Electronic Colloquium on Computational Complexity, Report No. 199 (2018)

Bootstrapping Results for Threshold Circuits "Just Beyond" Known Lower Bounds

Lijie Chen [*] and Roei Tell [†]

November 24, 2018

Abstract

The best-known lower bounds for the circuit class $\mathcal{TC}^0$ are only slightly super-linear. Similarly, the best-known algorithm for derandomization of this class is an algorithm for quantified derandomization (i.e., a weak type of derandomization) of circuits of slightly super-linear size. In this paper we show that even very mild quantitative improvements of either of the two foregoing results would already imply super-polynomial lower bounds for $\mathcal{TC}^0$. Specifically:

1. If for every $c > 1$ and sufficiently large $d \in \mathbb{N}$ it holds that n-bit $\mathcal{TC}^0$ circuits of depth d require $n^{1+c^{-d}}$ wires to compute certain $\mathcal{NC}^1$-complete functions, then $\mathcal{TC}^0 \neq \mathcal{NC}^1$. In fact, even lower bounds for $\mathcal{TC}^0$ circuits of size $n^{1+c^{-d}}$ against these functions when $c > 1$ is fixed and sufficiently small would yield lower bounds for polynomial-sized circuits. Lower bounds of the form $n^{1+c^{-d}}$ against these functions are already known, but for a fixed $c \approx 2.41$ that is too large to yield new lower bounds via our results.
2. If there exists a deterministic algorithm that gets as input an n-bit $\mathcal{TC}^0$ circuit of depth d and $n^{1+(1.61)^{-d}}$ wires, runs in time $2^{n^{o(1)}}$, and distinguishes circuits that accept at most $B(n) = 2^{n^{1-(1.61)^{-d}}}$ inputs from circuits that reject at most $B(n)$ inputs, then $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$. An algorithm for this "quantified derandomization" task is already known, but it works only when the number of wires is $n^{1+c^{-d}}$, for $c > 30$, and with a smaller $B(n) \approx 2^{n^{1-(30/c)^d}}$.

Intuitively, the "takeaway" message from our work is that the gap between currently-known results and results that would suffice to get super-polynomial lower bounds for $\mathcal{TC}^0$ boils down to the precise constant $c > 1$ in the bound $n^{1+c^{-d}}$ on the number of wires. Our results improve previous results of Allender and Koucký (2010) and of the second author (2018), respectively, whose hypotheses referred to circuits with $n^{1+c/d}$ wires (rather than $n^{1+c^{-d}}$ wires). We also prove results similar to two results above for other circuit classes (i.e., $\mathcal{ACC}^0$ and $\mathcal{CC}^0$).

[*]CSAIL, MIT, Cambridge MA. Email: `lijieche@mit.edu`

[†]Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel. Email: `roei.tell@weizmann.ac.il`

ISSN 1433-8092

Contents

1 Introduction

The current paper studies the long-standing open problems of proving explicit lower bounds (i.e., lower bounds against explicit functions) for several classes of *non-uniform* families of Boolean circuits of *constant depth*. In particular, we focus on the circuit classes $\mathcal{TC}^0$, $\mathcal{ACC}^0$, and $\mathcal{CC}^0$ (for definitions of these classes see Section 3.1).

Our primary focus will be the class $\mathcal{TC}^0$, which consists of constant-depth circuit families with polynomially-many threshold gates (i.e., the gates can compute any linear threshold function).[1] This class is a prominent frontier in the study of circuit lower bounds. One motivation for studying $\mathcal{TC}^0$ is that this class naturally extends, and strictly contains, the well-studied classes $\mathcal{AC}^0$ and $\mathcal{AC}^0[2]$ of constant-depth circuits (for definitions of these circuit classes see Sections 3.1). Specifically, recall that $\mathcal{TC}^0$ circuits can compute any symmetric function (because linear threshold gates can be used to compute the indicator function for any particular Hamming weight of its input). In particular, $\mathcal{TC}^0$ circuits can compute the parity function, which is hard for $\mathcal{AC}^0$ (see [FSS84; Ajt83; Yao85; Hås87]), and the majority function, which is hard for $\mathcal{AC}^0[q]$ for any fixed prime power q (see [Raz87; Smo90]). Another motivation for the study of $\mathcal{TC}^0$ circuits is that they can be viewed as a simplistic model of a neural network with constantly-many hidden layers.

In sharp contrast to the successes in proving lower bounds for $\mathcal{AC}^0$ and for $\mathcal{AC}^0[q]$, the challenge of proving explicit super-polynomial lower bounds for $\mathcal{TC}^0$ has been a major open problem since the 80's. In fact, currently it has not even been ruled out that $\mathcal{TC}^0$ circuits of arbitrary polynomial size can decide $\mathcal{EXP}^{\mathcal{NP}}$. Following the recent lower bounds for the class $\mathcal{ACC}^0$ [Wil13; MW18], the problem of proving lower bounds for $\mathcal{TC}^0$ was highlighted by several authors as a major current frontier in complexity theory (see, e.g., the first open problem in [Aar17]). Currently, the best-known unconditional lower bounds for $\mathcal{TC}^0$ of arbitrary constant depth are only slightly super-linear, and assert that $\mathcal{TC}^0$ circuits of depth d require $n^{1+c^{-d}}$ wires, for $c = 1 + \sqrt{2} \approx 2.41$, to compute the parity function (and other functions in $\mathcal{P}$);[2] these lower bounds were proved by Impagliazzo, Paturi, and Saks [IPS97], and recently extended to average-case lower bounds by Chen, Santhanam, and Srinivasan [CSS16].[3]

A partial explanation for the fact that our lower bounds are only slightly super-linear was given in the work of Allender and Koucký [AK10]. They showed that for various specific $\mathcal{NC}^1$-complete functions (which will be detailed in the next section) the following holds: To prove that the function cannot be computed by $\mathcal{TC}^0$ circuits of depth d_0 and size n^k, it suffices to prove that it cannot be computed by $\mathcal{TC}^0$ circuits of depth $d \gg d_0$ and size $n^{1+c/d}$ (where $c > 1$ is a constant that depends on d_0 and k). Thus, proving "mild" super-linear lower bounds of the form $n^{1+O(1/d)}$ against specific functions would already yield lower bounds against these functions for polynomial-sized $\mathcal{TC}^0$ circuits; such a phenomenon has been recently coined "hardness magnification" by Oliveira and Santhanam [OS18]. Nevertheless, proving lower bounds of the form $n^{1+O(1/d)}$ may already require proofs that are not "natural", in the sense of Razborov and Rudich [RR97]; this is because Miles and Viola [MV15] showed a candidate pseudorandom function computable in $\mathcal{TC}^0$ of depth d with $n^{1+O(1/d)}$ wires.

[1]An alternative common definition for $\mathcal{TC}^0$ allows the gates to only compute the majority function; see Section 3.1 for details.

[2]Throughout the paper, the letter n will always denote the number of inputs to the circuit or function.

[3]Better lower bounds for depth-two circuits (and for depth-three circuits with a top majority gate) were proved by Kane and Williams [KW16]; these lower bounds are still sub-cubic.

An appealing approach to circumvent the potential "natural proofs" obstacle is to use Williams' *algorithmic method* [Wil13]; that is, to construct a deterministic circuit-analysis algorithm for $\mathcal{TC}^0$, thereby obtaining $\mathcal{TC}^0$ lower bounds. In fact, as shown in [Tel18], even a "weak" circuit-analysis algorithm for $\mathcal{TC}^0$ circuits with only $n^{1+O(1/d)}$ wires would suffice to get $\mathcal{TC}^0$ lower bounds. Specifically, recall that in the classical derandomization problem (often called the Circuit Acceptance Probability Problem) we are given as input a description of a Boolean circuit, and want to deterministically decide whether the circuit accepts all but a third of its inputs or rejects all but a third of its inputs. The aforementioned "weak" circuit-analysis problem is the quantified derandomization problem, introduced by Goldreich and Wigderson [GW14]; this is a relaxation of the classical derandomization problem, in which we want to decide whether the circuit accepts almost all of its inputs or rejects almost all of its inputs:

Definition 1 *(quantified derandomization). For a function $B : \mathbb{N} \to \mathbb{N}$, the* quantified derandomization problem for a circuit class $\mathcal{C}$ with B exceptional inputs *is the following: Given an input a description of a circuit $C \in \mathcal{C}$ over n input bits, deterministically decide whether C accepts all but $B(n)$ of its inputs, or C rejects all but $B(n)$ of its inputs.*

Indeed, for $B(n) = 2^n/3$ the quantified problem coincides with the classical problem, but we will be interested in values of $B(n) \ll 2^n/3$, for which the problem may be easier. In [Tel18] it is shown that if there exists an algorithm with runtime $2^{n^{o(1)}}$ for quantified derandomization of $\mathcal{TC}^0$ circuits of depth d and $n^{1+30/d}$ wires and with $B(n) = 2^{n^{1-\exp(-d)}}$ exceptional inputs, then there exists an algorithm for classical derandomization of all of $\mathcal{TC}^0$ (i.e., of arbitrary polynomial size) with runtime $2^{n^{1-\Omega(1)}}$; consequently, using [Wil13; SW13; BV14], it would follow that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$. Nevertheless, the best-known algorithm for quantified derandomization of $\mathcal{TC}^0$ with such a $B(n)$ (also from [Tel18]) can only handle circuits with $n^{1+c^{-d}}$ wires (for some $c > 1$), which falls short of the required size.[4]

To summarize, we currently unconditionally know both lower bounds and algorithms for quantified derandomization for $\mathcal{TC}^0$ circuits with depth d and $n^{1+c^{-d}}$ wires (for some $c > 1$). On the other hand, previous results assert that lower bounds against various specific functions for circuits of size $n^{1+O(1/d)}$, or alternatively quantified derandomization of circuits of such size with a sub-exponential $B(n)$, would yield lower bounds for all of $\mathcal{TC}^0$. We call the latter results bootstrapping results, as they assert that relatively "weak" lower bounds or algorithms for quantified derandomization would suffice to get super-polynomial lower bounds for $\mathcal{TC}^0$.

