued matrix, where its ith row $(x_{i,1}, \ldots, x_{i,N})$ gives the binary representation of a number $x^{(i)} \in \{0, 1, \ldots, 2^N - 1\}$ in the usual way (with $x_{i,1}$ being the most significant bit). The function

[1] Observe that by reduplicating inputs, any weighted threshold function f given by $\sum_{i=1}^{n} w_i x_i \geq t$ can be computed by an unweighted threshold gate of size $|w_1| + \cdots + |w_n|$. We sometimes refer to this as the "weight of f."

$\mathrm{Sum}_{k,N}$ adds up the k numbers $x^{(1)},\ldots,x^{(k)}$ and $\mathrm{Add}_{k,N}$ outputs 1 if and only if the sum is at least 2^N.

With the definition of $\mathrm{Add}_{k,N}$ in place, our main result can be stated as follows:

Theorem 1.1. *Let d, n, and N be positive integers that satisfy $d \geq 2$, $n \geq 2^{60d}$, and $N \geq (2^{13}n)^d$. Then any depth-d monotone majority circuit computing $\mathrm{Add}_{d,N}$ must have size at least*

$$2^{n/(2^{60d})}.$$

This lower bound is nearly optimal for any fixed $d \geq 2$, as we prove the following upper bound.

Theorem 1.2. *Let $k, d, N \geq 2$ be three positive integers. Then there exists a depth-d monotone majority circuit of size at most*

$$2^{6(N^{1/d}\log k+\log N)}$$

that computes $\mathrm{Add}_{k,N}$.

Remark 1. *For any fixed constant $d \geq 2$, Theorems 1.1 and 1.2 together show that the smallest depth-d monotone majority circuit that computes $\mathrm{Add}_{d,N}$ (note that this function has $dN = \Theta(N)$ input variables) has size $\exp(\Theta(N^{1/d}))$. Moreover, by setting $d = c\sqrt{\log N}$ and $n = 2^{61d}$ for some small enough constant c satisfying $N \geq (2^{13}n)^d$, Theorem 1.1 implies that any depth-d monotone majority circuit computing $\mathrm{Add}_{d,N}$ has superpolynomial size (exponential in $2^{c\sqrt{\log N}}$).*

Remark 2. *As an easy consequence of Theorem 1.2, we also obtain a slightly weaker version of the main result of Beimel and Weinreb* [8]. *They showed that the "universal monotone threshold function"*[2] *$\mathrm{Add}_{O(N),O(N\log N)}$ can be computed by a $\mathrm{poly}(N)$-size and depth-$O(\log N)$ monotone circuit composed of fan-in two* AND *gates and unbounded fan-in* OR *gates. While our Theorem 1.2 above is tailored for small values of k, it implies that $\mathrm{Add}_{O(N),O(N\log N)}$ can be computed by a $\mathrm{poly}(N)$-size, depth-$O(\log^2 N)$ monotone circuit composed of fan-in two* AND / OR *gates only. (In more detail, it is enough to set $d = \log N$, and replace each majority gate by a $O(\log N)$-depth fan-in two* AND / OR *Boolean circuit.) We sketch a simpler construction in Appendix A that matches the parameters obtained in* [8] *in the case of the universal monotone threshold function.*

Another consequence of our lower bound as stated in Theorem 1.1 is a significant strengthening of the Ajtai–Gurevich lower bound [1] discussed earlier. We recall their result in more detail:

Theorem (Ajtai–Gurevich). *There exists an explicit sequence $f = \{f_n\}_{n\in\mathbb{N}}$ of monotone Boolean functions $f_n : \{0,1\}^n \to \{0,1\}$ such that: (i) $f \in \mathsf{AC}^0$, but (ii) $f \notin \mathsf{monAC}^0$: For any fixed constant d, any monotone depth-d* AND / OR *circuit computing f_n must have size at least $S_d(n)$, for some function $S_d(n) = n^{\omega(1)}$.*

Regarding part (*ii*) above, it is not immediately clear what is the best (largest) function $S_d(n)$ that can be extracted from the Ajtai–Gurevich proof. However, their f_n is easily seen to be computed by a monotone depth-2 circuit (a monotone DNF) of size $n^{\log n}$. Thus, we have $S_d(n) \leq n^{\log n}$ for all $d \geq 2$.

As an easy corollary of Theorem 1.1, we strengthen the Ajtai–Gurevich circuit lower bound (for a different monotone function in AC^0) in two ways: (1) by giving a lower bound against monotone *majority circuits* of constant depth (rather than monotone circuits of AND / OR gates only) and (2) by achieving an *exponential* size lower bound for any fixed depth (rather than a bound which is at most $n^{\log n}$). Our theorem is the following:

Theorem 1.3. *There exists an explicit sequence $g = \{g_n\}_{n\in\mathbb{N}}$ of monotone Boolean functions with $g_n : \{0,1\}^{n\log n} \to \{0,1\}$ such that: (i) $g \in \mathsf{AC}^0$ (in fact each g_n is computed by a $\mathrm{poly}(n)$-size, depth-3* AND / OR / NOT *circuit), but (ii) for any constant $d \geq 2$, every monotone depth-d majority circuit for g_n must have size $2^{\Omega(n^{1/d})}$.*

We establish Theorem 1.3 using Theorem 1.1 and the following lemma which we prove in Section 4.

Lemma 1.4. *For a suitable absolute constant $c : 0 < c < 1$, letting $k = (\log N)^c$, the function $\mathrm{Add}_{k,N}$ can be computed by a $\mathrm{poly}(N)$ size* AND / OR / NOT *circuit of depth* 3.

Proof of Theorem 1.3. We take $g_N \stackrel{\text{def}}{=} \mathrm{Add}_{k,N}$,[3] with k being $(\log N)^c$, where $c : 0 < c < 1$ is the constant in Lemma 1.4. Then Part (*i*) of the theorem follows from Lemma 1.4. Part (*ii*) follows from our main lower bound, Theorem 1.1, by observing that any circuit for $\mathrm{Add}_{k,N}$ yields a circuit for $\mathrm{Add}_{d,N}$ (by setting the last $k-d$ rows of the input to 0) since d is a fixed constant. □

It is interesting to note that our proof of Theorem 1.1 uses very different arguments from those of Ajtai and Gurevich. The heart of their proof is a "switching lemma" for monotone functions on hypergrids (see the excellent exposition of their proof given in [10]) whereas our approach does not use switching lemmas at all.

1.2 Related Work and Our Techniques

In addition to papers discussed above, the works of Yao [34] and Håstad and Goldmann [18] are relevant in the context of our lower bound result. Let Sipser_{d+1} be the read-once monotone n-variable formula of depth $d+1$ that has alternating layers of AND and OR gates (see [18] for a detailed description of this function). Strengthening the earlier result of [34], [18] showed that a depth-d circuit of *weighted* monotone threshold gates computing Sipser_{d+1} must have size at least $2^{\Omega(n^{1/2d})}$. In contrast, our Theorem 1.1 only establishes a lower bound against constant-depth monotone circuits of *unweighted* threshold gates, but — crucially — we establish the lower bound for a much "simpler" monotone function, i.e., $\mathrm{Add}_{d,N}$, which is computed by a *single* weighted monotone threshold gate. Indeed, the main challenge of our work is to push through a lower bound for such a heavily constrained target function.

Sketch of the upper bound construction. We give a brief sketch of the upper bound construction employed in the proof of Theorem 1.2. Assume for simplicity that $N = n^d$. We view the input string $(x_{i,j}) \in \{0,1\}^{k\times N}$ as a k-row, N-column, and 0/1-valued matrix. Recall that $\mathrm{Add}_{k,N}(x) = 1$ iff the addition of the k numbers encoded in x produces a "carry" bit. Our depth-d circuit is described by a recursive construction that computes generalized carry-bit functions $c^{(\gamma)}_{\alpha,\beta}$ over distinct blocks of columns $S_\gamma \subseteq [N]$ of the original input matrix $(x_{i,j})$. Intuitively, $c^{(\gamma)}_{\alpha,\beta}(x) = 1$ if and only if a carry

[2] It is called the universal monotone threshold function because it can simulate any monotone weighted threshold function over N inputs.

[3] One can add dummy input variables to make the number of input bits $N\log N$.

of value at least α is generated when adding the numbers in the submatrix corresponding to S_γ, assuming this block of variables receives a carry of value β from the block of variables to the right.

We start with a decomposition of the column set $[N]$ into n^{d-1} blocks of length n, and compute in a single layer (and in parallel) the values of the corresponding generalized carry-bit functions described above. It is not hard to see that each function $c^{(\gamma)}_{\alpha,\beta}$ can be computed by a threshold gate whose total weight is at most

$$O(k \cdot 2^n) = O(k \cdot 2^{N^{1/d}}),$$

using $N = n^d$ and $0 \le \alpha, \beta \le k-1$. The base case is particularly simple, since these gates will directly depend on the input bits $x_{i,j}$.

In the recursive step of the construction, n adjacent blocks of columns considered in the previous step are merged, and we compute new generalized carry bit functions under a different partition of the column set $[N]$. Observe that after d steps there is only one block of columns, and the circuit will in particular compute whether the sum of the original numbers in x produces a carry bit. Implementing this recursive step is slightly more complicated, as the new gates added to the circuit will rely on the generalized carry bit functions associated to the column decomposition employed in the layer of gates immediately below.

For technical reasons, in order to prove the correctness of the recursion, we assume a certain lower bound on the size of each initial block of columns as a function of N, k, and d. Furthermore, the upper bound in Theorem 1.2 works for any choice of parameters. We refer to the proof of Theorem 1.2 for more details.

Sketch of the lower bound proof. Now we briefly discuss some of the high-level ideas behind the proof of Theorem 1.1. At the heart of the proof is a sequence of carefully (and inductively) constructed pairs of distributions denoted by $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$, for $\ell = 1, \ldots, d$, both of which are over $\{0,1\}^{(\ell+1)\times N_\ell}$ (i.e., over possible inputs to the function $\mathrm{Add}_{\ell+1,N_\ell}$, for some N_ℓ to be specified later). The first distribution $\mathcal{YES}_\ell$ in the pair is supported on strings x that have $\mathrm{Add}_{\ell+1,N_\ell}(x) = 1$, and $\mathcal{NO}_\ell$ is supported on x with $\mathrm{Add}_{\ell+1,N_\ell}(x) = 0$. The key property of these pairs of distributions, which yields our lower bound, is that considered together, each pair of $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ is "hard" for "small" monotone majority circuits of depth ℓ in a suitable sense. In a bit more detail, we prove inductively that for any "small" depth-ℓ circuit F, we have

$$\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{YES}_\ell}\left[F(\boldsymbol{x}) = 1\right] + \mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{NO}_\ell}\left[F(\boldsymbol{y}) = 0\right] \le 1 + \tau_\ell, \quad (1)$$

for a suitable $0 < \tau_\ell \ll 1$. As long as (1) holds for some $\tau_\ell < 1$, F cannot correctly computes $\mathrm{Add}_{\ell+1,N_\ell}$ on all inputs. One can use (1) to conclude that no "small" depth-d monotone majority circuit can compute Add_{d+1,N_d}. (Note that this is slightly weaker than the claim of Theorem 1.1 about depth-d circuits against the addition of d numbers instead of $d+1$; see footnote 4.)

Since the goal of $\mathcal{YES}_\ell$ and $\mathcal{NO}_\ell$ is to work against monotone circuits, a natural design principle is to have the first distribution $\mathcal{YES}_\ell$ supported on minimal 1-strings x and the second distribution $\mathcal{NO}_\ell$ supported on maximal 0-strings y of $\mathrm{Add}_{\ell+1,N_\ell}$, which means that the sums of x and y are 2^{N_ℓ} and $2^{N_\ell}-1$, respectively. This will be the case for our construction. While the two distributions $\mathcal{YES}_\ell$ and $\mathcal{NO}_\ell$ will look quite different, we show that no "small" depth-ℓ monotone majority circuit can distinguish them.