1.1 Our main contributions

The main contribution of this work is a significant improvement of *both* of the foregoing bootstrapping results. Specifically, we prove that lower bounds for $\mathcal{TC}^0$ circuits of size $n^{1+c^{-d}}$ (for various small values of $c > 1$) against certain specific functions, or quantified derandomization of such circuits with a sub-exponential $B(n)$, would already imply lower bounds for polynomial-sized $\mathcal{TC}^0$ circuits.

Indeed, the size of the circuits in our results is *"just beyond"* the size of circuits for which we already have *unconditional* lower bounds and algorithms for quantified derandomization with a sub-exponential $B(n)$. Thus, loosely speaking, the key take-away from our work is that the difference between what we unconditionally know

[4]In this paper we always measure the size of a circuit as its number of *wires*.

and what we need in order to get super-polynomial lower bounds for $\mathcal{TC}^0$ boils down to the precise constant $c > 1$ in the size bound $n^{1+c^{-d}}$.

1.1.1 Bootstrapping $\mathcal{TC}^0$ lower bounds

Our starting point for the first result is the following known lower bounds. Recall that $\mathcal{TC}^0$ circuits of constant depth $d \in \mathbb{N}$ require $n^{1+\Omega(c^{-d})}$ wires, where $c = 1 + \sqrt{2} \approx 2.41$, in order to compute various $\mathcal{NC}^1$-complete functions (where $\mathcal{NC}^1$-completeness is under $\mathcal{AC}^0$-Karp-reductions). These functions include the Boolean Formula Evaluation problem (BFE), the word problem over S_5 (W_{S_5}), and s-t connectivity on directed graphs of width 5 (W5-STCONN); for definitions of these functions see Section 3.2.[5]

Indeed, since these functions are $\mathcal{NC}^1$-complete, a common conjecture is that $\mathcal{TC}^0$ circuits require super-polynomial size to compute them. Our first result is that in order to prove that these functions cannot be computed by $\mathcal{TC}^0$ circuits of fixed depth d_0 and polynomial size n^k, it suffices to improve the known lower bounds against these functions to be of the form $n^{1+c^{-d}}$ where $c = c(k, d_0) > 1$ is sufficiently small:

Theorem 2 *(hardness magnification for $\mathcal{NC}^1$-complete functions in $\mathcal{TC}^0$). Fix any problem $\Pi \in \{BFE, W_{S_5}, W5\text{-}STCONN\}$. Then, for any $d_0, k \in \mathbb{N}$ there exists $c > 1$ such that the following holds. If for infinitely many constants $d \in \mathbb{N}$ the problem Π cannot be solved by $\mathcal{TC}^0$ circuits of depth d and $n^{1+c^{-d}}$ wires, then the problem Π cannot be solved by $\mathcal{TC}^0$ circuits of depth d_0 and n^k wires.*

Taking one step further, Theorem 2 also implies that lower bounds of the form $n^{1+\exp(-d)}$ against the aforementioned $\mathcal{NC}^1$-complete functions would imply *super-polynomial* lower bounds for $\mathcal{TC}^0$ against these functions. Indeed, such lower bounds would imply that $\mathcal{TC}^0 \neq \mathcal{NC}^1$. [6] Specifically:

Corollary 3 *(lower bounds of the form $n^{1+\exp(-d)}$ suffice to separate $\mathcal{TC}^0$ from $\mathcal{NC}^1$). Fix any problem $\Pi \in \{BFE, W_{S_5}, W5\text{-}STCONN\}$. Assume that for every $c > 1$ there exist infinitely many $d \in \mathbb{N}$ such that $\mathcal{TC}^0$ circuits of depth d need more than $n^{1+c^{-d}}$ wires to solve the problem Π. Then, $\mathcal{TC}^0 \neq \mathcal{NC}^1$.*

Let us elaborate on the precise lower bounds (i.e., values of $c > 1$) that are needed in order to deduce stronger lower bounds via Theorem 2 and Corollary 3. Working out the precise parameters underlying Theorem 2 (see Corollary 21), it turns out that to resolve long-standing open problems it suffices to prove lower bounds for a fixed $c > 1$ that is not so small. For example, to prove cubic lower bounds for depth-two $\mathcal{TC}^0$ circuits (i.e., $\mathsf{LTF} \circ \mathsf{LTF}$ circuits[7]) it suffices to prove lower bounds of the form $n^{1+c^{-d}}$ for $c \approx 1.18$. Also, turning to Corollary 3, the hypothesis that lower bounds of the form $n^{1+c^{-d}}$ hold for *every* $c > 1$ might seem puzzling at first (since when c tends to one the size bound becomes close to quadratic). Note, however, that when using

[5]The reason that these functions require circuits of size $n^{1+c^{-d}}$ is that computing parity can be reduced to computing any of these functions with a linear overhead; see [AK10, Sec. 1.2] for further details.

[6]These functions are actually in uniform-$\mathcal{NC}^1$, so such lower bounds would imply that uniform-$\mathcal{NC}^1 \not\subseteq \mathcal{TC}^0$. However, the latter statement is equivalent to $\mathcal{TC}^0 \neq \mathcal{NC}^1$ (see Proposition 15).

[7]Recall that we defined $\mathcal{TC}^0$ using linear threshold gates, rather than majority gates. Exponential lower bounds for $\mathsf{MAJ} \circ \mathsf{MAJ}$ circuits are known [Haj+93; For+01].

Corollary 3 we are allowed to first fix $c > 1$, and then prove a lower bound only when the depth d is arbitrarily large (such that c^{-d} is arbitrarily small).[8]

We also show results similar to Theorem 2 and to Corollary 3 for the Boolean iterated matrix multiplication problem for matrices of size $n^{o(1)}$, which is in $\mathcal{NL}$ (see Section 3.2.2, and the discussion after Theorem 24).

Lastly, we mention that the results in this section hold not only for $\mathcal{TC}^0$, but for essentially any constant-depth circuit class (e.g., also for $\mathcal{ACC}^0$ and for $\mathcal{CC}^0$; see Theorems 19 and 26). However, we consider the results to be most appealing for $\mathcal{TC}^0$, for which near-matching lower bounds are already known. (The reason that we know near-matching lower bounds for $\mathcal{TC}^0$ but not for, say, $\mathcal{ACC}^0$, is that when referring only to circuits of size $n^{1+c^{-d}}$ for $c > 2$ the two classes are incomparable.)

1.1.2 Bootstrapping derandomization algorithms for $\mathcal{TC}^0$

Loosely speaking, our second main result is that a quantified derandomization algorithm for $\mathcal{TC}^0$ circuits of depth d and size $n^{1+c^{-d}}$ with a sub-exponential $B(n)$ would yield a corresponding algorithm for standard derandomization of all $\mathcal{TC}^0$, and hence imply that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$. Moreover, the hypothesis in the foregoing statement refers to a *fixed* constant $c \approx 1.61$; that is, the hypothesis refers to $\mathcal{TC}^0$ circuits of a specific size but nevertheless implies lower bounds for all of $\mathcal{TC}^0$.

Being more specific, the size of the circuits that the hypothesis refers to is "just beyond" the size required to compute the *parity* function, against which our best-known $\mathcal{TC}^0$ lower bounds hold. Specifically, let $\phi \geq \frac{1+\sqrt{5}}{2}$ such that for any $d \geq 2$, parity can be computed by $\mathcal{TC}^0$ circuits of depth d and size $n^{1+O(\phi^{-d})}$ (the bound on ϕ is due to the construction of Paturi and Saks [PS94], following [BBL92]). Then, the hypothesis in the following theorem refers to $\mathcal{TC}^0$ circuits of depth d and $n^{1+c^{-d}}$ wires, where $c > 1$ can be any fixed constant smaller than ϕ.

Theorem 4 *(a bootstrapping result for derandomization of $\mathcal{TC}^0$). Let $c > 1$ be any fixed constant smaller than ϕ (e.g., $c = 1.61$). Assume that for every sufficiently large $d \in \mathbb{N}$ there exists an algorithm for quantified derandomization of $\mathcal{TC}^0$ circuits with $n^{1+c^{-d}}$ wires and with $B(n) = 2^{n^{1-c^{-d}}}$ that runs in time $2^{n^{o(1)}}$. Then, there exists an algorithm for standard derandomization of $\mathcal{TC}^0$ that runs in time $2^{n^{o(1)}}$.*

Using known results that show that standard derandomization of $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$ (i.e., Williams' "algorithmic method", applied to the special case of $\mathcal{TC}^0$; see [Wil13; SW13; BV14]), we deduce that quantified derandomization as in the hypothesis of Theorem 4 implies lower bounds for $\mathcal{TC}^0$.

Corollary 5 *(quantified derandomization of sparse $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$). Assume that there exists an algorithm as in the hypothesis of Theorem 4. Then, $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$.*

The currently-known algorithm for quantified derandomization of $\mathcal{TC}^0$ works when both the size of the circuit and the number of exceptional inputs are slightly smaller than in the hypothesis in Theorem 4 and Corollary 5; specifically, the known algorithm works for circuits of size $n^{1+c^{-d}}$, where $c > 30$, and with $B(n) = 2^{n^{1-\delta}}$, where

[8]Alternatively, any lower bound of the form $n^{1+2^{-f(d)}}$ where f is a sub-linear function (e.g., the bound $n^{1+2^{-d/\log^*(d)}}$) also satisfies the hypothesis of Corollary 3.

$\delta = (d/30) \cdot (30/c)^d$ (see [Tel18, Thm. 5.1]).[9] Nevertheless, the running time of the known algorithm is $n^{\text{poly}\log\log(n)}$, whereas the running time of the algorithm in the hypothesis in Corollary 5 may be much larger (i.e., $2^{n^{o(1)}}$). Moreover, we can further relax the requirements from the algorithm in the hypothesis in Corollary 5, relying on known relaxations of Williams' algorithmic method (see Corollary 45 for details).

A main technical result that underlies Theorem 4 is a construction of uniform $\mathcal{TC}^0$ circuits of depth $d \geq 7$ and size $n^{1+\exp(-d)}$ that compute all the outputs of a seeded extractor on a given input. Specifically, we prove that there exists an extractor $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ for min-entropy $k = n^{1-\exp(-d)}$ with $m = n^{\exp(-d)}$ output bits and seed length $t = (1 + \exp(-d)) \cdot \log(n)$ such that the mapping of $x \in \{0,1\}^n$ to $\{Ext(x,z)\}_{z \in \{0,1\}^t}$ is computable by a uniform depth-d circuit of size $n^{1+\exp(-d)}$ (see Theorem 8). Note that the latter circuit has $2^t \cdot m = n^{1+\exp(-d)}$ output bits and only $n^{1+\exp(-d)}$ wires; thus, the circuit essentially performs an efficient "batch computation" of all the outputs of the extractor (i.e., the outputs corresponding to all seeds) on a given input $x \in \{0,1\}^n$. This construction relies, in turn, on a construction of uniform $\mathcal{TC}^0$ circuits of depth d and size $n^{1+\exp(-d)}$ that compute a code with constant relative distance and a slightly sub-constant rate (see Theorem 10).

Finally, relying on Corollary 5, we suggest a potential path towards super-polynomial lower bounds for $\mathcal{TC}^0$. Loosely speaking, quantified derandomization of a circuit $C : \{0,1\}^n \to \{0,1\}$ with $B(n)$ exceptional inputs reduces to deterministically finding a representation of a "simple" function that approximates C in a subset $S \subseteq \{0,1\}^n$ of size $|S| \gg B(n)$ (see Section 6 for a precise statement). Thus, Corollary 5 implies that to prove that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$, it suffices to find a representation of a "simple" function that approximates a $\mathcal{TC}^0$ circuit in a subset of subexponential size, when the circuit is of size that is "just beyond" the size required to compute the parity function. We discuss this issue more precisely and pose a concrete open problem in Section 6.

1.2 Bootstrapping results for $\mathcal{ACC}^0$ and for $\mathcal{CC}^0$

Our main results in the previous sections focused proving lower bounds for the class $\mathcal{TC}^0$ against classes such as $\mathcal{NC}^1$ and $\mathcal{NEXP}$. In the current section we focus on the problems of proving separations among *potentially weaker* classes. Specifically, we focus on the problems of separating $\mathcal{ACC}^0$ from $\mathcal{TC}^0$, and of separating $\mathcal{CC}^0$ from $\mathcal{ACC}^0$. Similarly to our $\mathcal{TC}^0$ results, the hypotheses in the current section refer to lower bounds or to quantified derandomization for circuits of size $n^{1+\Omega(1)}$; however, in contrast to our $\mathcal{TC}^0$ results, the corresponding *known* lower bounds for the classes in the current section refer only to circuits of even smaller (super-linear) size.

1.2.1 Bootstrapping lower bounds and derandomization of $\mathcal{ACC}^0$

Recall that $\mathcal{ACC}^0$ is the class of constant-depth circuit families with polynomially-many gates that can compute the $\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}$ functions as well as the MOD_m function, for any fixed integer $m \in \mathbb{N}$ (i.e., the Boolean function that evaluates to one if and only if the sum of its inputs is zero modulo m).

The class $\mathcal{ACC}^0$ is a subclass of $\mathcal{TC}^0$, and a widely-believed conjecture of Barrington [Bar89, Sec. 7] is that $\mathcal{ACC}^0 \neq \mathcal{TC}^0$; that is, the conjecture is that MAJ re-

[9]The parameters of the known construction can be optimized such that the number 30 will be replaced by some other constant $\alpha > 3$. However, even after such an optimization, the parameters of the known construction will still be weaker than those required in the hypothesis in Corollary 5.

quires $\mathcal{ACC}^0$ circuits of super-polynomial size. Nevertheless, as far as we are aware of, there are currently no known super-linear lower bounds for computing MAJ by $\mathcal{ACC}^0$ circuits. Analogously to Corollary 3, we show that proving super-linear lower bounds of the form $n^{1+\exp(-d)}$ for computing MAJ by $\mathcal{ACC}^0$ circuits would imply super-polynomial lower bounds for computing MAJ by $\mathcal{ACC}^0$ circuits; that is, in order to prove that $\mathcal{ACC}^0 \neq \mathcal{TC}^0$ it suffices to prove lower bounds of the form $n^{1+\exp(-d)}$:

Theorem 6 *(hardness magnification for MAJ in $\mathcal{ACC}^0$). Assume that for every constant $c > 1$ there exist infinitely many $d \in \mathbb{N}$ such that MAJ cannot be computed by $\mathcal{ACC}^0$ circuits of depth d with $n^{1+c^{-d}}$ wires. Then, MAJ $\notin \mathcal{ACC}^0$, and consequently $\mathcal{ACC}^0 \neq \mathcal{TC}^0$.*

We also consider the problem of derandomization of $\mathcal{ACC}^0$ with "one-sided error"; that is, the problem of deterministically distinguishing $\mathcal{ACC}^0$ circuits with acceptance probability one from $\mathcal{ACC}^0$ circuits with acceptance probability at most half. The best currently-known algorithm for this problem is the *satisfiability* algorithm of Williams [Wil11], which runs in time $2^{n-n^{\Omega(1)}}$ (and distinguishes circuits with acceptance probability one from circuits with acceptance probability less than one).

Nevertheless, we do have much faster algorithms for *quantified* derandomization of limited subclasses of $\mathcal{ACC}^0$. Specifically, we currently have *polynomial-time* algorithms for quantified derandomization of "structured subclasses" of $\mathcal{AC}^0[\oplus]$ of depth three and small super-linear size (e.g., size $O(n)$ or $n + n^{\Omega(1)}$) with a subexponential $B(n)$; for precise details see [GW13, Sec. 6] and [Tel17a, Sec. 6.2]. We show that improving these results to quantified derandomization of $\mathcal{ACC}^0$ circuits of depth $d \geq 5$ and size $n^{1+\gamma_d}$ with a subexponential $B(n) \approx 2^{n^{1-\gamma_d/8}}$, where $\gamma_d > 0$ can be any sufficiently small constant, would imply corresponding algorithms for standard derandomization of $\mathcal{ACC}^0$ with "one-sided error". See Section 5.4 for further details.

1.2.2 Bootstrapping $\mathcal{CC}^0$ lower bounds

Recall that $\mathcal{CC}^0$ is the class of constant-depth circuit families with polynomially-many gates that can compute *only* the MOD_m function, for any fixed integer $m \in \mathbb{N}$. The class $\mathcal{CC}^0$ is a subclass of $\mathcal{ACC}^0$, and a common conjecture is that $\mathcal{CC}^0 \neq \mathcal{ACC}^0$; that is, that $\mathcal{CC}^0$ circuits require super-polynomial size to compute the AND function. Nevertheless, the best-known lower bounds for computing AND by $\mathcal{CC}^0$ circuits, which were proved by Chattopadhyay *et al.* [Cha+06], are of the form $n \cdot f_d(n)$, where $f_d(n)$ is a slowly-growing function that depends on the circuit depth d (i.e., $\omega(1) \leq f_d(n) \leq \log^*(n)$ for any $d \geq 2$). Similarly to Corollary 3 and to Theorem 6, we show that improving the latter lower bound to be of the form $n^{1+\exp(-d)}$ would imply super-polynomial lower bounds for computing AND in $\mathcal{CC}^0$:

Theorem 7 *(hardness magnification for AND in $\mathcal{CC}^0$). Assume that for every constant $c > 1$ there exist infinitely many $d \in \mathbb{N}$ such that AND cannot be computed by $\mathcal{CC}^0$ circuits of depth d with $n^{1+c^{-d}}$ wires. Then, $\mathcal{CC}^0 \neq \mathcal{ACC}^0$.*

Moreover, similarly to Theorem 2, proving lower bounds for specific and sufficiently small values of $c > 1$ would yield *polynomial* lower bounds for computing AND in $\mathcal{CC}^0$; see Corollary 23.