We establish (1) via an inductive argument on ℓ. It is not too difficult to show that the base case $\ell = 1$ holds; this is presented in Lemma 2.1. We explain the strategy used in the proof of Lemma 2.1 since similar high-level ideas are also used in the main induction, though in a more sophisticated fashion. We start with the definition of $\mathcal{YES}_1$ and $\mathcal{NO}_1$. We view a string $x \in \{0,1\}^{2\times N_1}$ as a $2\times N_1$ matrix with the ith column containing $x_{1,i}$ and $x_{2,i}$. To draw an $\boldsymbol{x}$ from $\mathcal{YES}_1$, one first draws a random column $\boldsymbol{R}$ uniformly from $[N_1]$. Then the $\boldsymbol{R}$-th column of $\boldsymbol{x}$ is set to $(1,1)$, each j-th column with $j < \boldsymbol{R}$ is set to $(0,1)$ or $(1,0)$ with equal probability $1/2$, and each column with $j > \boldsymbol{R}$ is set to $(0,0)$. To draw a $\boldsymbol{y}$ from $\mathcal{NO}_1$, one simply sets each column independently to $(0,1)$ or $(1,0)$ with equal probability. (See Section 2.1.) To prove (1) against any "small" majority gate F, we define a coupling of the two distributions as follows: $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{R}) \sim \mathcal{D}$, where $\boldsymbol{z}$ is drawn from $\mathcal{NO}_1$, $\boldsymbol{R}$ is uniform over $[N_1]$, $\boldsymbol{x}$ is obtained from $\boldsymbol{z}$ and $\boldsymbol{R}$ by changing the $\boldsymbol{R}$-th column of $\boldsymbol{z}$ to $(1,1)$ and columns after $\boldsymbol{R}$ to $(0,0)$, and $\boldsymbol{y}$ is obtained from $\boldsymbol{z}$ and $\boldsymbol{R}$ by flipping the $\boldsymbol{R}$-th column of $\boldsymbol{z}$. (So this way the marginal distributions of $\boldsymbol{x}$ and $\boldsymbol{y}$ are $\mathcal{YES}_1$ and $\mathcal{NO}_1$, respectively.)

Let F be a majority gate with a non-negative integer weight $w_{i,j}$ over each of the $2N_1$ inputs. The high-level strategy is to show that if $F(\boldsymbol{x}) = 1$ and $F(\boldsymbol{y}) = 0$ with high probability conditioning on $\boldsymbol{R} = r$, then a certain event about $\boldsymbol{z}$ must occur at r with high probability. On the other hand, using the size bound on F, we show that such events about $\boldsymbol{z}$ can only occur at a small number of values of $r \in [N_1]$. As a result, F cannot be correct on both $\boldsymbol{x} \sim \mathcal{YES}_1$ and $\boldsymbol{y} \sim \mathcal{NO}_1$ with high probability. More specifically, conditioning on $\boldsymbol{R} = r$, a necessary condition on $\boldsymbol{z}$ for F to be correct on both $\boldsymbol{x}$ and $\boldsymbol{y}$ (obtained from $\boldsymbol{z}$ and r in the coupling) is that the weight of the variables in the r-th column of $\boldsymbol{z}$ (i.e., $w_{1,r}\boldsymbol{z}_{1,r} + w_{2,r}\boldsymbol{z}_{2,r}$) must be strictly larger than the total weight of $\boldsymbol{z}$ from columns to the right of r. This follows by comparing the total weights of $\boldsymbol{x}$ and $\boldsymbol{y}$. Intuitively this can only happen at a small number of r's since every time this happens, the total weight of $\boldsymbol{z}$ from columns $j \ge r$ gets doubled compared to that from columns $j \ge r+1$; on the other hand, since F has small weight, the total weight of $\boldsymbol{z}$ must be small. This finishes our sketch of the proof of Lemma 2.1.

Compared to Lemma 2.1 for the base case of $\ell = 1$, significant care is required to carry out the inductive step. In fact, in order for the inductive hypothesis to be "strong enough to prove itself," we require an analogue of (1) both for the pair $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ and for a second pair of distributions $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$, where each of the two distributions $\mathcal{YES}'_\ell$ and $\mathcal{NO}'_\ell$, like $\mathcal{YES}_\ell$ and $\mathcal{NO}_\ell$, is supported on $\{0,1\}^{(\ell+1)\times N_\ell}$. All these distributions are highly structured; without going into too much detail for now, we mention that every x in the support of $\mathcal{YES}_\ell, \mathcal{NO}_\ell, \mathcal{YES}'_\ell, \mathcal{NO}'_\ell$ has

$$\mathrm{Sum}_{\ell+1,N_\ell}(x) = 2^{N_\ell}, 2^{N_\ell}-1, 2^{N_\ell}-1, \text{ and } 2^{N_\ell}-(\ell+1),$$

respectively. (This tight structure is essential in enabling the induction to go through.) We further mention that the inductive argument that establishes the case ℓ from the case $\ell-1$ requires careful analysis of yet a third carefully constructed pair $(\mathcal{YES}^*_\ell, \mathcal{NO}^*_\ell)$ of distributions. Roughly speaking, the $\mathcal{YES}^*_\ell$ distribution is an amalgamation of n copies of the $\mathcal{YES}'_{\ell-1}$, $\mathcal{NO}_{\ell-1}$, and $\mathcal{YES}_{\ell-1}$ distributions, and the $\mathcal{NO}^*_\ell$ distribution is an amalgamation of n copies of the $\mathcal{YES}'_{\ell-1}$, $\mathcal{NO}_{\ell-1}$, and $\mathcal{NO}'_{\ell-1}$ distributions; see the

discussion below and Figure 1 for a visual depiction of the construction. The $\mathcal{YES}'_\ell$, $\mathcal{NO}'_\ell$, $\mathcal{YES}_\ell$ and $\mathcal{NO}_\ell$ distributions may in turn (for each $\ell > 1$) be viewed as relatively simple augmentations of $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$; see Figure 2. See Section 2 for full details. [4]

We give some intuition behind the inductive step. Using the inductive hypothesis, we have that both pairs $(\mathcal{YES}_{\ell-1}, \mathcal{NO}_{\ell-1})$ and $(\mathcal{YES}'_{\ell-1}, \mathcal{NO}'_{\ell-1})$ are hard for small depth-$(\ell-1)$ circuits in the sense given in (1), for some small $\tau_{\ell-1}$. Now we define $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ over $\{0,1\}^{\ell \times N_\ell}$, with $N_\ell = nN_{\ell-1}$. To this end, we view an input as comprised of n sections, each section being an input string from $\{0,1\}^{\ell \times N_{\ell-1}}$ (see Figure 1). A draw of $\boldsymbol{x} \sim \mathcal{YES}^*_\ell$ is obtained by first uniformly sampling a section $\mathbf{T} \in [n]$, and for each section $j < \mathbf{T}$ sample a string from $\mathcal{NO}_{\ell-1}$ with probability $1/2$ and a string from $\mathcal{YES}'_{\ell-1}$ with probability $1/2$; for the $\mathbf{T}$-th section sample a string from $\mathcal{YES}_{\ell-1}$; and for each section $j > \mathbf{T}$ set every bit to be 0. To draw a $\boldsymbol{y} \sim \mathcal{NO}^*_\ell$, everything is the same except that for the $\mathbf{T}$-th section we sample a string from $\mathcal{NO}'_{\ell-1}$, and for each section $j > \mathbf{T}$ we set every bit to be 1.

Let F be a depth-ℓ circuit. Let $H_1, \ldots, H_m$ be the inputs of the output gate of F (with multiplicities) and h be the threshold (so F outputs 1 when at least h of $H_1, \ldots, H_m$ output 1). As F is a "small" depth-ℓ circuit, m is small and each H_i is also computed by a small depth-$(\ell-1)$ circuit. We now use the same high-level strategy as the base case (Lemma 2.1) to show that such an F cannot be correct on both $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ with high probability, by showing (a) if F is correct on both distributions ($F(\boldsymbol{x}) = 1$ when $\boldsymbol{x} \sim \mathcal{YES}^*_\ell$ and $F(\boldsymbol{y}) = 0$ when $\boldsymbol{y} \sim \mathcal{NO}^*_\ell$) with high probability, conditioning on $\mathbf{T} = t$ for some specific $t \in [n]$, then some event must occur at t with a decent probability; (b) such events cannot occur too many times as m is small.

For (a) we take the simplest case of $\mathbf{T} = 1$ as an example, and assume for now that $h < m/2$. If F is correct on $\boldsymbol{y} \sim \mathcal{NO}^*_\ell$ (conditioning on $\mathbf{T} = 1$) with high probability, one can use an averaging argument to show that a large fraction of H_i output 0 with a decent probability on $\boldsymbol{y}$, which can be written as $\boldsymbol{y}_1 \circ \mathbf{1}$ with $\boldsymbol{y}_1 \sim \mathcal{NO}'_{\ell-1}$ and $\mathbf{1}$ being the all-1 string in the remaining $n-1$ sections. However, this means that the probability of such an H_i being correct (i.e. outputting 1) on $\boldsymbol{x}_1 \circ \mathbf{1}$ with $\boldsymbol{x}_1 \sim \mathcal{YES}'_{\ell-1}$ cannot be very close to 1; otherwise we can hardwire the last $n-1$ sections of H_i to 1 and obtain a small depth-$(\ell-1)$ circuit that performs well on the pair of distributions $(\mathcal{YES}'_{\ell-1}, \mathcal{NO}'_{\ell-1})$, contradicting with the inductive hypothesis. As a result, when a string $\boldsymbol{x}_1$ is drawn from $\mathcal{YES}'_{\ell-1}$, we expect to see a large fraction of H_i output 0 on $\boldsymbol{x}_1 \circ \mathbf{1}$. However, this means that by applying a restriction that fixes the first section to be $\boldsymbol{x}_1$, such H_i would always output 0 and become trivial (so we can remove them): if $H_i(\boldsymbol{x}_1 \circ \mathbf{1}) = 0$, then after the restriction H_i always outputs 0 because it is a *monotone* circuit.

Note that in the sketch above we did not assume that F works well on $\mathcal{YES}^*_\ell$; this is used only in the other case when $h \geq m/2$. Combining these two cases, we conclude that (a) if F works well for both $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ when $\mathbf{T} = t$ for some specific t, then one can apply a random restriction on the t-th section to remove a decent fraction of H_i's. Since m is small, we have (b) the latter can only happen for a small number of times. The inductive step follows by combining (a) and (b).

[4] Notice that the argument we just sketched implies that $\mathrm{Add}_{d+1,N}$ is hard against depth-d circuits. A more careful analysis at the end of the argument using the distributions $(\mathcal{YES}^*_d, \mathcal{NO}^*_d)$ instead of $(\mathcal{YES}_d, \mathcal{NO}_d)$ would allow us to obtain the same lower bound for adding d numbers instead, as stated in Theorem 1.1.

Notation and Organization. Recall that a *restriction* ρ of a function f is an assignment fixing some of the input variables of f. We write "$f \restriction \rho$" to denote f restricted by ρ, a new function over the rest of variables. We use boldface, lower-case letters such as $\boldsymbol{x}$ and $\boldsymbol{y}$ to denote string-valued random variables, and use boldface, capital letters such as $\mathbf{X}$ and $\mathbf{Y}$ to denote real-valued random variables.

The rest of the paper is organized as follows. We prove Theorem 1.1 in Section 2 and Theorem 1.2 in Section 3, respectively. The proof of Lemma 1.4, which completes the proof of Theorem 1.3, can be found in Section 4. We sketch a construction of monotone circuits for the universal monotone threshold function that matches the parameters obtained by Beimel and Weinreb [8] in the appendix.

2 THE LOWER BOUND

We prove Theorem 1.1 in this section. Throughout the section we use d, n and N to denote the three positive integers in the statement of Theorem 1.1 with $d \geq 2$, $n \geq 2^{60d}$ and $N \geq (2^{13}n)^d$.

This section is organized as follows. In Sections 2.1 and 2.2 we define inductively two pairs $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ and $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ of distributions over strings $\{0,1\}^{(\ell+1)\times N_\ell}$ for ℓ from 1 to d, where N_ℓ is specified later and satisfies $N_1 < \cdots < N_d \leq N$. An important property of these distributions is that every $\boldsymbol{x}$ drawn from $\mathcal{YES}_\ell$, $\mathcal{NO}_\ell$, $\mathcal{YES}'_\ell$, $\mathcal{NO}'_\ell$ has $\mathrm{Sum}_{\ell+1,N_\ell}(\boldsymbol{x})$ equal to

$$2^{N_\ell}, \quad 2^{N_\ell} - 1, \quad 2^{N_\ell} - 1 \quad \text{and} \quad 2^{N_\ell} - (\ell+1),$$

respectively. From the definition of $(\mathcal{YES}_1, \mathcal{NO}_1)$, $(\mathcal{YES}'_1, \mathcal{NO}'_1)$ it is not too difficult to show that both pairs of distributions are *very hard* for monotone depth-1 majority circuits (Lemma 2.1), i.e. no majority gate with small weights can output 1 on strings drawn from $\mathcal{YES}_1$ with probability p_1 and at the same time output 0 on strings drawn from $\mathcal{NO}_1$ with probability p_2 such that $p_1 + p_2$ is slightly larger than 1 (and the same holds for $\mathcal{YES}'_1$ and $\mathcal{NO}'_1$). As we mentioned in the proof sketch, the proof of Lemma 2.1 serves as a warm-up since it is simple but shares some ideas that will be used later in the more challenging proof of the induction step.