1.3 Organization

In Section 2 we present high-level overviews of the proofs of our main results. In Section 3 we review standard definitions of the notions in our paper and state some well-known results regarding these notions. In Section 4 we prove our "hardness magnification" results; that is, Theorem 2, Corollary 3, and Theorems 6 and 7. In Section 5 we prove our result regarding quantified derandomization; that is, Theorem 4, Corollary 5, and the result regarding quantified derandomization of $\mathcal{ACC}^0$ (that was mentioned in the end of Section 1.2.1). Finally, in Section 6 we present an open problem and show that its resolution would imply that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$.

2 Proof overviews

As mentioned in the introduction, the results in this paper strengthen the results both in [AK10] and in [Tel18]. Our high-level proof strategies follow the same strategies outlined in these papers, and we obtain the stronger results by improving the technical tools underlying both papers. Specifically, to prove our results we construct uniform $\mathcal{TC}^0$ circuit families of size $n^{1+\exp(-d)}$ for solving various computational tasks: Among those are the tasks of encoding a *balanced error-correcting code*; of computing the outputs of an *averaging Boolean sampler* (equivalently, a seeded randomness extractor); and of computing strong *self-reductions* for natural computational problems (e.g., for the monoid problem, for BFE, and for BIMM).

2.1 Bootstrapping lower bounds

We first present the proof idea behind our "hardness magnification" results (i.e., the results in Sections 1.1.1, 1.2.1, and 1.2.2). The main idea here is actually quite simple, and should be accessible even to readers without specific prior knowledge.

Following Allender and Koucký [AK10], our starting point is the monoid problem: Given n elements from a monoid Σ of constant size,[10] the output is the multiplication of the n elements over Σ (for a precise definition see Section 3.2). Let us recall the original proof from [AK10]: For essentially any circuit class and monoid, assuming that the monoid problem can be solved in size n^k and depth $d_0 = O(1)$, they show that for infinitely many $d \geq d_0$, the problem can also be solved in depth d and size $n^{1+O(1/d)}$. To do so, for any $\epsilon > 0$, they partition the n inputs into blocks of size $n^{\epsilon/k}$, compute the function on each block (which can be done in size n^{ϵ}) and obtain $n^{1-\epsilon/k}$ inputs; then they partition the latter inputs into blocks of size $n^{\epsilon/k}$ and compute the function on each block, and repeat this process until they end up with a single block and compute the final output. The point is that this process computes the function correctly due to the *associativity* of the monoid operation. The size of this circuit is dominated by the bottom layer of circuits (i.e., the layer just above the original n inputs), which is of size $n^{\epsilon} \cdot n^{1-\epsilon/k} < n^{1+\epsilon}$, and the depth of the circuit is $(k \cdot d_0)/\epsilon$.

Let us abstract this construction, and represent it by a tree. The inputs are represented by n nodes at the bottom, and each sub-circuit (on $n^{\epsilon/k}$ inputs) is represented by an internal node, which is connected to nodes in the layer beneath it.[11] Fixing a cost function $c(m) = m^k$, we define the cost of the tree to be the sum, over all nodes, of

[10]Recall that a monoid is simply a set with an associative binary operation and an identity element.

[11]This representation hides the fact that each subcircuit is of depth d_0, and just represents it by a single node. Indeed, this fact does not crucially affect our analysis.

the cost of the degree of the node; that is, $Cost(T) = \sum_{v \in Nodes(T)} c(\deg(v))$. The construction from [AK10] thus corresponds to a perfect $(n^{\epsilon/k})$-ary tree of depth $d = k/\epsilon$, which has cost $O(n^{1+\epsilon})$. We therefore ask the following question: Given $\epsilon > 0$, can we construct a tree with cost $O(n^{1+\epsilon})$ that is of depth smaller than k/ϵ? The answer to this problem turns out to be positive: The minimum-depth tree for cost $O(n^{1+\epsilon})$ has depth only $d \approx k \cdot \ln(1/\epsilon)$, instead of $d = k/\epsilon$. This tree is constructed by a *greedy* strategy: Starting from the top node, at each level of the tree we set the fan-in of all nodes in the level such that the total cost of the level will be $n^{1+\epsilon}$.

Let us be more specific, and prove that this yields our claimed improvement. For $i \geq 0$, we denote the number of nodes at level i (i.e., at distance i from the root, which is the top node) by n_i, and set their fan-in to be $\left(n^{(1+\epsilon)}/n_i\right)^{1/k}$, such that the cost of this level is $n^{1+\epsilon}$. (Indeed, at the root we have that $n_0 = 1$ and the fan-in is $n^{(1+\epsilon)/k}$.) Then, the number of nodes at level $i+1$ is $n_{i+1} = n_i \cdot \left(n^{(1+\epsilon)}/n_i\right)^{1/k}$. Solving this recursion, we have that $n_i = n^{\alpha_i}$, where $\alpha_i = (1+\epsilon) \cdot \left(1 - (1-1/k)^i\right)$. At the last (bottom) level d we will have n^{α_d} nodes, and we will connect them to n inputs; thus, their fan-in will be $n^{1-\alpha_d}$, and the cost of level d will be $n^{\alpha_d + k \cdot (1-\alpha_d)} = n^{k-(k-1)\cdot\alpha_d}$. Setting $d = k \cdot \ln\left(\frac{(1+\epsilon)\cdot(1-1/k)}{\epsilon}\right)$, we have that $\alpha_d = 1 - \epsilon/(k-1)$, and thus the cost of the last level d is also $n^{1+\epsilon}$, and we are finished.[12]

The construction above can be viewed as a generalization of a construction of Beame, Brisson, and Ladner [BBL92] for computing parity in sparse $\mathcal{TC}^0$. They originally generalized their construction to all symmetric functions, whereas we generalize the construction to any instance of the monoid problem. (For the special case of parity, the subsequent optimized construction in [PS94] achieves better parameters.)

We stress that the foregoing proof did not depend on the particular circuit class; indeed, the proof holds for essentially any constant-depth circuit class. The proofs of our other hardness magnification results are a bit more involved, and depend on details of the specific computational problem, but they nevertheless rely on the same core idea that is represented by the improved tree structure. We refer the reader to the beginning of Section 4, which includes an index with pointers to these proofs.

2.2 Bootstrapping derandomization of $\mathcal{TC}^0$

Towards presenting the proof of our second main result, we say that a circuit family $\{C_n : \{0,1\}^n \to \{0,1\}\}_{n \in \mathbb{N}}$ of constant depth d is extremely sparse if for any sufficiently large $n \in \mathbb{N}$, the number of wires in C_n is at most $n^{1+c^{-d}}$, for some constant $c > 1$.

We prove Theorem 4 by showing an efficient reduction of *standard* derandomization of *all* of $\mathcal{TC}^0$ to *quantified* derandomization of *extremely sparse* $\mathcal{TC}^0$. Specifically, we construct an algorithm that is given a $\mathcal{TC}^0$ circuit C of depth d_0 over m input bits, and outputs an extremely sparse $\mathcal{TC}^0$ circuit C' of depth $d > d_0$ over $n = \text{poly}(m)$ input bits such that if C accepts (resp., rejects) at least $2/3$ of its inputs then C' accepts (resp., rejects) all but $B(n) = 2^{n^{1-\exp(-d)}}$ of its inputs. The circuit C' will use its input in order to sample inputs for C by a seeded extractor (equivalently, by an averaging sampler), and output the majority of the evaluations of C on these inputs.

[12]For completeness, in Appendix B we prove that the foregoing greedy solution is optimal. Alternatively, to quickly see that the optimal solution will yield $d = O(\log(1/\epsilon))$, note that *parity* is a monoid problem, and recall that computing parity in $\mathcal{TC}^0$ of depth d requires $n^{1+\exp(-d)}$ wires [IPS97].

The main technical challenge that underlies this approach is constructing an extractor $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that the mapping of input $x \in \{0,1\}^n$ to the 2^t m-bit outputs of the extractor on all seeds can be computed by *uniform extremely sparse $\mathcal{TC}^0$ circuits*. This is indeed the technical result that underlies Theorem 4, and it improves a previous construction from [Tel18], in which the circuit had $n^{1+O(1/d)}$ wires. As mentioned in the introduction, our circuit has $2^t \cdot m = n^{1+\exp(-d)}$ output bits but it uses only $n^{1+\exp(-d)}$ wires, and so to construct it we need to perform an efficient "batch computation" of the extractor on all seeds.

Theorem 8 *(an extractor in extremely sparse $\mathcal{TC}^0$). There exists a polynomial-time algorithm that gets as input 1^n and a constant $d \geq 7$, and outputs a $\mathcal{TC}^0$ circuit $C : \{0,1\}^n \to (\{0,1\}^m)^s$ of depth d and size $n^{1+\exp(-d)}$ that satisfies the following: The function $Ext(x,i) = C(x)_i$ is a (k,ϵ)-extractor for min-entropy $k = n^{1-\exp(-d)}$, with output length $m = n^{\exp(-d)}$, seed length $t = \log(s) = (1+\exp(-d)) \cdot \log(n)$ and error $\epsilon = 1/m$.*

Recall that Theorem 4 asserts that standard derandomization reduces to quantified derandomization of circuits with $n^{1+c^{-d}}$ wires, for any $c < \phi = \frac{1+\sqrt{5}}{2}$. By carefully accounting for the parameters in our construction, the number of wires in the circuit from Theorem 8 is indeed $n^{1+c^{-d}}$, for any $c < \phi$ (see Proposition 43). However, in the current high-level overview we will not care about the specific constant c.