Then we prove our main technical lemma in Section 2.3 (Lemma 2.4), which shows by induction that both pairs $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ and $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ are hard in the same sense for small depth-ℓ majority circuits over $\{0,1\}^{(\ell+1)\times N_\ell}$ for each $\ell \in [d]$, with $(\mathcal{YES}_1, \mathcal{NO}_1)$ and $(\mathcal{YES}'_1, \mathcal{NO}'_1)$ serving as the base case (a bit more formally, in order to conclude this we also need Lemmas 2.2 and 2.3). Theorem 1.1 for $\mathrm{Add}_{d+1,N}$ (instead of $\mathrm{Add}_{d,N}$ as stated) follow directly from $N_d \leq N$ and the property that strings $\boldsymbol{x}$ drawn from $\mathcal{YES}_d$ and $\mathcal{NO}_d$ have $\mathrm{Sum}_{d+1,N_d}(\boldsymbol{x})$ equal to 2^{N_d} and $2^{N_d} - 1$, respectively. (Note that, although the second pair $(\mathcal{YES}'_d, \mathcal{NO}'_d)$ is not needed in the proof of Theorem 1.1 once Lemma 2.4 has been established, the intermediate pairs $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ do play a crucial role in the inductive definition of these distributions and the proof.)

In order to extend the result to $\mathrm{Add}_{d,N}$ (as stated in Theorem 1.1) we rely on another auxiliary pair of distributions $(\mathcal{YES}^*_d, \mathcal{NO}^*_d)$ constructed during the proof, which is described in Section 2.2. We finally use Lemma 2.4 to prove Theorem 1.1 in Section 2.4.

2.1 The Initial Two Pairs of Distributions

Let d, n, N be positive integers in the statement of Theorem 1.1. Let

$$\varepsilon = 2^{-12d} \quad \text{and} \quad N_1 = n \cdot (1/\varepsilon).$$

Given $z \in \{0,1\}^{2\times N_1}$, the *$j$-th column of z* corresponds to a pair of positions $(1,j)$ and $(2,j)$, $j \in [N_1]$.

We define two pairs of probability distributions $(\mathcal{YES}_1, \mathcal{NO}_1)$ and $(\mathcal{YES}'_1, \mathcal{NO}'_1)$ over $\{0,1\}^{2\times N_1}$, and show that they are hard for monotone depth-1 majority circuits of not-too-large size. We define the distributions via the following sampling processes.

- A string $\boldsymbol{x} \sim \mathcal{YES}_1$ is generated as follows. Let $\boldsymbol{R} \sim [N_1]$ be uniformly random. We set both bits in the $\boldsymbol{R}$-th column of $\boldsymbol{x}$ to 1. For every $j > \boldsymbol{R}$, we set both bits in the j-th column of $\boldsymbol{x}$ to 0. For every $j < \boldsymbol{R}$, we set the j-th column of $\boldsymbol{x}$ to $(1,0)$ or $(0,1)$ independently and with equal probability. For example, writing an $x \in \mathrm{supp}(\mathcal{YES}_1)$ as a matrix, it would look like
$$x = \left[\begin{array}{ccccccc|c|ccccc} 1 & 0 & 0 & 1 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 1 & 0 & \cdots & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \end{array}\right]$$
and we have $\mathrm{Sum}_{2,N_1}(x) = 2^{N_1}$.
- A string $\boldsymbol{y} \sim \mathcal{NO}_1$ is generated by setting its j-th column to $(1,0)$ or $(0,1)$ independently and with equal probability for each $j \in [N_1]$. So a $y \in \mathrm{supp}(\mathcal{NO}_1)$ would look like
$$y = \begin{array}{cccccccccccc} 0 & 1 & 0 & 0 & \cdots & 1 & 1 & 0 & 1 & \cdots & 1 & 0 \\ 1 & 0 & 1 & 1 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 1 \end{array}$$
and we have $\mathrm{Sum}_{2,N_1}(y) = 2^{N_1} - 1$. $\mathcal{YES}'_1$ is the same as $\mathcal{NO}_1$. So each string $x \in \mathrm{supp}(\mathcal{YES}'_1)$ has sum $2^{N_1} - 1$.
- Finally, a $\boldsymbol{y} \sim \mathcal{NO}'_1$ is obtained as follows. First, sample a random $\boldsymbol{x} \sim \mathcal{YES}_1$. Then let $\boldsymbol{y}$ be the string obtained by negating each bit of $\boldsymbol{x}$. So a $y \in \mathrm{supp}(\mathcal{NO}'_1)$ looks like
$$y = \left[\begin{array}{ccccccc|c|ccccc} 1 & 1 & 0 & 1 & \cdots & 0 & 0 & 1 & 1 & \cdots & 1 & 1 \\ 0 & 0 & 1 & 0 & \cdots & 1 & 0 & 1 & 1 & \cdots & 1 & 1 \end{array}\right]$$
and we have $\mathrm{Sum}_{2,N_1}(y) = 2^{N_1} - 2$.

Recall that a monotone depth-1 majority circuit of size s is just a monotone weighted majority gate with total weight s. We show in the following lemma that both pairs of distributions $(\mathcal{YES}_1, \mathcal{NO}_1)$ and $(\mathcal{YES}'_1, \mathcal{NO}'_1)$ defined above are hard for a monotone depth-1 circuit (to be correct on both $\mathcal{YES}_1$ and $\mathcal{NO}_1$, or on both $\mathcal{YES}'_1$ and $\mathcal{NO}'_1$, with nontrivial probability) unless the weight s is large.

Lemma 2.1. *For any depth-1 monotone majority circuit F over $\{0,1\}^{2\times N_1}$ of size at most 2^{n-1}, we have*

$$\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{YES}_1}\left[F(\boldsymbol{x}) = 1\right] + \mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{NO}_1}\left[F(\boldsymbol{y}) = 0\right] \le 1 + \varepsilon, \quad (2)$$

$$\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{YES}'_1}\left[F(\boldsymbol{x}) = 1\right] + \mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{NO}'_1}\left[F(\boldsymbol{y}) = 0\right] \le 1 + \varepsilon. \quad (3)$$

Proof. We present the proof of the first inequality for the pair $(\mathcal{YES}_1, \mathcal{NO}_1)$. A similar argument works for $(\mathcal{YES}'_1, \mathcal{NO}'_1)$.

Consider an auxiliary distribution $\mathcal{D}$ (essentially a coupling of $\mathcal{YES}_1$ and $\mathcal{NO}_1$) supported over

$$\{0,1\}^{2\times N_1} \times \{0,1\}^{2\times N_1} \times \{0,1\}^{2\times N_1} \times [N_1]$$

and defined in the following way. A draw $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{R}) \sim \mathcal{D}$ is obtained by selecting a uniformly random $\boldsymbol{R} \sim [N_1]$, a string $\boldsymbol{z} \sim \mathcal{NO}_1$, letting $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{z}, \boldsymbol{R}) \in \{0,1\}^{2\times N_1}$ be the string obtained by replacing the $\boldsymbol{R}$-th column of $\boldsymbol{z}$ by $(1,1)$ and setting the j-th column of $\boldsymbol{z}$ to $(0,0)$ for every $j > \boldsymbol{R}$, and letting $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{z}, \boldsymbol{R})$ be the string obtained from $\boldsymbol{z}$ by flipping the two bits of $\boldsymbol{z}$ in the $\boldsymbol{R}$-th column. Observe that the marginal distributions $\mathcal{D}_{\boldsymbol{x}}$ and $\mathcal{D}_{\boldsymbol{y}}$ are identical to $\mathcal{YES}_1$ and $\mathcal{NO}_1$, respectively. Thus, the LHS of Equation (2) is equal to

$$\begin{aligned} &\mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{R})\sim\mathcal{D}}\left[F(\boldsymbol{x}) = 1\right] + \mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{R})\sim\mathcal{D}}\left[F(\boldsymbol{y}) = 0\right] \\ &\quad = \mathbf{Pr}\left[F(\boldsymbol{x}) = 1 \text{ or } F(\boldsymbol{y}) = 0\right] + \mathbf{Pr}\left[F(\boldsymbol{x}) = 1 \text{ and } F(\boldsymbol{y}) = 0\right] \\ &\quad \le 1 + \mathbf{Pr}\left[F(\boldsymbol{x}) = 1 \text{ and } F(\boldsymbol{y}) = 0\right]. \end{aligned}$$

Hence to prove the lemma, it is enough to show that

$$q \stackrel{\text{def}}{=} \mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{R})\sim\mathcal{D}}\left[F(\boldsymbol{x}) = 1 \text{ and } F(\boldsymbol{y}) = 0\right] \le \varepsilon. \quad (4)$$

For each $r \in [N_1]$, let $\mathbf{Z}_r$ denote an indicator random variable defined on $\mathcal{D}$ that is 1 whenever

$$w_r(\boldsymbol{z}) \;>\; \sum_{\ell > r} w_\ell(\boldsymbol{z}),$$

where we have

$$w_j(z) \stackrel{\text{def}}{=} w_{1,j} \cdot z_{1,j} + w_{2,j} \cdot z_{2,j},$$

and $w_{i,j}$ is the weight corresponding to the input variable of F at position (i,j). Equivalently, $\mathbf{Z}_r = 1$ if and only if the weight of $\boldsymbol{z}$ with respect to F at the r-th column is strictly larger than the sum of the weights collected from all succeeding columns.

We will employ the following claim to establish Equation (4).

Claim 1. *For every $j \in [N_1]$, we have*

$$q_j \stackrel{\text{def}}{=} \mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{R})\sim\mathcal{D}}\left[F(\boldsymbol{x}) = 1 \text{ and } F(\boldsymbol{y}) = 0 \,\middle|\, \boldsymbol{R} = j\right] \le \mathbf{Pr}_{\mathcal{D}}\left[\mathbf{Z}_j = 1\right].$$

Proof. We consider first the case where $j = 1$. The conditions of $F(\boldsymbol{x}) = 1$ and $\boldsymbol{R} = 1$ imply that $w_{1,1} + w_{2,1} \ge t$, where t denotes the threshold of F. Furthermore, because $F(\boldsymbol{y}) = 0$, it must be the case that $\sum_{r=1}^{N_1} w_r(\boldsymbol{y}) < t$. These inequalities give us

$$w_{1,1} + w_{2,1} - w_1(\boldsymbol{y}) > \sum_{r>1} w_r(\boldsymbol{y}). \quad (5)$$

The LHS is exactly $w_1(\boldsymbol{z})$ and the RHS is the same as $\sum_{r>1} w_r(\boldsymbol{z})$ since $\boldsymbol{y}$ is obtained by flipping the first column of $\boldsymbol{z}$. Therefore,

$$q_1 \le \mathbf{Pr}_{(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\boldsymbol{R})\sim\mathcal{D}}\left[w_1(\boldsymbol{z}) > \sum_{r>1} w_r(\boldsymbol{z}) \,\middle|\, \boldsymbol{R} = 1\right] = \mathbf{Pr}_{\mathcal{D}}\left[\mathbf{Z}_1 = 1\right],$$

where the last equation used the independence of $\boldsymbol{z}$ and $\boldsymbol{R}$.