As a first step, let us describe a *non-uniform* construction of $\mathcal{TC}^0$ circuits as in Theorem 8; that is, we will first show that such circuits *exist*, and later show that they can be constructed uniformly. The underlying extractor will be Trevisan's extractor [Tre01]. In order to compute the mapping $x \mapsto \{Ext(x,z)\}_{z\in\{0,1\}^t}$ when Ext is Trevisan's extractor, we first need to encode x to a codeword $\bar{x}$ of a *balanced code* (i.e., every codeword has relative Hamming weight close to $1/2$), and then determine the $s = 2^t$ outputs of the extractor as projections of the bits of $\bar{x}$, according to appropriate *combinatorial designs* (in the sense of Nisan and Wigderson [NW94]). Note that, crucially, to compute the all the outputs of the extractor on a given input $x \in \{0,1\}^n$, we only need to encode $x \mapsto \bar{x}$ *once*, and all the outputs are projections of bits of $\bar{x}$.

Our non-uniform circuit first encodes $x \in \{0,1\}^n$ to a codeword $\hat{x} \in \{0,1\}^{O(n)}$ of a code with constant rate and *constant relative distance*. Gál *et al.* [Gál+13] showed that there exist depth-two circuits with *only parity gates* that can compute such a code using only $n \cdot \text{poly}\log(n)$ wires. To convert such a "parity-gates" circuit into a $\mathcal{TC}^0$ circuit, we replace each parity gate g with fan-in n_g in by a $\mathcal{TC}^0$ circuit computing parity with depth d and $n_g^{1+\exp(-d)}$ wires, using the constructions of [BBL92; PS94]. The resulting $\mathcal{TC}^0$ circuit has depth $2d$, and its number of wires is at most $\sum_{g \text{ gate}} \left(n_g^{1+\exp(-d)}\right) < \left(\sum_g n_g\right)^{1+\exp(-d)} \leq (n \cdot \text{poly}\log(n))^{1+\exp(-d)} = n^{1+\exp(-d)}$.

We now amplify the distance of the code from $\Omega(1)$ to approximately $1/2$, using the strategy of Naor and Naor [NN93]. Specifically, we map $\hat{x}$ to a new codeword $\bar{x}$ such that every bit of $\bar{x}$ corresponds to a walk of an appropriate length ℓ on an expander on $[|\hat{x}|] = [O(n)]$, and the bit in $\bar{x}$ is the parity of the bits of $\hat{x}$ encountered during this walk. Trevisan's extractor requires the distance to be $1/2 - \delta$, where $\delta \approx 1/m^2 = 1/n^{\exp(-d)}$; to get such a distance, it suffices for the walk to be of length $O(\log(1/\delta))$, and thus the length of $\bar{x}$ is $O(n \cdot \text{poly}(1/\delta)) = n^{1+\exp(-d)}$ (assuming that $n = \text{poly}(m)$ is sufficiently large). Since each bit in $\bar{x}$ is the parity of $O(\log(n))$ bits in $\hat{x}$, we can compute each bit in $\bar{x}$ by a depth-two circuit with $O(\log^2(n))$ wires, and the resulting circuit will be of depth $2d+2$ and $n^{1+\exp(-d)}$ wires.

Finally, the circuit outputs projections of $\bar{x}$ according to predetermined combinatorial designs. We will use *weak designs*, introduced by Raz, Reingold, and Vadhan [RRV02], which suffice for Trevisan's extractor. Instantiating a construction of weak designs from [Tel18] to our parameter setting, in which the required min-entropy is $n^{1-\exp(-d)}$, there exist weak designs in a universe of size $t = (1 + \exp(-d)) \cdot \log(n)$ (see Lemma 41). Thus we have that $s = 2^t = n^{1+\exp(-d)}$ as we wanted.

The remaining challenge is to turn the construction above into a *uniform* construction. Most parts of the construction are already uniform: The circuits for the distance amplification step, the $\mathcal{TC}^0$ circuits for computing parity, and the weak designs, can all be constructed in polynomial time. Thus, the only "non-uniform" part is the sparse "parity-gates" circuit that computes a code with constant relative distance. Indeed, Gál *et al.* [Gál+13] posed the construction of a uniform "parity gate" circuit with parameters matching the ones of their non-uniform construction as an important open problem, noting that this problem is closely related to the long-standing open problems of explicitly constructing superconductors and unbalanced lossless expanders.

Fortunately, for our purposes we can afford a mild degradation in the parameters of the code construction. Specifically, our code does not need to have constant rate, since rate $1/n^{o(1)}$ also suffices for our purposes (because we map $\hat{x}$ to $\bar{x}$ of length $|\hat{x}|^{1+\exp(-d)}$ in the distance amplification step anyway). Also, the "parity gates" circuit that we start from does not need to have $n \cdot \mathrm{poly}\log(n)$ wires, and a circuit with $n^{1+o(1)}$ wires also suffices for our purposes (because we convert each gate g into a $\mathcal{TC}^0$ circuit with $n_g^{1+\exp(-d)}$ wires anyway). Indeed, we construct a uniform "parity gates" circuit of depth two with $n \cdot \exp(\mathrm{poly}\log\log(n))$ wires that computes a code with rate $1/n^{\exp(\mathrm{poly}\log\log(n))}$ and constant relative distance (see Proposition 31).

The construction of this code (and circuit) appears in Section 5.1. Let us now describe this construction, at a high level. Since we will only use parity gates, our code will be linear; hence, to get constant relative distance, we only need to ensure that on any non-zero input $x \in \{0,1\}^n$ a constant fraction of the output gates will be set to one. The basic building-blocks in our construction are *range detectors*, as defined in [Gál+13]. Intuitively, these are functions that map any n-bit input with Hamming weight between $w/2$ and w (for some $w \in [n]$) to an m-bit output with Hamming weight between $.01m$ and $.99m$, using only *parity* gates. More formally:

Definition 9 *(range detectors). An (n, m, w, ℓ, h)-*range detector *is a function $D : \{0,1\}^n \to \{0,1\}^m$ such that every output bit of D is a parity of input bits, and for any $x \in \{0,1\}^n$ with Hamming weight in $[w/2, w]$ it holds that the Hamming weight of $D(x)$ is in $[\ell, h]$.*

We are interested in constructing, for every $w < n$, an $(n, m, w, .01m, .99m)$-range detector, for some m, that only uses *few* wires (i.e., it only uses $n^{1+o(1)}$ wires).[13] In [Gál+13] it is shown that for every $w < n$, there *exist* $(n, m, w, .01m, .99m)$-range detectors with $m = O(w \cdot \log(n))$ that can be computed by a layer of parity gates with only $O(n \cdot \log(n))$ wires (see [Gál+13, Sec. 1.2 and Cor. 19]). For our uniform circuit, we need to *explicitly* construct range detectors; we do this relying on the explicit construction of *unbalanced lossless expanders* by Capalbo *et al.* [Cap+02]. Specifically, in Section 5.1.1 we rely on the latter expanders to explicitly construct $(n, m, w, .01m, .99m)$-range detectors with $m = O(w \cdot \exp(\mathrm{poly}\log\log(n)))$ that can be computed by a layer of parity gates with $n \cdot \exp(\mathrm{poly}\log\log(n))$ wires.

[13]Indeed, for $w = \Omega(n)$ this is easy (e.g., for $.02n \leq w \leq .99n$, the identity mapping with $n = m$ yields a suitable range detector), and the main challenge is to handle smaller values of w.

In the circuit that encodes our code, the layer above the input gates contains $\log(n)$ range detectors: For each $i = 1, ..., \log(n)$ and $w_i = 2^i$, the layer contains an $(n, m_i, w_i, .01m_i, .99m_i)$-range detector with $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires and $m_i = w_i \cdot \exp(\operatorname{poly}\log\log(n))$ outputs. Thus, this layer contributes overall $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires to the circuit. We add to this construction a top layer of $n_0 = \max_{i\in[\log(n)]}\{m_i\} = \exp(\operatorname{poly}\log\log(n))$ gates with the following property: If a constant fraction of the outputs of at least one range detector are set to one, then a constant fraction of the top gates *touch* a gate in the middle layer that is set to one.[14] To do so, for each range detector, we split the n_0 top gates into m_i blocks of n_0/m_i gates, and connect each output of the range detector to all top gates in the corresponding block.

Thus, for any non-zero $x \in \{0,1\}^n$, a constant fraction of the top gates touch a gate in the middle layer that is set to one. Now, note that each of the top n_0 gates has degree exactly $\log(n)$. Using another idea from [Gál+13], we replace each such gate g with $O(\log(n))$ parity gates that compute a good error-correcting code on the $\log(n)$ gates that fed to g. Hence, if g touched a middle gate that is set to one, then a constant fraction of the $O(\log(n))$ gates that replaced g are set to one. Thus, on any non-zero input $x \in \{0,1\}^n$, a constant fraction of the top gates will be set to one, and it follows that our linear code has constant relative distance. The number of gates in the top layer is $\hat{n} = n_0 \cdot O(\log(n)) = n \cdot \exp(\operatorname{poly}\log\log(n))$, and each of them has degree $\log(n)$, yielding overall $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires between the middle layer and the top layer. Finally, replacing parity gates by $\mathcal{TC}^0$ circuits, we get the following:

Proposition 10 *(uniform extremely sparse $\mathcal{TC}^0$ circuit for encoding an "almost-good" code; see Proposition 35). For every $d \geq 4$ there exists a uniform family of $\mathcal{TC}^0$ circuits of depth d that, for every $n \in \mathbb{N}$, encode a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance, where $\hat{n} = n \cdot \exp(\operatorname{poly}\log\log(n))$, using at most $n^{1+\exp(-d)}$ wires.*

Note that the use of $n^{1+\exp(-d)}$ wires in the circuit in Theorem 10 is unavoidable if we want the code to be linear (since computing even a single parity of $\Omega(n)$ bits in $\mathcal{TC}^0$ requires $n^{1+\exp(-d)}$ wires).

3 Preliminaries

We denote by $\exp(n)$ any function of the form c^n for some constant $c > 1$. We denote $\log(x) = \log_2(x)$, and $\ln(x)$ will be the natural logarithm. We typically denote random variables by boldface, and uniform random variables over $\{0,1\}^n$ by $\mathbf{u}_n$.

3.1 Circuit complexity classes

A circuit family is a collection of circuits $\{C_n : \{0,1\}^n \to \{0,1\}^{m(n)}\}$, for some function $m : \mathbb{N} \to \mathbb{N}$. A circuit class is a collection of circuit families. The size of a circuit is the number of *wires* in the circuit, and the size of a circuit family is a function of the input length that upper-bounds the size of circuits in the family. We will mainly consider classes in which the size of each circuit family is bounded by some polynomial; however, for a circuit class $\mathcal{C}$, we will sometimes also abuse notation by referring to $\mathcal{C}$-circuits with various other size bounds. Unless explicitly stated otherwise, the gates in all circuit classes that we consider have *unbounded fan-in*.

[14]This does not yet suffice for the full construction, since a top gate might touch an *even* number of gates in the middle layer that are set to one.

Some circuit classes that we consider cannot compute any function (see, e.g., $\mathcal{CC}[2]$ below). We say that a circuit class is complete if circuits of sufficiently large size from the class can compute any function. Intuitively, any class whose gates can compute a complete Boolean basis (e.g., AND, OR, and NOT) is complete.

Definition 11 *(the classes $\mathcal{CC}^0[q]$ and $\mathcal{AC}^0[q]$). For an integer q, the function $MOD_q : \{0,1\}^* \to \{0,1\}$ is one if and only if the number of ones in the input is not divisible by q. The class $\mathcal{CC}^0[q]$ is the class of constant-depth circuit families with polynomially-many unbounded fan-in MOD_q gates. We denote $\mathcal{CC}^0 = \cup_{q\geq 2}\mathcal{CC}^0[q]$. Extending $\mathcal{CC}^0[q]$, the class $\mathcal{AC}^0[q]$ is the class of constant-depth circuit families consisting of polynomially-many unbounded fan-in AND, OR and MOD_q gates, along with unary NOT gates. We denote $\mathcal{ACC}^0 = \cup_{q\geq 2}\mathcal{AC}^0[q]$.*

Recall that for any prime power q it holds that $\mathcal{CC}^0[q]$ is *not* complete (since a constant-depth $\mathcal{CC}^0[q]$ circuit of any size computes a polynomial of degree $\mathrm{poly}(q) = O(1)$ over $\mathbb{F}_q$), whereas if q is composite then $\mathcal{CC}^0[q]$ is complete (see [BST90]).

Definition 12 *(the class $\mathcal{TC}^0$). A linear threshold function (LTF) is a function $\Phi : \{0,1\}^n \to \{0,1\}$ of the form $\Phi(x) = \mathrm{sgn}(\langle x, w\rangle - \theta)$, where $w \in \mathbb{R}^n$ and $\theta \in \mathbb{R}$ and $\langle x, w\rangle = \sum_{i\in[n]} x_i \cdot w_i$ is the standard inner-product over $\mathbb{R}$.* [15] *The class $\mathcal{TC}^0$ is the class of constant-depth circuit families with polynomially-many LTF gates with unbounded fan-in.*

Note that AND, OR, and NOT are specific instances of LTFs, and therefore $\mathcal{TC}^0$ is complete. An equivalent definition for $\mathcal{TC}^0$ is as the class of constant-depth circuit families with polynomially-many majority gates with unbounded fan-in (for proof that the definitions are equivalent, see [GHR92; GK98]).

Definition 13 *(the class $\mathcal{NC}^1$). The class $\mathcal{NC}^1$ is the class of $O(\log n)$-depth circuit families consisting of fan-in two AND and OR gates and unary NOT gates.*

Many of our results hold for any "typical" circuit class with *constant depth* and gates of *unbounded fan-in*, where by "typical" we mean that no overly-restrictive structural constraints are imposed (e.g., the class is closed to parallel compositions and constant depth increase). We will state our results as applying to all typical constant-depth circuit classes, and to be formal we define this notion as follows:

Definition 14 *(typical constant-depth circuit classes). The* typical constant-depth circuit classes *are $\mathcal{CC}^0, \mathcal{AC}^0, \mathcal{ACC}^0, \mathcal{TC}^0$, and $\mathcal{CC}^0[q], \mathcal{AC}^0[q]$ for any $q \in \mathbb{N}$.*

We stress again that our results that are stated as applying to typical constant-depth circuit classes also apply to classes other than the ones in Definition 14. Finally, we recall that for any typical constant-depth class $\mathcal{C}$ it holds that $\mathcal{C} \neq \mathcal{NC}^1$ if and only if $\mathcal{C}$ does not contain uniform-$\mathcal{NC}^1$. For simplicity, we define uniform-$\mathcal{NC}^1$ using the notion of $\mathcal{P}$-uniformity; that is, a formula family is uniform if there exists a polynomial-time algorithm that on input 1^n outputs the n-bit formula in the family.

Proposition 15 *($\mathcal{C} = \mathcal{NC}^1$ iff uniform-$\mathcal{NC}^1 \subseteq \mathcal{C}$). Let $\mathcal{C}$ be any typical class of constant-depth circuits such that circuit families in $\mathcal{C}$ can be of arbitrarily large polynomial size. Then, $\mathcal{C} = \mathcal{NC}^1$ if and only if uniform-$\mathcal{NC}^1 \subseteq \mathcal{C}$.*

[15] When dealing with LTFs we can assume, without loss of generality, that $\langle w, x\rangle \neq \theta$ for every $x \in \{0,1\}^n$ (because for every Boolean function over $\{0,1\}^n$ that is computable by an LTF there exists an LTF that computes the function such that $\langle w, x\rangle \neq \theta$ for every $x \in \{0,1\}^n$).

Proof. We prove that if uniform-$\mathcal{NC}^1 \subseteq \mathcal{C}$ then $\mathcal{C} = \mathcal{NC}^1$ (the other direction is immediate). Loosely speaking, the proof relies on the fact that there exists a "universal" uniform-$\mathcal{NC}^1$ circuit family, i.e. a uniform family of $\mathcal{NC}^1$ formulas that can evaluate any $\mathcal{NC}^1$ formula that is given to them as input.

Specifically, consider the Boolean Formula Value problem, denoted BFV, in which the input is (Φ, z) such that Φ is a formula and z is an assignment to Φ, and the output is $\Phi(z)$.[16] As proved by Buss [Bus87], the problem BFV is in uniform-$\mathcal{NC}^1$. Thus, if $\mathcal{C}$ contains uniform-$\mathcal{NC}^1$, then there exists a family of $\mathcal{C}$-circuits for BFV, denoted $\{C_n\}_{n\in\mathbb{N}}$. Now, fix any $S \in \mathcal{NC}^1$, and let $\{\Phi_m : \{0,1\}^m \to \{0,1\}\}_{m\in\mathbb{N}}$ be a formula family of polynomial size that decides S. For every $m \in \mathbb{N}$, we "hard-wire" the description of Φ_m into the inputs of the circuit C_n, where $n = |\Phi_m| + m$ (and $|\Phi_m|$ is the length of the description of Φ_m), while leaving the m variables needed to describe the assignment z alive. After this "hard-wiring", we obtain a circuit family that decides S, and since $|\Phi_m| \leq \text{poly}(m)$, the size of C_n after the "hard-wiring" is still polynomial in its new input length m. ■

Proposition 15 was stated for uniform-$\mathcal{NC}^1$ where the notion of uniformity is $\mathcal{P}$-uniformity. However, the statement of Proposition 15 also holds if we consider much more restrictive notions of uniformity; indeed, the statement holds for any notion of uniformity such that BFV is in uniform-$\mathcal{NC}^1$ (for further details see [Bus87]).

3.2 Complete problems for $\mathcal{NC}^1$ and $\mathcal{NL}$

Let $\mathcal{C}$ be a class of functions $\{0,1\}^* \to \{0,1\}$, and let $\mathcal{F}$ be a class of functions $\{0,1\}^* \to \{0,1\}^*$. We say that a set $A \subseteq \{0,1\}^*$ is complete for $\mathcal{C}$ under $\mathcal{F}$-reductions if $A \in \mathcal{C}$, and for every set $B \in \mathcal{C}$ there exists a function $f \in \mathcal{F}$ such that such that for all $x \in \{0,1\}^*$ it holds that $x \in B$ if and only if $f(x) \in A$. When we just say that A is complete for $\mathcal{C}$, without mentioning $\mathcal{F}$, we mean that A is complete for $\mathcal{C}$ under $\mathcal{AC}^0$ reductions.[17] We now recall the definitions of several problems that are complete for $\mathcal{NC}^1$ under $\mathcal{AC}^0$ reductions or for $\mathcal{NL}$ under $\mathcal{NC}^1$ reductions.