For the general case of $j > 1$ the claim can be proved similarly by adapting the argument in the natural way. □

Claim 1 and the definitions of probabilities q and q_j imply that

$$N_1 \cdot q \;=\; \sum_{j=1}^{N_1} q_j \;\le\; \sum_{j=1}^{N_1} \mathbf{Pr}_{\mathcal{D}}\left[\mathbf{Z}_j = 1\right] \;=\; \mathbf{E}_{\mathcal{D}}\left[\sum_{j=1}^{N_1} \mathbf{Z}_j\right].$$

In particular, there is a string $z^* \in \mathrm{supp}(\mathcal{NO}_1)$ and a set $S \subseteq [N_1]$ with $|S| \ge N_1 \cdot q$ such that

$$w_r(z^*) > \sum_{\ell > r} w_\ell(z^*), \quad (6)$$

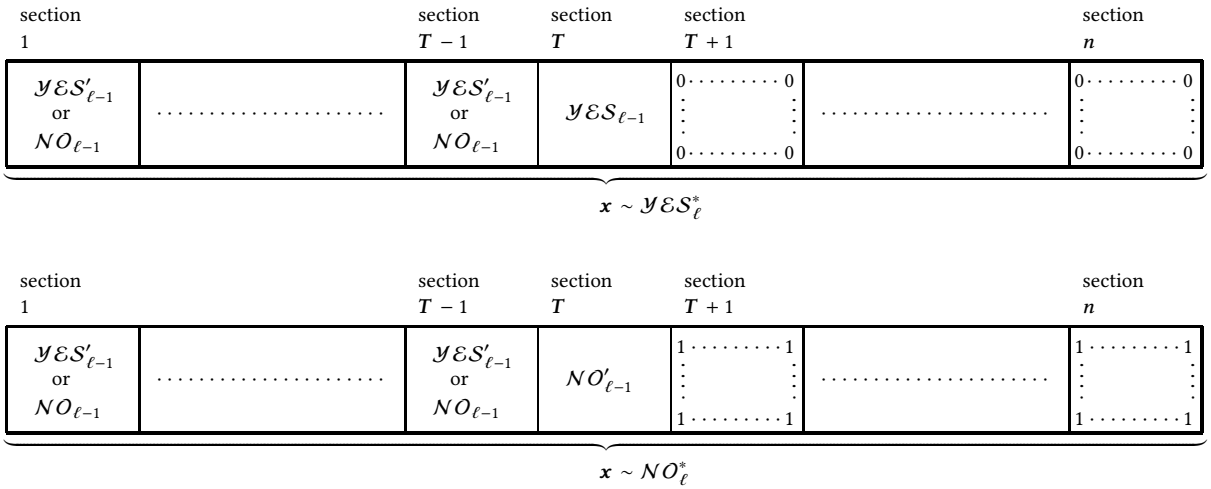

Figure 1: Illustrations of how the $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ distributions are defined from the $\mathcal{YES}'_{\ell-1}, \mathcal{NO}'_{\ell-1}, \mathcal{YES}_{\ell-1}$ and $\mathcal{NO}_{\ell-1}$ distributions.

for each $r \in S$. Recall that the weight associated with each variable in F is a non-negative integer (this is the place where we use the assumption that F is monotone), and note that the total weight of F is at least $\sum_{r\geq 1} w_r(z^*)$. It follows from (6) that F has total weight at least $2^{|S|-1}$. However, by our assumption, the total weight of F is at most 2^{n-1} (this is the place where we use the assumption that F has low weight) . Altogether, we get from these inequalities and $N_1 = n \cdot (1/\varepsilon)$ that $q \leq \varepsilon$, which completes the proof. □

2.2 A Sequence of Pairs of Pairs of Distributions

Suppose that we have defined two pairs of distributions $(\mathcal{YES}_{\ell-1}, \mathcal{NO}_{\ell-1})$ and $(\mathcal{YES}'_{\ell-1}, \mathcal{NO}'_{\ell-1})$ over $\{0,1\}^{\ell \times N_{\ell-1}}$ for some $\ell : 2 \leq \ell \leq d$, where a string $\boldsymbol{x}$ drawn from $\mathcal{YES}_{\ell-1}$, $\mathcal{NO}_{\ell-1}$, $\mathcal{YES}'_{\ell-1}$ and $\mathcal{NO}'_{\ell-1}$ has $\mathrm{Sum}_{\ell,N_{\ell-1}}(\boldsymbol{x})$ equal to

$$2^{N_{\ell-1}}, \quad 2^{N_{\ell-1}} - 1, \quad 2^{N_{\ell-1}} - 1, \quad \text{and} \quad 2^{N_{\ell-1}} - ((\ell-1)+1), \quad (7)$$

respectively. Note that the pairs $(\mathcal{YES}_1, \mathcal{NO}_1)$ and $(\mathcal{YES}'_1, \mathcal{NO}'_1)$ have this property. Our aim is to inductively define $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ and $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ over $\{0,1\}^{(\ell+1)\times N_\ell}$, where

$$N_\ell \stackrel{\text{def}}{=} n \cdot N_{\ell-1} + 1 \leq 2^\ell n^{\ell-1} N_1 = (2n)^\ell 2^{12d} \leq (2^{13}n)^d \leq N,$$

for $\ell \in \{2,\ldots,d\}$, and an $\boldsymbol{x}$ drawn from $\mathcal{YES}_\ell$, $\mathcal{NO}_\ell$, $\mathcal{YES}'_\ell$ and $\mathcal{NO}'_\ell$ has $\mathrm{Sum}_{\ell+1,N_\ell}(\boldsymbol{x})$ equal to

$$2^{N_\ell}, \quad 2^{N_\ell} - 1, \quad 2^{N_\ell} - 1, \quad \text{and} \quad 2^{N_\ell} - (\ell+1). \quad (8)$$

For this, we start by defining a pair of distributions $(\mathcal{YES}^*_\ell, \mathcal{NO}^*_\ell)$ over $\{0,1\}^{\ell \times N^*_\ell}$ (note that the number of rows for these distributions, ℓ, is exactly the same as that of $\mathcal{YES}_{\ell-1}, \mathcal{NO}_{\ell-1}, \mathcal{YES}'_{\ell-1}$, and $\mathcal{NO}'_{\ell-1}$), with N^*_ℓ defined as

$$N^*_\ell = n \cdot N_{\ell-1} = N_\ell - 1.$$

To obtain $(\mathcal{YES}^*_\ell, \mathcal{NO}^*_\ell)$, we partition the N^*_ℓ columns into n *sections*, each with $N_{\ell-1}$ columns (and ℓ rows). (So the first section consists of $x_{i,j}$ with $j \in [N_{\ell-1}]$, the second section consists of $x_{i,j}$ with $j \in [N_{\ell-1}+1, 2N_{\ell-1}]$, and so forth.) A draw of a string from $\mathcal{YES}^*_\ell$ is obtained as follows: first we draw an integer $\boldsymbol{T}$ uniformly at random from $[n]$, and then

- (a) For each $i < \boldsymbol{T}$, we independently set the i-th section to be a string drawn from $\mathcal{NO}_{\ell-1}$ with probability 1/2 or a string drawn from $\mathcal{YES}'_{\ell-1}$ with probability 1/2.
- (b) For each $i > \boldsymbol{T}$, we set the i-th section to be all 0.
- (c) Set the $\boldsymbol{T}$-th section to be a string drawn from $\mathcal{YES}_{\ell-1}$.

See Figure 1 for an illustration.

A draw of a string from $\mathcal{NO}^*_\ell$ is obtained in a similar fashion. First we draw $\boldsymbol{T}$ from $[n]$ uniformly at random, and then

- (a') For each $i < \boldsymbol{T}$, we independently set the i-th section to be a string drawn from $\mathcal{NO}_{\ell-1}$ with probability 1/2 or a string drawn from $\mathcal{YES}'_{\ell-1}$ with probability 1/2. (This is the same as step (a) in the definition of $\mathcal{YES}^*_\ell$.)
- (b') For each $i > \boldsymbol{T}$, we set the i-th section to be all 1.
- (c') Set the $\boldsymbol{T}$-th section to be a string drawn from $\mathcal{NO}'_{\ell-1}$.

Again see Figure 1 for an illustration. Given (7), we see that an $\boldsymbol{x}$ drawn from $\mathcal{YES}^*_\ell$ (or $\mathcal{NO}^*_\ell$) has sum equal to $2^{N^*_\ell}$ or $(2^{N^*_\ell} - \ell)$.

With the definitions of $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ in hand, we now use them to define $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ and $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ so that each $\boldsymbol{x}$ drawn from these distributions should have $\mathrm{Sum}_{\ell+1,N_\ell}(\boldsymbol{x})$ equal to the values given in (8). Recall that $N_\ell = N^*_\ell + 1$.

A string $\boldsymbol{x} = (x_{i,j}) \in \{0,1\}^{(\ell+1)\times N_\ell}$ drawn from $\mathcal{YES}'_\ell$ is obtained as follows. First we draw a string $\boldsymbol{z}$ from $\mathcal{YES}^*_\ell$ and put it in columns $\{2,\ldots,N_\ell\}$ and rows $\{1,\ldots,\ell\}$ of $\boldsymbol{x}$, i.e., $x_{i,j} = z_{i,j-1}$ for all $i \in [\ell]$ and $j \in \{2,\ldots,N_\ell\}$. For the remaining positions (in

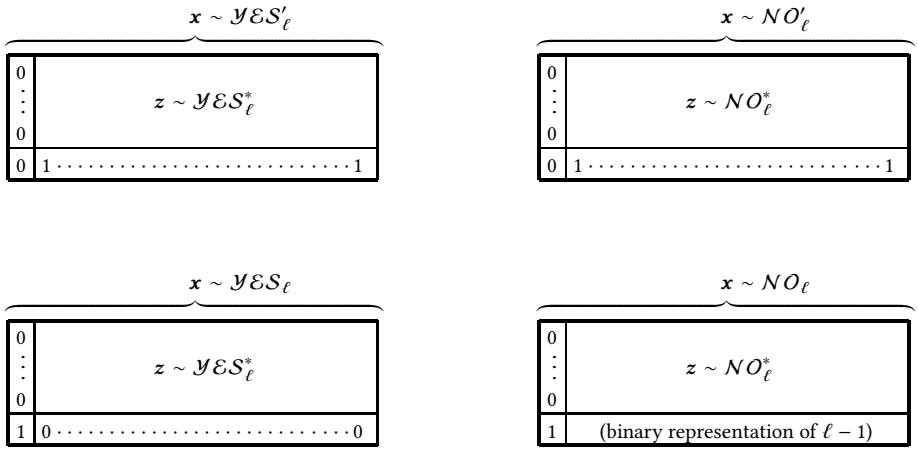

Figure 2: Illustrations of how the $\mathcal{YES}'_\ell, \mathcal{NO}'_\ell, \mathcal{YES}_\ell$ and $\mathcal{NO}_\ell$ distributions are defined from the $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ distributions.

the first column and the last row), we set $x_{i,1} = 0$ for all $i \in [\ell + 1]$ and $x_{\ell+1,j} = 1$ for all $j > 1$. The other distribution $\mathcal{NO}'_\ell$ is defined similarly, except that we draw the string z from $\mathcal{NO}^*_\ell$ instead of from $\mathcal{YES}^*_\ell$. This is illustrated in Figure 2.

For the other pair $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$, a string x drawn from $\mathcal{YES}_\ell$ is obtained as follows. As before, we first draw a string z from $\mathcal{YES}^*_\ell$ and put it in columns $\{2, \ldots, N_\ell\}$ and rows $\{1, \ldots, \ell\}$ of x. Then we set $x_{\ell+1,1} = 1$ and all other variables on the first row and last column of x to be 0. For the other distribution $\mathcal{NO}_\ell$, we similarly draw z from $\mathcal{NO}^*_\ell$ and put it in columns $\{2, \ldots, N_\ell\}$ and rows $\{1, \ldots, \ell\}$ of x. We set $x_{\ell+1,1} = 1$ and all other variables on the first column to be 0. We set the last row, i.e., $x_{\ell+1,j}$ with $j \in \{2, \ldots, N_\ell\}$, to be the binary representation of $\ell - 1$. (This is well defined since $N^*_\ell \geq n \geq 2^{60d} \gg \log d \geq \log \ell$.) See Figure 2.

We recall that $N_d \leq N$, and record the following simple fact:

Fact 1. *A string x from $\mathcal{YES}_\ell, \mathcal{NO}_\ell, \mathcal{YES}'_\ell, \mathcal{NO}'_\ell$ has*

$$\mathrm{Sum}_{\ell+1,N_\ell}(x) = 2^{N_\ell}, 2^{N_\ell} - 1, 2^{N_\ell} - 1 \text{ and } 2^{N_\ell} - (\ell + 1).$$

A very important property of the $(\mathcal{YES}_\ell, \mathcal{NO}_\ell)$ pair and the $(\mathcal{YES}'_\ell, \mathcal{NO}'_\ell)$ pair — which in fact motivated the definitions of these pairs of distributions in terms of $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ — is that they are *at least as hard* to distinguish as $(\mathcal{YES}^*_\ell, \mathcal{NO}^*_\ell)$ for monotone majority circuits. This is made more formal in the following two lemmas. The first lemma is trivial since $\mathcal{YES}'_\ell$ and $\mathcal{NO}'_\ell$ are obtained from $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ by adding bits of the same value.