3.2.1 The monoid problem

Recall that a monoid Σ is a set with an associative binary operation, which we think of as multiplication, and with an identity element $1 \in \Sigma$. Denoting $\ell = \lceil \log(|\Sigma|) \rceil$, fix an encoding $Bin : \Sigma \to \{0,1\}^\ell$. Intuitively, the monoid problem for Σ with encoding Bin is that of computing the multiplication of n input elements $\sigma_1, ..., \sigma_n \in \Sigma$. More formally, this is the problem of computing the function $f : \Sigma^* \to \Sigma$ such that for every $m \in \{\ell \cdot n : n \in \mathbb{N}\}$, when the input x can be parsed as $x = (Bin(\sigma_1), ..., Bin(\sigma_n))$ where $\sigma_i \in \Sigma$ for all $i \in [n]$, then the output is $f(x) = Bin(\prod_{i\in[n]} \sigma_i)$ (for inputs x that cannot be parsed in a valid manner, the output is $f(x) = 0^\ell$).

As proved by Barrington [Bar89], several natural problems that are complete for $\mathcal{NC}^1$ under $\mathcal{AC}^0$ reductions are instances of the monoid problem. (The formal presentation of the monoid problem that makes these problems complete is as a set of pairs $((\sigma_1, ..., \sigma_n), \sigma) \subseteq \{0,1\}^*$ such that $\prod_{i\in[n]} \sigma_i = \sigma$; yet, the complexity of the latter

[16] We denote this problem by BFV to distinguish it from the problem BFE, in which the formula Φ is guaranteed to be *balanced* (see Section 3.2.3).

[17] In some sources, the function f has to be *uniform*. In this paper we focus on lower bounds for non-uniform circuits, and therefore we do not require the reductions to be uniform.

version of the problem is essentially identical to that of the functional version, which we work with.) Since the problems remain complete under any injective encoding Bin, we suppress Bin in our notation.

The word problem over S_5 (W_{S_5}). In this problem the monoid Σ is the symmetric group S_5 (i.e., the group of all permutations on five elements). The binary operation on S_5 is the composition of the permutations, and its identity element is the identity permutation. In other words, the input is a sequence of permutations $\sigma_1, ..., \sigma_n \in S_5$, and the output is their composition $\sigma_1 \circ ... \circ \sigma_n$.

The s-t connectivity problem on directed graphs of width 5 (W5-STCONN). In this problem the monoid Σ is the set of of 5×5 matrices over $\{0,1\}$ (i.e., $\Sigma = \{0,1\}^{5\times 5}$). The binary operation is the matrix product of two matrices over the ring $(\{0,1\}, \mathsf{OR}, \mathsf{AND})$; that is, for any $A^{(1)}, A^{(2)} \in \Sigma$, the product $A^{(1)} \times A^{(2)}$ is defined such that $(A^{(1)} \times A^{(2)})_{i,j} = \vee_{k\in[5]}(A_{i,k} \wedge B_{k,j})$. The identity element is the identity matrix. [18]

3.2.2 The Boolean iterated matrix multiplication problem

Fix parameters $m \geq \Delta \in \mathbb{N}$. The Boolean Iterated Matrix Multiplication problem), denoted $\mathsf{BIMM}_{m,\Delta} : \{0,1\}^{m\cdot\Delta^2} \to \{0,1\}^{\Delta^2}$, is defined as follows. The input is parsed as m matrices, each of dimension $\Delta \times \Delta$ and with Boolean entries; and the output is the iterative multiplication of these matrices over the ring $(\{0,1\}, \mathsf{OR}, \mathsf{AND})$ (i.e., for two matrices $A^{(1)}, A^{(2)}$ we have that $(A^{(1)} \times A^{(2)})_{i,j} = \vee_{k\in[\Delta]}(A_{i,k} \wedge B_{k,j})$).

The main difference of BIMM from the W5-STCONN problem is that the size of the matrices in BIMM is a parameter $\Delta = \Delta(n)$ that may grow with the input size (instead of $\Delta = 5$ as in W5-STCONN). When $\Delta = n$ takes the maximal value, the problem $\mathsf{BIMM}_{n,n}$ is complete for $\mathcal{NL}$ under $\mathcal{NC}^1$ reductions (see [Coo85]).

3.2.3 The Boolean formula evaluation problem (BFE)

Intuitively, the Boolean Formula Evaluation Problem (BFE) is the problem of evaluating a given formula, which corresponds to a complete binary tree, at a given assignment. More formally, let $\Sigma = \{0, 1, \wedge, \vee, \oplus\}$. [19] For any $n \in \mathbb{N}$ of the form $n = 2^{\ell+1} - 1$, we think of an input $x \in \Sigma^n$ as consisting of two parts $x = (\Phi, z)$ such that $\Phi \in \Sigma^{2^\ell - 1}$ and $z \in \Sigma^{2^\ell}$. The first part Φ represents a formula of depth ℓ over $\{\wedge, \vee, \oplus\}$ (i.e., a complete binary tree of depth ℓ whose nodes are labeled by $\{\wedge, \vee, \oplus\}$). The second part $z \in \{0,1\}^\ell$ represents an assignment to Σ. When the input x can be parsed in this

[18] To see why this problem is called s-t connectivity on directed graphs of width 5, recall that a directed graph is of width 5 if its vertices can be partitioned into layers of (at most) five vertices each, such that the layers are linearly ordered, and every edge goes from vertices of one layer to the vertices of the next layer. In such a graph, every two consecutive layers form a bipartite graph with five vertices in each side, which can be represented by a 5×5 adjacency matrix. Thus, in the s-t connectivity problem we get n adjacency matrices (corresponding to a graph with $n+1$ layers), and output a matrix whose (i,j) entry is one if and only if there is a path from the i^{th} vertex in the first layer to the j^{th} vertex in the last layer.

[19] Similarly to [AK10], we define the problem over a basis that includes the $\oplus$ function in order to easily deduce a lower bound for it, by reducing from [IPS97; CSS16].

manner, the output of the problem is $\Phi(z)$, i.e. the evaluation of the formula represented by Φ at the assignment represented by z. When the input x cannot be parsed in this manner, the output of the problem is 0.

At this point we need to be more specific about the parsing of $\Phi \in \Sigma^{2^\ell - 1}$ to a formula. To define the problem we can use any fixed bijection φ between the nodes in the depth-ℓ complete binary tree and $[2^\ell - 1]$ (e.g., we map the root to 1, and for any node v mapped to a number $id(v)$, the left and right children of v are mapped to $2 \cdot id(v)$ and $2 \cdot id(v) + 1$, correspondingly), and letting the label of a node v in the formula be $\Phi_{\varphi(v)}$. The crucial point is that the mapping of nodes to symbols is *fixed*, which means that if we want to know the labels of a sub-formula of Φ, we only need to look at *fixed locations* in the input (rather than parse the input in order to know which locations are needed). [20] For proof that this formulation of the BFE problem is $\mathcal{NC}^1$-complete under $\mathcal{AC}^0$ reductions, see [Bus87] and [BIS90, Proof of Lemma 7.2].

3.3 Seeded extractors and averaging samplers

We recall the standard definitions of seeded extractors and of averaging samplers, and state the well-known equivalence between the two.

Definition 16 *(seeded extractors). A function $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is a (k, ϵ)-*extractor *if for every distribution $\mathbf{x}$ on $\{0,1\}^n$ such that $\max_{x \in \{0,1\}^n} [\Pr[\mathbf{x} = x]] \leq 2^{-k}$ it holds that the distribution $Ext(\mathbf{x}, \mathbf{u}_t)$ is ϵ-close to the uniform distribution on $\{0,1\}^m$ in statistical distance.*

Definition 17 *(averaging samplers). A function $Samp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ is an* averaging sampler with accuracy $\epsilon > 0$ and error $\delta > 0$ *if it satisfies the following. For every $T \subseteq \{0,1\}^m$, for all but a δ-fraction of the strings $x \in \{0,1\}^n$ it holds that $\Pr_{z \in \{0,1\}^t}[Samp(x, z) \in T] = |T|/2^m \pm \epsilon$.*

Proposition 18 *(seeded extractors are equivalent to averaging samplers). Let $f : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$. Then, the following two assertions hold:*

1. *If f is a (k, ϵ)-extractor, then f is an averaging sampler with accuracy ϵ and error $\delta = 2^{k-n}$.*
2. *If f is an averaging sampler with accuracy ϵ and error δ, then f is an $(n - \log(\epsilon/\delta), 2\epsilon)$-extractor.*

For a proof of Proposition 18 see, e.g., [Vad12, Cor. 6.24]. In the current paper we will only use the first item of Proposition 18.

4 Improved hardness magnification results

In this section we present our improved hardness magnification results. The results in the current section are batched according to the technical tools used in their proofs, rather than the circuit classes to which they apply:

[20]This is in contrast to an encoding of Φ that is an arbitrary list of gates and wires, which makes the problem complete for $\mathcal{L}$; see [Ete97].