Lemma 2.2. *Given any monotone majority circuit F over $\{0,1\}^{(\ell+1)\times N_\ell}$, there is a monotone majority circuit F^* over $\{0,1\}^{\ell \times N^*_\ell}$ of the same size and depth as F such that*

$$\mathbf{Pr}_{x \in \mathcal{YES}'_\ell}\left[F(x) = 1\right] + \mathbf{Pr}_{y \in \mathcal{NO}'_\ell}\left[F(y) = 0\right] = \mathbf{Pr}_{x \in \mathcal{YES}^*_\ell}\left[F^*(x) = 1\right] + \mathbf{Pr}_{y \in \mathcal{NO}^*_\ell}\left[F^*(y) = 0\right].$$

Proof. Given F, we hard-wire the variables in the first column to be 0 and the rest of the variables in the last row to be 1. Let F^* denote the new monotone majority circuit obtained from F of the same size and depth. The definition of $\mathcal{YES}'_\ell, \mathcal{NO}'_\ell$ from $\mathcal{YES}^*_\ell$, $\mathcal{NO}^*_\ell$ implies that

$$\mathbf{Pr}_{x \in \mathcal{YES}'_\ell}\left[F(x) = 1\right] = \mathbf{Pr}_{x \in \mathcal{YES}^*_\ell}\left[F^*(x) = 1\right] \text{ and}$$
$$\mathbf{Pr}_{y \in \mathcal{NO}'_\ell}\left[F(y) = 0\right] = \mathbf{Pr}_{y \in \mathcal{NO}^*_\ell}\left[F^*(y) = 0\right].$$

The lemma then follows. □

The second lemma relies on the monotonicity of circuits.

Lemma 2.3. *Given any monotone majority circuit F over $\{0,1\}^{(\ell+1)\times N_\ell}$, there is a monotone majority circuit F^* over $\{0,1\}^{\ell \times N^*_\ell}$ of the same size and depth as F such that*

$$\mathbf{Pr}_{x \in \mathcal{YES}_\ell}\left[F(x) = 1\right] + \mathbf{Pr}_{y \in \mathcal{NO}_\ell}\left[F(y) = 0\right] \leq \mathbf{Pr}_{x \in \mathcal{YES}^*_\ell}\left[F^*(x) = 1\right] + \mathbf{Pr}_{y \in \mathcal{NO}^*_\ell}\left[F^*(y) = 0\right].$$

Proof. Given F, we hard-wire $x_{\ell+1,1}$ to be 1 and the rest of the variables in the first column and the last row to be 0. Let F^* denote the resulting monotone majority circuit obtained from F of the same size and depth. The definition of $\mathcal{YES}_\ell$, $\mathcal{NO}_\ell$ from $\mathcal{YES}^*_\ell$, $\mathcal{NO}^*_\ell$ implies that

$$\mathbf{Pr}_{x \in \mathcal{YES}_\ell}\left[F(x) = 1\right] = \mathbf{Pr}_{x \in \mathcal{YES}^*_\ell}\left[F^*(x) = 1\right] \text{ and}$$
$$\mathbf{Pr}_{y \in \mathcal{NO}_\ell}\left[F(y) = 0\right] \leq \mathbf{Pr}_{y \in \mathcal{NO}^*_\ell}\left[F^*(y) = 0\right],$$

where the inequality follows from the monotonicity of F.

The lemma then follows. □

2.3 The Key Induction Lemma

Given distributions defined in Sections 2.1 and 2.2, we prove the following key technical lemma. Recall $\varepsilon = 2^{-12d}$. Let $M = 2^{\varepsilon^5 n}$.

Lemma 2.4. *Let $\ell \in \{2, \dots, d\}$. Suppose that any depth-$(\ell-1)$ monotone majority circuit F over $\{0,1\}^{\ell \times N_{\ell-1}}$ of size at most M has*

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}_{\ell-1}}\left[F(\boldsymbol{x})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}_{\ell-1}}\left[F(\boldsymbol{y})=0\right] \le 1 + 7^{\ell-2}\varepsilon,$$

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}'_{\ell-1}}\left[F(\boldsymbol{x})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[F(\boldsymbol{y})=0\right] \le 1 + 7^{\ell-2}\varepsilon. \quad (9)$$

Then any depth-ℓ monotone majority circuit F^* over $\{0,1\}^{\ell \times N^*_\ell}$ of size at most M satisfies

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}^*_{\ell}}\left[F^*(\boldsymbol{x})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}^*_{\ell}}\left[F^*(\boldsymbol{y})=0\right] \le 1 + 7^{\ell-1}\varepsilon.$$

Proof. Recall that strings drawn from $\mathcal{YES}^*_\ell$ and $\mathcal{NO}^*_\ell$ have n sections. We refer to strings in $\{0,1\}^{\ell \times N_{\ell-1}}$ as *section strings*.

We begin by defining some useful distributions $\mathcal{D}_1, \dots, \mathcal{D}_n$ over concatenations of section strings where $\mathcal{D}_t$ is supported on concatenations of $t-1$ section strings. First, let $\mathcal{D}$ denote the following distribution over section strings: $\boldsymbol{x} \sim \mathcal{D}$ is drawn from $\mathcal{NO}_{\ell-1}$ with probability $1/2$ and is drawn from $\mathcal{YES}'_{\ell-1}$ with probability $1/2$. For each $t \in [n]$, we use $\mathcal{D}_t$ to denote the distribution of the concatenation of $t-1$ section strings, each drawn from $\mathcal{D}$ independently. (So $\mathcal{D}_t$ is a distribution over $\{0,1\}^{\ell \times (t-1) N_{\ell-1}}$.) Note that in the special case when $t=1$, $\mathcal{D}_1$ is supported on the empty string only. Also note that for $t \in [n]$, $\mathcal{D}_t$ is generated according to (a) or (a') from Section 2.2 (recall that (a) and (a') are the same).

As in the statement of Lemma 2.4, let F^* be a depth-ℓ monotone majority circuit over $\{0,1\}^{\ell \times N^*_\ell}$ of size at most M. We say a string $z \in \mathrm{supp}(\mathcal{D}_t)$ for some $t \in [n]$ is *good* with respect to F^* if

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}_{\ell-1}}\left[F^*(z \circ \boldsymbol{x} \circ \mathbf{0})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[F^*(z \circ \boldsymbol{y} \circ \mathbf{1})=0\right]$$

is at least $1 + 6\delta$, where we write $\mathbf{0}$ and $\mathbf{1}$ to denote the all-0 and all-1 strings in $\{0,1\}^{\ell \times (n-t) N_\ell}$, and δ is set to be $7^{\ell-2}\varepsilon$.

Now we fix a $t \in [n]$ and fix a good string $z \in \mathrm{supp}(\mathcal{D}_t)$. Let ρ_z be the restriction that fixes the first $t-1$ sections of variables of F^* to be z and leaves the remaining $n-(t-1)$ sections unfixed. As z is good, we have that $F^* \restriction \rho_z$ is nontrivial (i.e., $F^* \restriction \rho_z \not\equiv 0$ or 1). We write $H_1, \dots, H_m$ (with multiplicities) to denote the set of all depth-$(\ell-1)$ sub-circuits rooted at children of the output gate of F^* such that $H_i \restriction \rho_z$ is nontrivial. In other words, we assume that the same sub-circuit may appear multiple times in this list if the output majority gate in F^* contains multiple wires to it. Since the size of (F^*) is at most M, the fan-in of the output majority gate of F^* is at most M, and consequently $m \le M$. Since $F^* \restriction \rho$ is nontrivial there is a positive integer $h \in [M]$ such that $F^* \restriction \rho_z$ outputs 1 if and only if at least h many of $H_1 \restriction \rho_z, \dots, H_m \restriction \rho_z$ output 1. The following claim shows that with non-negligible probability, a random $\boldsymbol{x} \sim \mathcal{D}$ is such that "many" H_i's become trivial (i.e., compute a constant function) after a restriction by $\rho_{z \circ \boldsymbol{x}}$.

The proof of Claim 2 crucially relies on the monotonicity of H_i. In particular, it used the property that if $H_i(z \circ x \circ \mathbf{1}) = 0$ for some x, then H_i must become trivial after the restriction $\rho_{z \circ x}$.

Claim 2. *Suppose that $z \in \mathrm{supp}(\mathcal{D}_t)$ is a good string. Then*

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{D}}\left[\left|\{i \in [m] : H_i \restriction \rho_{z \circ \boldsymbol{x}} \text{ is trivial}\}\right| \ge \delta^2 m/2\right] \ge \delta/4.$$

Proof. We consider two cases: $h \ge m/2$ or $h < m/2$. We focus on the latter below and the former case is symmetric. Assume that $h < m/2$. Since z is good, we have

$$\mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[F^*(z \circ \boldsymbol{y} \circ \mathbf{1})=0\right] \ge 1 + 6\delta - 1 = 6\delta.$$

If $y \in \mathrm{supp}(\mathcal{NO}'_{\ell-1})$ satisfies $F^*(z \circ y \circ \mathbf{1}) = 0$, then by $h < m/2$ it must be the case that at least $m/2$ of H_i's have $H_i(z \circ y \circ \mathbf{1}) = 0$, so

$$\mathbf{E}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[\text{number of } H_i\text{'s with } H_i(z \circ \boldsymbol{y} \circ \mathbf{1}) = 0\right] \ge 3\delta m. \quad (10)$$

Let I denote the set of $i \in [m]$ such that

$$\mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[H_i(z \circ \boldsymbol{y} \circ \mathbf{1}) = 0\right] \ge 2\delta. \quad (11)$$

Then we have from (10) that

$$|I| \cdot 1 + (m - |I|) \cdot 2\delta \ge 3\delta m,$$

which implies that $|I| \ge \delta m$.

We write ρ to denote the restriction over $\{0,1\}^{\ell \times N^*_\ell}$ that fixes the first $t-1$ sections of input variables to be z and the last $(n-t)$ sections of input variables to be all 1, and leaves only the variables in the t-th section unfixed. So each $H_i \restriction \rho$ is a depth-$(\ell-1)$ monotone majority circuit over $\{0,1\}^{\ell \times N_{\ell-1}}$ of size at most M. Then combining (11) and the assumption (9) of the lemma, applied to $H_i \restriction \rho$, we have that each $i \in I$ satisfies

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}'_{\ell-1}}\left[H_i(z \circ \boldsymbol{x} \circ \mathbf{1}) = 1\right] \le 1 + \delta - 2\delta = 1 - \delta,$$

and thus,

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}'_{\ell-1}}\left[H_i(z \circ \boldsymbol{x} \circ \mathbf{1}) = 0\right] \ge \delta. \quad (12)$$

Observe that, if an $x \in \mathrm{supp}(\mathcal{YES}'_{\ell-1})$ satisfies $H_i(z \circ x \circ \mathbf{1}) = 0$, then we have $H_i \restriction \rho_{z \circ x} \equiv 0$ by the *monotonicity* of H_i. Let $\mathbf{X}$ be a random variable that denotes the number of H_i's that become trivial after $\rho_{z \circ \boldsymbol{x}}$, where $\boldsymbol{x} \sim \mathcal{YES}'_{\ell-1}$. So by (12) the expectation of $\mathbf{X}$ is at least $\delta|I|$. Let q denote the probability of $\mathbf{X} \ge \delta|I|/2$. The lower bound $\mathbf{E}[\mathbf{X}] \ge \delta|I|$ implies that

$$q \cdot |I| + (1-q) \cdot \delta|I|/2 \ge \delta|I|,$$

and thus $q \ge \delta/2$. Plugging in $|I| \ge \delta m$, we have that $\mathbf{X} \ge \delta^2 m/2$ with probability at least $\delta/2$.

Finally, taking into account that a draw of $\boldsymbol{x} \sim \mathcal{D}$ is drawn from $\mathcal{YES}'_{\ell-1}$ with probability $1/2$, we see that with probability at least $\delta/4$ over a draw of $\boldsymbol{x} \sim \mathcal{D}$, we have that at least $\delta^2 m/2$ many H_i's become trivial after $\rho_{z \circ \boldsymbol{x}}$. This finishes the proof of the claim. □

Claim 2 implies that when $\boldsymbol{z} \sim \mathcal{D}_t$ is good (with respect to F^*), then with probability at least $\delta/4$ over a random draw of $\boldsymbol{x} \sim \mathcal{D}$, the restriction $\rho_{\boldsymbol{z} \circ \boldsymbol{x}}$ trivializes at least $(\delta^2/2)$-fraction of the depth-$(\ell-1)$ sub-circuits of F that are *not* trivialized by $\rho_{\boldsymbol{z}}$. Intuitively, this is useful because it means that we have a good chance of getting a significant simplification of F (shrinking the fan-in of the top gate by a lot), and since F is of size at most M this cannot happen too many times. (This is the place where we use the assumption on the size of F.) On the other hand, if $\boldsymbol{z}$ is not good, then

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}_{\ell-1}}\left[F^*(\boldsymbol{z} \circ \boldsymbol{x} \circ \mathbf{0})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}'_{\ell-1}}\left[F^*(\boldsymbol{z} \circ \boldsymbol{y} \circ \mathbf{1})=0\right]$$

is at most $1 + 6\delta$, which is also useful for our purpose of bounding

$$\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{YES}^*_{\ell}}\left[F^*(\boldsymbol{x})=1\right] + \mathbf{Pr}_{\boldsymbol{y} \sim \mathcal{NO}^*_{\ell}}\left[F^*(\boldsymbol{y})=0\right] \quad (13)$$

from above by $1 + 7\delta$.

To finish the proof of the lemma, we take the following alternative but equivalent view of (13). Let $\boldsymbol{z}_1, \dots, \boldsymbol{z}_n$ be a sequence of

random section strings, each drawn from $\mathcal{D}$ independently. By the definition of $\mathcal{YES}^*_\ell, \mathcal{NO}^*_\ell$ (recall Figure 1), (13) $\times n$ is equal to

$$\mathbf{E}_{z_1,\ldots,z_n}\left[\sum_{t=1}^{n}\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{YES}_{\ell-1}}\left[F^*(z_1\circ\cdots\circ z_{t-1}\circ\boldsymbol{x}\circ\mathbf{0})=1\right]+\sum_{t=1}^{n}\mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{NO}'_{\ell-1}}\left[F^*(z_1\circ\cdots\circ z_{t-1}\circ\boldsymbol{y}\circ\mathbf{1})=0\right]\right].$$

This can be viewed as the expectation of a random variable $\mathbf{\Gamma}$ generated as follows: (1) start with $\mathbf{\Gamma}=0$; (2) for each "round" $t=1,\ldots,n$, independently draw z_t from $\mathcal{D}$ and add the following to $\mathbf{\Gamma}$:

$$\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{YES}_{\ell-1}}\left[F^*(z_1\circ\cdots\circ z_{t-1}\circ\boldsymbol{x}\circ\mathbf{0})=1\right]+\mathbf{Pr}_{\boldsymbol{y}\sim\mathcal{NO}'_{\ell-1}}\left[F^*(z_1\circ\cdots\circ z_{t-1}\circ\boldsymbol{y}\circ\mathbf{1})=0\right].$$

So it suffices to show that $\mathbf{E}[\mathbf{\Gamma}]\le(1+7\delta)n$.

For each of the n rounds $t=1,\ldots,n$, exactly one of the following two possibilities must hold:

(1) The current string $z_1\circ\cdots\circ z_{t-1}$ is not good. In this case $\mathbf{\Gamma}$ goes up by at most $1+6\delta$ in the t-th round. Otherwise,

(2) The current string $z_1\circ\cdots\circ z_{t-1}$ is good. In this case $\mathbf{\Gamma}$ can go up by at most 2 in the t-th round, but by our previous analysis (i.e., Claim 2), the number of nontrivial depth-$(\ell-1)$ subcircuits of F^* (with multiplicities) rooted at children of the output gate of F^* drops by a factor of $(1-\delta^2/2)$ with probability at least $\delta/4$ when the draw of z_t in the t-th round extends the restriction to $\rho_{z_1\circ\cdots\circ z_t}$. Note that F^* has size at most

$$M\le 2^{\varepsilon^5 n}$$

so it can survive at most $2\delta^3 n$ many such drops before F^* becomes trivial; to see this, observe that

$$(1-\delta^2/2)^{2\delta^3 n}\le\exp\left(-(\delta^2/2)\cdot(2\delta^3 n)\right)=\exp\left(-\delta^5 n\right)<2^{-\varepsilon^5 n}. \tag{14}$$

Note further that once F^* becomes trivial, $\mathbf{\Gamma}$ goes up by 1 in every subsequent round.

We let $\mathbf{S}$, a random variable, denote the total number of rounds $t\in[n]$ such that the current string $z_1\circ\cdots\circ z_{t-1}$ is good. (Note that once F^* becomes trivial the current string cannot be good.) We claim that $\mathbf{S}\le 32\delta^2 n$ with high probability.

Claim 3. *$\mathbf{S}\le 32\delta^2 n$ with probability at least $1-\exp(-n\delta^4/2)$.*

Proof. We say that round t is good if the current string $z_1\circ\cdots\circ z_{t-1}$ is good. We say that F^* is *hit* in the t-th round, if $z_1\circ\cdots\circ z_{t-1}$ is good and the number of depth-$(\ell-1)$ subcircuitscuits of F^* (with multiplicities) that are trivial under the restriction $\rho_{z_1\circ\cdots\circ z_{t-1}}$ drops by a factor of at least $(1-\delta^2/2)$ under the restriction $\rho_{z_1\circ\cdots\circ z_{t-1}\circ z_t}$. Then we can write $\mathbf{Pr}[\mathbf{S}\ge 32\delta^2 n]$ as the sum of the following two probabilities (and bound them separately):

$$\mathbf{Pr}\left[\mathbf{S}\ge 32\delta^2 n\ \&\ F^*\text{ is hit}>2\delta^3 n\text{ many times during the first }32\delta^2 n\text{ of the good rounds}\right]$$

and

$$\mathbf{Pr}\left[\mathbf{S}\ge 32\delta^2 n\ \&\ F^*\text{ is hit}\le 2\delta^3 n\text{ many times during the first }32\delta^2 n\text{ of the good rounds}\right].$$

The first of these probabilities is zero because of (14), i.e. if F^* is hit $2\delta^3 n$ times then it is trivialized so no subsequent rounds can be good and thus F^* cannot be hit again.

We focus on upper bounding the second probability. For each i from 1 to $32\delta^2 n$ we define the following random variable $\mathbf{Y}_i$ where

$$\mathbf{Y}_i=\begin{cases}1 & \text{if } F^*\text{ is hit in the } i\text{-th good round}\\ & \text{or there are fewer than } i \text{ good rounds}\\ 0 & \text{otherwise (there are at least } i \text{ good rounds}\\ & \text{and } F^*\text{ is not hit in the } i\text{th good round).}\end{cases}$$

The second probability we are interested in is at most $\mathbf{Pr}[\sum_i \mathbf{Y}_i\le 2\delta^3 n]$. By Claim 2, we have

$$\mathbf{E}\left[\mathbf{Y}_i\mid\mathbf{Y}_1=b_1,\ldots,\mathbf{Y}_{i-1}=b_{i-1}\right]\ge\delta/4 \tag{15}$$

for all i and all $b_1,\ldots,b_{i-1}\in\{0,1\}$. Let $\mathbf{X}_0\equiv 0$ and

$$\mathbf{X}_i=\mathbf{X}_{i-1}+\mathbf{Y}_i-\mathbf{E}\left[\mathbf{Y}_i\mid\mathbf{Y}_1,\cdots,\mathbf{Y}_{i-1}\right].$$

Then $\mathbf{X}_0,\mathbf{X}_1,\ldots$ is a martingale that satisfies $|\mathbf{X}_i-\mathbf{X}_{i-1}|\le 1$ with probability 1, and we have that

$$\mathbf{X}_{32\delta^2 n}=\sum_{i=1}^{32\delta^2 n}\left(\mathbf{Y}_i-\mathbf{E}\left[\mathbf{Y}_i\mid\mathbf{Y}_1,\cdots,\mathbf{Y}_{i-1}\right]\right)\le\sum_{i=1}^{32\delta^2 n}\mathbf{Y}_i-8\delta^3 n,$$

using (15) for the inequality. Applying the Azuma-Hoeffding inequality (see, e.g., Theorem 5.1 of [13]) to the martingale sequence $\mathbf{X}_0,\mathbf{X}_1,\ldots$, we get that

$$\mathbf{Pr}\left[\sum_i\mathbf{Y}_i\le 2\delta^3 n\right]\le\mathbf{Pr}\left[\mathbf{X}_{32\delta^2 n}\le 2\delta^3 n-8\delta^3 n\right]\le\exp\left(-\frac{(6\delta^3 n)^2}{2\cdot 32\delta^2 n}\right)<\exp(-n\delta^4/2).$$

This finishes the proof of the claim. □

We are almost done with the proof of Lemma 2.4. By $\delta=7^{\ell-2}\varepsilon$,

$$\exp\left(-n\delta^4/2\right)\le\delta/4\quad\text{and}\quad\delta\le 2^{-8} \tag{16}$$

since $d\ge 2$, $n\ge 2^{60d}$, $\varepsilon=2^{-12d}$ and $\ell\in\{2,\ldots,d\}$. By Claim 3,

$$\mathbf{E}\left[\mathbf{\Gamma}\right]\le\exp(-n\delta^4/2)\cdot 2n+(1-\exp(-n\delta^4/2))\cdot\left(2\cdot 32\delta^2 n+(1+6\delta)\cdot(n-32\delta^2 n)\right)$$
$$<\delta n/2+64\delta^2 n+(1+6\delta)n\le(1+7\delta)n,$$

where we also used the two inequalities in (16).

This finishes the proof of Lemma 2.4. □

2.4 Proof of Theorem 1.1

Finally we combine all the ingredients to prove Theorem 1.1.

Recall that d, n, and N are positive integers that satisfy

$$d\ge 2,\quad n\ge 2^{60d},\quad\text{and}\quad N\ge(2^{13}n)^d\ge N_d.$$

We also have $\varepsilon=2^{-12d}$ and $M=2^{\varepsilon^5 n}$. We first prove by induction on ℓ that, for each $\ell=1,\ldots,d$, any monotone majority circuit F over $\{0,1\}^{(\ell+1)\times N_\ell}$ of depth ℓ and size at most M satisfies

$$\mathbf{Pr}_{\boldsymbol{x}\sim \mathcal{YES}_\ell}\left[F(\boldsymbol{x})=1\right]+\mathbf{Pr}_{\boldsymbol{y}\sim \mathcal{NO}_\ell}\left[F(\boldsymbol{y})=0\right]\le 1+7^{\ell-1}\varepsilon,$$

$$\mathbf{Pr}_{\boldsymbol{x}\sim \mathcal{YES}'_\ell}\left[F(\boldsymbol{x})=1\right]+\mathbf{Pr}_{\boldsymbol{y}\sim \mathcal{NO}'_\ell}\left[F(\boldsymbol{y})=0\right]\le 1+7^{\ell-1}\varepsilon. \quad (17)$$

The $\ell = 1$ base case follows from Lemma 2.1. Now assume that (??) holds for $\ell - 1$. By Lemma 2.4, any monotone majority circuit F^* over $\{0,1\}^{\ell\times N^*_\ell}$ of depth ℓ and size at most M satisfies

$$\mathbf{Pr}_{\boldsymbol{x}\sim \mathcal{YES}^*_\ell}\left[F^*(\boldsymbol{x})=1\right]+\mathbf{Pr}_{\boldsymbol{y}\sim \mathcal{NO}^*_\ell}\left[F^*(\boldsymbol{y})=0\right]\le 1+7^{\ell-1}\varepsilon. \quad (18)$$

It follows from Lemmas 2.2 and 2.3 that every monotone majority circuit F over $\{0,1\}^{(\ell+1)\times N_\ell}$ of depth ℓ and size at most M satisfies (??). This finishes the induction.

We finish the proof using $(\mathcal{YES}^*_d, \mathcal{NO}^*_d)$ over $\{0,1\}^{d\times N^*_d}$, with $N^*_d = N_d - 1 < N$. Given (18) and the fact of $7^{d-1}\varepsilon < 1$, no depth-d monotone majority circuit over $\{0,1\}^{d\times N^*_d}$ of size at most M can compute Add_{d,N^*_d} correctly on all inputs. This is because every $\boldsymbol{x}$ from $\mathcal{YES}^*_d$ has sum $2^{N^*_d}$ and hence $\mathrm{Add}_{d,N^*_d}(\boldsymbol{x}) = 1$, while every $\boldsymbol{y}$ from $\mathcal{NO}^*_d$ has sum $2^{N^*_d} - d$ and hence we have $\mathrm{Add}_{d,N^*_d}(\boldsymbol{y}) = 0$. Since $N > N^*_d$, this establishes Theorem 1.1.

3 THE UPPER BOUND

We prove Theorem 1.2 in this section. We focus on the case when $N^{1/d} > 1$ is a positive integer, and give a depth-d monotone majority circuit that computes $\mathrm{Add}_{k,N}$ and has size at most

$$2^{3(N^{1/d}\cdot\log k+\log N)}. \quad (19)$$

For the general case, we let $n = \lceil N^{1/d}\rceil > 1$, and let s denote the smallest integer such that $n^s \ge N$ (so $s \le d$). Then we first construct a depth-s monotone majority circuit that computes Add_{k,n^s}, and then hard-wire the variables in the last $n^s - N$ columns to be 0 to get a circuit for $\mathrm{Add}_{k,N}$. The size bound given in the statement of Theorem 1.2 follows from (19) and the simple facts that $n \le 2N^{1/d}$ and $n^s \le nN \le N^2$. For the rest of the section, we assume that $n = N^{1/d} > 1$ is an integer.

First we note that the theorem (with the size bound as given in (19); the same below) is trivial if $N < \log k$ since implementing $\mathrm{Add}_{k,N}$ directly using a single MAJ gate takes a total weight of

$$k\cdot 2^N < 2^{3\log k}.$$

Assuming that $N \ge \log k$ below, we let $t \in \{1,\dots,d\}$ denote the smallest integer such that $n^t = N^{t/d} \ge \log k$. We also write $M = n^t$. It is clear by the choice of t that we have

$$M \le n\log k. \quad (20)$$

With the same reasoning the theorem is trivial if $M = N$. Below we assume that $t \le d-1$.

We need some notation for our construction. We say that $\mathcal{S} = (S_1,\dots,S_\ell)$ is an *ℓ-decomposition* of $[N]$ if there exist indices

$$1 = a^-_1 \le a^+_1 < a^-_2 \le a^+_2 < \ldots < a^-_\ell \le a^+_\ell = N$$

such that $\bigcup_{\gamma\in[\ell]} S_\gamma = [N]$ and $S_\gamma = \{a^-_\gamma, a^-_\gamma+1,\dots,a^+_\gamma\}$. In other words, $\mathcal{S}$ partitions $[N]$ into ℓ sequential intervals.

Let $(x_{i,j})_{i\in[k],j\in[N]}$ denote the set of input variables of $\mathrm{Add}_{k,N}$. Given an ℓ-decomposition $\mathcal{S}$, we define a sequence of "conditional" *carry-bit functions* $c^{(\gamma)}_{\alpha,\beta}(x)$ with $\alpha,\beta : 0 \le \alpha,\beta \le k-1$ and $\gamma\in[\ell]$. Each function depends only on $x_{i,j}$'s with $j \in S_\gamma$. For convenience, let $B_\gamma = [k]\times S_\gamma$ be the set of indices of these variables. Intuitively, for an assignment $x\in\{0,1\}^{k\times N}$, we have $c^{(\gamma)}_{\alpha,\beta}(x) = 1$ if and only if a carry of value at least α is generated/propagated by the input bits corresponding to B_γ, assuming this block of variables receives a carry of value β from the block to the right. Formally,

$$c^{(\gamma)}_{\alpha,\beta}(x) \overset{\text{def}}{=} 1 \iff \sum_{(i,j)\in B_\gamma} 2^{|S_\gamma|-(j+1-a^-_\gamma)}\cdot x_{i,j} + \beta \ge \alpha\cdot 2^{|S_\gamma|}. \quad (21)$$

For each $i\in\{0,\dots,d-t\}$, we use the notation $\mathcal{S}^{(i)}$ to denote the n^i-decomposition in which each set has size n^{d-i}. Note that, for the 1-decomposition $\mathcal{S}^{(0)} = (S^{(0)}_1)$, where $S^{(0)}_1 = \{1,\dots,N\}$, we have

$$\mathrm{Add}_{k,N}(x) = 1 \iff c^{(1)}_{1,0}(x) = 1, \quad (22)$$

for the function $c^{(1)}_{1,0}$ of $\mathcal{S}^{(0)}$.

Our construction is based on a recursive computation of functions $c^{(\gamma)}_{\alpha,\beta}(\cdot)$ associated to different decompositions $\mathcal{S}^{(r)}$, for r from $d-t$ back to 0, where each decomposition $\mathcal{S}^{(r+1)}$ is obtained via a refinement of the previous decomposition $\mathcal{S}^{(r)}$. More precisely we construct our monotone majority circuit for $\mathrm{Add}_{k,N}$ with the following intended behavior. The top gate of the circuit computes the bit $c^{(1)}_{1,0}(x)$ associated to the decomposition $\mathcal{S}^{(0)}$. However, this gate does not have access to x: it receives as input the output of carry-bit functions $c^{(\gamma)}_{\alpha,\beta}(x)$ corresponding to the finer n-decomposition $\mathcal{S}^{(1)}$ in which each block has n^{d-1} columns. This then leads to a recursive procedure, which unfolds as a depth-$(d-t+1)$ circuit described in more detail below (recall that $t \ge 1$).

In general our circuit has $d-t+1$ layers of majority gates, where gates at the ith layer compute carry-bit functions $c^{(\ell)}_{\alpha,\beta}$ corresponding to the $n^{d-t-i+1}$-decomposition $\mathcal{S}^{(d-t-i+1)}$. The base case, i.e. the first layer of majority gates that are supposed to compute $c^{(\ell)}_{\alpha,\beta}$ of $\mathcal{S}^{d-t}$, is done by a majority gate that follows directly the definition given in (21). It is clear that the size of each gate in the first layer is bounded from above by $k2^M$.

Due to the recursive nature of our construction, it is sufficient to describe how to compute the carry-bit functions corresponding to a decomposition $\mathcal{S}^{(r)}$ from the carry-bit functions corresponding to $\mathcal{S}^{(r+1)}$ for each $r\in\{0,1,\dots,d-t-1\}$. For convenience we fix an r below and write $\mathcal{S}'$ for $\mathcal{S}^{(r)}$ and $\mathcal{S}$ for $\mathcal{S}^{(r+1)}$. We also fix a set $S'\in\mathcal{S}'$ with $S' = S_1\cup\ldots\cup S_n$, where $S_1,\dots,S_n$ are sets in the ordered tuple $\mathcal{S}$ listed from left to right. We write $c_{u,v}$ to denote a carry-bit function of the block S that we need to compute, for some $u,v\in\{0,\dots,k-1\}$, and assume that we have already computed $c^{(\gamma)}_{\alpha,\beta}$ for each block S_γ, $\gamma\in[n]$, and for all $\alpha,\beta\in\{0,\dots,k-1\}$. The goal is to compute $c_{u,v}(x)$ given the bits $c^{(\gamma)}_{\alpha,\beta}(x)$.

We start with a general observation about carry-bit functions of a block. We say that $(\alpha,\beta) \prec (\alpha',\beta')$ if either $\alpha < \alpha'$, or $\alpha = \alpha'$ and $\beta \ge \beta'$. Given a block S_γ, note that $c^{(\gamma)}_{\alpha,\beta}$ has the following monotonicity property. (Note that the assumption of $|S_\gamma| \ge \log k$ always holds given our choice of t and trivial cases ruled out at the beginning of the section.)

Claim 4. *Assume that $|S_\gamma| \ge \log k$. If $(\alpha,\beta)\prec(\alpha',\beta')$, then*

$$c^{(\gamma)}_{\alpha,\beta}(x) \ge c^{(\gamma)}_{\alpha',\beta'}(x)$$

on every input string x for $\mathrm{Add}_{k,N}$.

Proof. We consider the two cases of $(\alpha, \beta) \prec (\alpha', \beta')$. If $\alpha = \alpha'$ and $\beta \geq \beta'$, the claim follows immediately from (21).

Assume that $\alpha < \alpha'$, where $\beta, \beta' \in \{0, \ldots, k-1\}$ are arbitrary. Clearly it suffices to argue that

$$c^{(\gamma)}_{\alpha', k-1}(x) = 1 \implies c^{(\gamma)}_{\alpha'-1, 0}(x) = 1.$$

Using (21), this assumption is equivalent to

$$\sum_{(i,j) \in B_\gamma} 2^{|S_\gamma| - (j+1-a^-_\gamma)} \cdot x_{i,j} + (k-1) \geq \alpha' \cdot 2^{|S_\gamma|}. \tag{23}$$

In order to show $c^{(\gamma)}_{\alpha'-1, 0}(x) = 1$, we need to verify that

$$\sum_{(i,j) \in B_\gamma} 2^{|S_\gamma| - (j+1-a^-_\gamma)} \cdot x_{i,j} \geq (\alpha' - 1) \cdot 2^{|S_\gamma|}.$$

Using (23) it is sufficient to have $k - 1 \leq 2^{|S_\gamma|}$. This follows from the assumption of the claim, which completes the proof. □

The description of the majority gate that computes $c_{u,v}(x)$ for the block S' in $\mathcal{S}'$ using $c^{(\gamma)}_{\alpha,\beta}(x)$ for blocks $S_1, \ldots, S_n$ in $\mathcal{S}$ is based on the following lemma.

Lemma 3.1. *Assume that $|S_\gamma| \geq \log k$ for every $\gamma \in [n]$. Then we have $c_{u,v}(x) = 1$ if and only if*

$$v + \sum_{\gamma=1}^{n} \left(\left(\sum_{\alpha=1, \beta=0}^{k-1} c^{(\gamma)}_{\alpha,\beta}(x) \right) \cdot k^{n-\gamma} \right) \geq u \cdot k^n. \tag{24}$$

Proof. We consider (24) as a sum in base k over $k(k-1)$ rows and n columns of variables, with v extra 1's on column n (which corresponds to the least significant position). Let p_γ denote the (base k) carry from column γ to column $\gamma - 1$ in (24), and let q_γ denote the (base 2) carry from block γ to block $\gamma - 1$ in our decomposition of x after adding v to block n (without taking into account the remaining columns of x not covered by $S_1 \cup \ldots \cup S_n$).

We prove by induction that $p_\gamma = q_\gamma$, for all γ from n to 1. Notice that this establishes the lemma. For the base case when $\gamma = n$, we consider the following two cases:

(1) If $c^{(n)}_{\alpha,\beta} = 0$ for all $\alpha \geq 1$ and $\beta \geq 0$, then $q_n = 0$ (since we have $c^{(n)}_{1,k-1} = 0$ and $v \leq k-1$). This implies that $p_n = q_n = 0$.
(2) Otherwise, let (α_n, β_n) denote the largest pair (under $\prec$) with $c^{(n)}_{\alpha_n,\beta_n} = 1$. It follows from Claim 4 that $q_n = \alpha_n$ if $\beta_n \leq v$, and $q_n = \alpha_n - 1$ if $\beta_n > v$. We also have

$$v + \sum_{\alpha=1, \beta=0}^{k-1} c^{(n)}_{\alpha,\beta} = (\alpha_n - 1) \cdot k + (k - \beta_n + v).$$

It follows from this equation and the characterization of q_n that the (base k) carry $p_n = q_n$.

The induction step is similar. We assume that $p_{\gamma+1} = q_{\gamma+1}$, and prove that $p_\gamma = q_\gamma$. We focus on the γ-th column from (24) and block γ, and consider the following two cases:

(1) If $c^{(n)}_{\alpha,\beta} = 0$ for all $\alpha \geq 1$ and $\beta \geq 0$, then $q_\gamma = 0$ (since we have $c^{(\gamma)}_{1,k-1} = 0$ and $q_{\gamma+1} \leq k-1$). This implies that $p_\gamma = q_\gamma = 0$.
(2) Otherwise, let $(\alpha_\gamma, \beta_\gamma)$ denote the largest pair with $c^{(\gamma)}_{\alpha_\gamma,\beta_\gamma} = 1$. Using Claim 4, q_γ is α_γ if $\beta_\gamma \leq q_{\gamma+1}$, and q_γ is $\alpha_\gamma - 1$ if $\beta_\gamma > q_{\gamma+1}$. By the inductive hypothesis,

$$p_{\gamma+1} + \sum_{\alpha=1, \beta=0}^{k-1} c^{(n)}_{\alpha,\beta} = (\alpha_\gamma - 1) \cdot k + (k - \beta_\gamma + q_{\gamma+1}).$$

It follows from this equation and the characterization of q_γ that $p_\gamma = q_\gamma$.

This finishes the induction, and the proof of the lemma. □

Lemma 3.1, (21), and our previous discussions complete the description of the circuit for $\mathrm{Add}_{k,N}$. Moreover, its correctness follows easily from (22) and Lemma 3.1. It remains to analyze the size of the resulting depth-$(d-t+1)$ majority circuit.

We upper bound its size layer by layer as follows. As discussed earlier, the size of each majority gate in the first layer is at most $k2^M$, and there are n^{d-t} many of them. Furthermore, for the i-th layer of the circuit, where $i > 1$, there are $n^{(d-t-i+1)}$ gates each of which has size at most

$$k(k-1) \cdot \frac{k^n - 1}{k-1} < k^{n+1},$$

as given in Lemma 3.1. Using (20), the majority circuit for $\mathrm{Add}_{k,N}$ has overall size at most

$$n^{d-t} \cdot k2^M + \sum_{i=2}^{d-t+1} n^{d-t+1-i} \cdot k^{n+1} \leq Nk2^M + 2Nk^{n+1} \leq 2^{3(N^{1/d} \log k + \log N)}.$$

The construction presented here uses MAJ gates and majority circuits. We sketch in Appendix A an alternative construction with respect to semi-unbounded fan-in AND/OR circuits.

4 COMPLETING THE PROOF OF THEOREM 1.3: PROOF OF LEMMA 1.4

Recall the well-known technique of carry-save addition, also known as the "3-to-2 trick," for addition of binary numbers (see e.g., Section 1.2.3 of [22]). This "trick" states that there is a (multi-output) circuit that takes as input three n-bit binary numbers X, Y, Z and outputs two $(n+1)$-bit binary numbers A, B such that (*i*) $A + B = X + Y + Z$, and (*ii*) each output bit A_i or B_i depends on at most 3 of the input bits. By applying this trick in parallel to the N-bit integers $x^{(1)}, \ldots, x^{(k)}$ that are the rows of the input to $\mathrm{Add}_{k,N}$, we obtain $\lceil 2k/3 \rceil$ many $(N+1)$-bit integers whose sum equals

$$x^{(1)} + \cdots + x^{(k)}.$$

Recursing $O(\log k)$ times, we see that there are two $(N + O(\log k))$-bit integers (call them y and z) such that

$$y + z = x^{(1)} + \cdots + x^{(k)}.$$

A naive composition of "3-to-2 trick" circuits in a tree of depth $O(\log k)$ to compute y, z would yield a circuit of depth $\Theta(\log \log N)$. To avoid this blowup in circuit depth, we proceed differently, by observing that each each bit y_i, z_i depends on at most

$$3^{O(\log k)} \leq \log N$$

of the original input bits of the $x^{(i)}$'s, and exploiting this locality to get a depth-3 circuit overall.

In more detail, let y_i denote the bit in the "2^i-position" of the binary representation of y, so

$$y = \sum_{i=0}^{N+O(\log k)} y_i \cdot 2^i \quad \text{and similarly} \quad z = \sum_{i=0}^{N+O(\log k)} z_i \cdot 2^i.$$

We define "generate" and "propagate" bits for each bit position of $y+z$ in the standard way, $g_i \stackrel{\text{def}}{=} y_i \wedge z_i$ and $p_i \stackrel{\text{def}}{=} y_i \vee z_i$, so $g_i = 1$ if the bits in the 2^i-position generate a carry into the 2^{i+1}-position; $p_i = 1$ iff the bits in the 2^i-position propagate an incoming carry into the 2^i-position onward to the 2^{i+1}-position. Observe that each p_i, g_i depends on at most $2 \log N$ of the original input bits.

The sum $y + z$ is at least 2^N iff one of the following events hold:

- Event A: at least one of the bits $y_N, y_{N+1}, \ldots$, or $z_N, z_{N+1}, \ldots$ is 1. This can be expressed as
$$A = \bigvee_{j \geq N} (y_j \vee z_j).$$
Since y_j, z_j each depend on at most $\log N$ of the original input variables, each of them can be expressed as a poly(N)-size DNF over the original input variables, and thus A can be expressed as a poly(N)-size DNF.
- Event B: a carry bit is propagated into the 2^N-position. Event B can be expressed as
$$B = \bigvee_{j=1}^{N-1} \left(g_j \wedge \left(\bigwedge_{j<i<N} p_i \right) \right).$$
As each p_i depends on at most $2 \log N$ of the original input variables, it can be expressed as a poly(N)-size CNF; the same holds for g_j, so
$$\left(g_j \wedge \left(\bigwedge_{j<i<N} p_i \right) \right)$$
can be expressed as a poly(N)-size CNF, and thus Event B can be expressed as a poly(N)-size, depth-3 OR-AND-OR circuit.

As a result, $A \vee B$ can be expressed as a poly(N)-size, depth-3 OR-AND-OR circuit over the input variables. The lemma is proved.

APPENDIX

A UPPER BOUND FOR THE UNIVERSAL MONOTONE THRESHOLD GATE

We sketch in this section a construction of monotone circuits for the universal monotone threshold function that matches the parameters obtained by Beimel and Weinreb [8]. More precisely, we describe a polynomial size $O(\log N)$-depth AND/OR circuit for the function $\text{Add}_{O(N),O(N \log N)}$, where OR gates have unbounded fan-in, while AND gates have fan-in two.

Our construction relies on a more general reduction from $\text{Add}_{k,N}$ to a certain graph connectivity problem. To begin, we start with an ℓ-decomposition $\mathcal{S}$ of $\text{Add}_{k,N}$ (see Section 3 for more details), and assume (for now) that we are given the corresponding (conditional) carry-bit functions $c^{(\gamma)}_{\alpha,\beta}(x)$, where $\alpha, \beta \in \{0, \ldots, k-1\}$ and $\gamma \in [\ell]$.

Given these bits, we can view them as a layered directed graph $G_{\mathcal{S},x} = (V, E)$ which depends on x and $\mathcal{S}$ as follows. The vertices of G are partitioned into $\ell + 1$ layers, which we number for convenience from ℓ to 0. The first and last layers are special, and contain a single vertex only. The remaining layers each contain k vertices. The (directed) edges of this graph leave the γ-th layer and reach the $(\gamma-1)$-th layer. We use the output bit of each $c^{(\gamma)}_{\alpha,\beta}$ to decide whether an edge is present in this graph. The idea is that there will be a path from the ℓ-th layer to the 0-th layer if and only if $\text{Add}_{k,N}(x) = 1$.

More precisely, we view

$$V = L_\ell \cup L_{\ell-1} \cup \ldots \cup L_0,$$

where $L_\ell = \{s\}$, $L_0 = \{t\}$, and $L_\gamma = \{v_{\gamma,0}, \ldots, v_{\gamma,k-1}\}$, for $\ell > \gamma > 0$. The edge set $E \subseteq V \times V$ is defined as follows.

- $(s, v_{\ell-1,j}) \in E$ iff $c^{(\ell)}_{1,j} = 1$, where $j \in \{0, \ldots, k-1\}$;
- $(v_{1,j}, t) \in E$ iff $c^{(1)}_{j,0} = 1$, where $j \in \{0, \ldots, k-1\}$;
- For $\ell - 1 \geq \gamma \geq 2$ and $0 \leq \alpha, \beta \leq k-1$, $(v_{\gamma,\alpha}, v_{\gamma-1,\beta}) \in E$ iff we have $c^{(\gamma)}_{\alpha,\beta} = 1$.

There is no other edge in E.

Given vertices u, v in a graph G, we write $u \rightsquigarrow v$ if there exists a directed path from u to v in G. Our construction is based on the following observation.

Lemma A.1. *Given an ℓ-decomposition $\mathcal{S}$ for $\text{Add}_{k,N}$ and an x,*

$$\text{Add}_{k,N}(x) = 1 \iff s \rightsquigarrow t \text{ in } G_{\mathcal{S},x}.$$

Proof. We provide a sketch of the argument. If $\text{Add}_{k,N}(x) = 1$, consider the sequence of carries generated during the actual computation of $\sum_{i \in [k]} x^{(i)}$ by the standard binary addition algorithm. At least one final carry is generated in this process, since the sum is at least 2^N. The correct carry values computed during intermediate steps of the addition algorithm correspond to a path from s to t in $G_{\mathcal{S},x}$. On the other hand, if there exists a path from s to t in this graph, then an inductive argument starting from t and proceeding backwards to s shows that, during each step of the addition algorithm, at least some number of carries must be produced when we add the integers $x^{(1)}, \ldots, x^{(k)}$. In particular, there must be at least one final carry bit, which implies that $\text{Add}_{k,N}(x) = 1$. □

To sum up, in order to compute $\text{Add}_{k,N}$ from the carry-bit functions it is enough to solve a directed s-t-connectivity problem on a graph with $O(N)$ layers, where each layer contains $O(k)$ vertices.

The computation of the carry-bit functions can be done efficiently in the case of the universal monotone threshold function if we start with an $\Omega(N \log N)$-decomposition. More precisely, each such function can be written as a monotone majority gate over a polynomial number of input bits, which is known to admit efficient monotone circuits as needed in our construction.

Finally, the upper bound follows from the well-known construction of monotone circuits for s-t-connectivity on layered graphs via divide-and-conquer.

REFERENCES

[1] Miklós Ajtai and Yuri Gurevich. 1987. Monotone versus positive. *J. ACM* 34, 4 (1987), 1004–1015.
[2] Eric Allender. 1989. A Note on the Power of Threshold Circuits. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*. 580–584.

[3] Noga Alon and Ravi B. Boppana. 1987. The monotone circuit complexity of Boolean functions. *Combinatorica* 7, 1 (1987), 1–22.

[4] Kazuyuki Amano and Akira Maruoka. 2005. On the Complexity of Depth-2 Circuits with Threshold Gates. In *Proceedings of the 30th International Symposium on Mathematical Foundations of Computer Science*. 107–118.

[5] Alexander E. Andreev. 1985. On a method for obtaining lower bounds for the complexity of individual monotone functions. *Soviet Math. Dokl* 31, 3 (1985), 530–534.

[6] James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. 1994. The Expressive Power of Voting Polynomials. *Combinatorica* 14, 2 (1994), 135–148.

[7] Richard Beigel, Nick Reingold, and Daniel A. Spielman. 1995. PP Is Closed under Intersection. *J. Comput. Syst. Sci.* 50, 2 (1995), 191–202.

[8] Amos Beimel and Enav Weinreb. 2005. Monotone Circuits for Weighted Threshold Functions. In *Proceedings of the 20th Annual IEEE Conference on Computational Complexity*. 67–75.

[9] Olaf Beyersdorff, Edward A. Hirsch, Jan Krajícek, and Rahul Santhanam. 2014. Optimal algorithms and proofs. *Dagstuhl Reports* 4, 10 (2014), 51–68.

[10] Eric Blais, Dominik Scheder, and Li-Yang Tan. 2013. Ajtai-Gurevich Redux. (2013). Manuscript.

[11] Xi Chen, Igor Oliveira, and Rocco Servedio. 2015. Addition is exponentially harder than counting for shallow monotone circuits. arXiv: 1508.03061 (2015).

[12] Michael Dertouzos. 1965. *Threshold Logic: A Synthesis Approach*. MIT Press.

[13] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press.

[14] Yoav Freund and Robert E. Schapire. 1997. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *J. Comput. Syst. Sci.* 55, 1 (1997), 119–139.

[15] Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. 1992. Majority Gates vs. General Weighted Threshold Gates. *Computational Complexity* 2 (1992), 277–300.

[16] Mikael Goldmann and Marek Karpinski. 1993. Simulating threshold circuits by majority circuits. In *Proceedings of the 25th annual ACM Symposium on Theory of Computing*. 551–560.

[17] Johan Håstad. 2010. Some Results in Circuit Complexity. (2010). Presentation at *China Theory Week* (CTW). Slides available at http://conference.itcs.tsinghua.edu.cn/CTW2010/content/Slides/1.pdf.

[18] Johan Håstad and Mikael Goldmann. 1991. On the Power of Small-Depth Threshold Circuits. *Computational Complexity* 1 (1991), 113–129.

[19] Thomas Hofmeister. 1992. The Power of Negative Thinking in Constructing Threshold Circuits for Addition. In *Proceedings of the 7th Annual Structure in Complexity Theory Conference*. 20–26.

[20] Thomas Hofmeister. 1996. A Note on the Simulation of Exponential Threshold Weights. In *Proceedings of the 2nd Annual International Conference on Computing and Combinatorics*. 136–141.

[21] Stasys Jukna. 2012. *Boolean Function Complexity - Advances and Frontiers*. Springer.

[22] Thomson Leighton. 1992. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes*. Morgan Kaufmann.

[23] Marvin Minsky and Seymour Papert. 1968. *Perceptrons - An Introduction to Computational Geometry*. MIT Press.

[24] Saburo Muroga. 1971. *Threshold Logic and its Applications*. Wiley.

[25] Moni Naor and Omer Reingold. 2004. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM* 51, 2 (2004), 231–262.

[26] Ian Parberry. 1994. *Circuit Complexity and Neural Networks*. MIT Press.

[27] Alexander A. Razborov. 1985. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady* 31, 6 (1985), 354–357.

[28] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

[29] Alexander A. Sherstov. 2007. Powering requires threshold depth 3. *Inf. Process. Lett.* 102, 2-3 (2007), 104–107.

[30] Kai-Yeung Siu and Jehoshua Bruck. 1991. On the Power of Threshold Circuits with Small Weights. *SIAM J. Discrete Math.* 4, 3 (1991), 423–435.

[31] Alexei P. Stolboushkin. 1995. Finitely Monotone Properties. In *Proceedings of the 10th Annual IEEE Symposium on Logic in Computer Science*. 324–330.

[32] Éva Tardos. 1988. The gap between monotone and non-monotone circuit complexity is exponential. *Combinatorica* 8, 1 (1988), 141–142.

[33] Alan Taylor and William Zwicker. 1992. A characterization of weighted voting. *Proc. Amer. Math. Soc.* 115, 4 (1992), 1089–1094.

[34] Andrew Chi-Chih Yao. 1989. Circuits and Local Computation. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*. 186–196.

[35] Andrew Chi-Chih Yao. 1990. On ACC and Threshold Circuits. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*. 619–627.