- In Section 4.1 we prove hardness magnification results for any instance of the monoid problem, and in particular for W_{S_5}, W5-STCONN, and AND; that is, we prove Theorems 2 and 7. The proofs of these results follow the high-level description in Section 2.1.

- In Section 4.2 we prove hardness magnification results for MAJ and for BIMM; that is, we prove the version of Theorem 2 that holds for $\mathsf{BIMM}_{n,n^{o(1)}}$, and also prove Theorem 6. The proofs of these results follow by analyzing problems that are similar to the monoid problem, but in which the size of the monoid may grow with the input length.

- Finally, in Section 4.3 we prove our hardness magnification result for BFE; that is, we prove the version of Theorem 2 that holds for BFE. The main idea in the proof of this result is similar to the idea presented in Section 2.1, but it requires an implementation that is specific to the BFE problem.

4.1 Monoid problems: W_{S_5}, W5-STCONN, and AND

Our main theorem in this section is a strong self-reduction for the monoid problem: For $d \gg d_0$, we reduce the task of computing the problem in depth d and size $n^{1+\exp(-d)}$ to the task of computing the problem in depth d_0 and size n^k.

We will first consider the promise problem version of the monoid problem: For a fixed monoid Σ and encoding $Bin : \Sigma \to \{0,1\}^\ell$ (where $\ell = \lceil \log(|\Sigma|) \rceil$), in this problem the input is guaranteed to be of the form $x = (Bin(\sigma_1), ..., Bin(\sigma_m))$, and the required output is $f(x) = Bin(\prod_{i \in [m]} \sigma_i)$.

Theorem 19 *(strong self-reduction for the monoid problem). Fix a monoid Σ and an encoding $Bin : \Sigma \to \{0,1\}^\ell$, where $\ell = \lceil \log(|\Sigma|) \rceil$. Let $\mathcal{C}$ be a typical constant-depth circuit class, and assume that for two integers $d_0, k \geq 2$, the promise-problem version of the monoid problem for Σ can be solved by a family of $\mathcal{C}$-circuits of depth d_0 and size n^k. Then, for infinitely many constants d, the promise-problem version of the monoid problem for Σ can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^2 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

Proof. For $n \in \ell \cdot \mathbb{N}$, we say that $x \in \{0,1\}^n$ is valid if $x = (Bin(\sigma_1), ..., Bin(\sigma_{n/\ell}))$. Let $\{f_n : \{0,1\}^n \to \{0,1\}^\ell\}_{n \in \mathbb{N}}$ be the Boolean function corresponding to the monoid problem for Σ with encoding Bin, and let $C = \{C_n\}_{n \in \mathbb{N}}$ be a circuit family such that for every $n \in \mathbb{N}$ it holds that C_n is a circuit $\{0,1\}^n \to \{0,1\}^\ell$ of size n^k and depth d_0, and for any valid input $x \in \{0,1\}^n$ it holds that $C_n(x) = f_n(x)$. In the proof we will "round" numbers to the nearest multiple of $\ell = \lceil \log(|\Sigma|) \rceil = O(1)$; for convenience, we denote the corresponding operations by $\lceil x \rceil_\ell = \ell \cdot \lceil x/\ell \rceil$ and $\lfloor x \rfloor_\ell = \ell \cdot \lfloor x/\ell \rfloor$.

For any $\epsilon > 0$, let $d_1 = \left\lceil k \cdot \ln\left(\frac{(1-1/k)\cdot(1+\epsilon)}{\epsilon}\right)\right\rceil + 1$. For any sufficiently large $n \in \ell \cdot \mathbb{N}$, we construct a circuit for f_n of depth $d = d_1 \cdot d_0$ whose number of wires is $O(n^{1+\epsilon}) < n^{1+2e^2 \cdot e^{-d/(k \cdot d_0)}}$.

Our circuit consists of d_1 layers, where layer $i = 1$ is the top layer, and layer $i = d_1$ is the bottom layer, which is connected to the inputs. For a sequence $0 = \alpha_1 < \alpha_2 < ... < \alpha_{d_1} < 1$ that we will define in a moment, each layer $i \in [d_1]$ is of depth d_0, and contains $n_i = \lceil n^{\alpha_i} \rceil$ copies of circuits from C, which get their inputs from disjoint blocks of bits in the layer beneath it. That is, the bottom layer d_1 contains $n_{d_1} = \lceil n^{\alpha_{d_1}} \rceil$ copies of circuits from C on $\lceil n/n_{d_1} \rceil_\ell$ or $\lfloor n/n_{d_1} \rfloor_\ell$ input bits (i.e., it contains $C_{\lceil n/n_{d_1} \rceil_\ell}$

and $C_{\lfloor n/n_{d_1} \rfloor_\ell}$), each of which computes $f_{\lceil n/n_{d_1} \rceil_\ell}$ or $f_{\lfloor n/n_{d_1} \rfloor_\ell}$ on a corresponding block of input bits. Then, for $i \in [d_1 - 1]$, the outputs of layer $i+1$ serve as inputs to layer i, which contains $n_i = \lceil n^{\alpha_i} \rceil$ copies of $C_{\lceil n_{i+1}/n_i \rceil_\ell}$ and $C_{\lfloor n_{i+1}/n_i \rfloor_\ell}$, each of which gets its input from a corresponding block. Since $\alpha_1 = 0$, the top layer $i = 1$ contains a single circuit from C.

Let us now specify the α_i's, from top to bottom: The top layer $i = 1$ has a single circuit, so $\alpha_1 = 0$; and for every $i = 2, ..., d_1$ we set $\alpha_i = \frac{1+\epsilon}{k} + (1 - 1/k) \cdot \alpha_{i-1}$. For $i \geq 2$, this recursion solves to

$$\alpha_i = \frac{1+\epsilon}{k} \cdot \sum_{j=0}^{i-2} (1 - 1/k)^j = (1+\epsilon) \cdot \left(1 - (1 - 1/k)^{i-1}\right) ,$$

and in particular in layer d_1 we have that $\alpha_{d_1} \geq 1 - \epsilon/(k-1)$.

The foregoing circuit is of depth $d_1 \cdot d_0$, and indeed solves the promise-problem version of the monoid problem for Σ (this is because each layer is connected to all outputs of the layer beneath it, we only use circuits from the family C, and by the associativity of the monoid). Thus, it remains to verify that the circuit has $O(n^{1+\epsilon})$ wires. To do so, note that for $i \in [d_1 - 1]$, in the i^{th} layer we have $\lceil n^{\alpha_i} \rceil$ circuits, and the number of inputs bits of each of them is at most $\lceil n_{i+1}/n_i \rceil_\ell < 2 \cdot n^{\alpha_{i+1} - \alpha_i}$. Thus, the total number of wires in the i^{th} layer is $O\left(n^{\alpha_i + k \cdot (\alpha_{i+1} - \alpha_i)}\right) = O(n^{1+\epsilon})$. In layer $i = d_1$, each circuit has $\lceil n/n_{d_1} \rceil_\ell < 2 \cdot n^{1-\alpha_{d_1}}$ input bits, and thus the total number of wires in layer d is $O\left(n^{\alpha_{d_1} + k \cdot (1 - \alpha_{d_1})}\right) \leq O\left(n^{k - (k-1) \cdot \alpha_{d_1}}\right) \leq O(n^{1+\epsilon})$. ■

The conclusion of Theorem 19 asserts that the monoid problem can be solved by depth-d circuits for infinitely-many $d \in \mathbb{N}$. Now we want to extend this conclusion to hold for almost all depths d. To do so, we note that the gap between the applicable depths in the proof of Theorem 19 is approximately d_0. Therefore, to get a circuit of an arbitrary depth d, we can use a circuit of depth d' such that $d - d_0 < d' < d$, and the bound on the number of wires will be $n^{1+\exp(-d')} < n^{1+\exp(-(d-d_0))}$. Specifically:

Corollary 20 *(a self-reduction for the monoid problem for any depth). Let Σ be a monoid, $\mathcal{C}$ be a circuit class, and $d_0, k \geq 2$ be integers that satisfy the hypothesis of Theorem 19. Then, for every sufficiently large constant d, the promise-problem version of the monoid problem for Σ can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

Proof. Let $d \in \mathbb{N}$ be sufficiently large. In the proof of Theorem 19, for every $\epsilon > 0$, we constructed a circuit for the monoid problem of depth $d_\epsilon = \left(\left\lceil k \cdot \ln\left(\frac{(1-1/k)\cdot(1+\epsilon)}{\epsilon}\right)\right\rceil + 1\right) \cdot d_0$ whose number of wires is at most $O(n^{1+\epsilon}) < n^{1+2e^2 \cdot e^{-d_\epsilon/(k \cdot d_0)}}$. Now, for $\epsilon > 0$ such that $d - d_0 \leq d_\epsilon < d$,[21] we think of the construction of depth d_ϵ as a construction of depth d whose number of wires is at most

$$n^{1+2e^2 \cdot e^{-d_\epsilon/(k \cdot d_0)}} \leq n^{1+2e^2 \cdot e^{-(d-d_0)/(k \cdot d_0)}} = n^{1+2e^{2+1/k} \cdot e^{-d/(k \cdot d_0)}} ,$$

where the inequality relied on the fact that $d_\epsilon \geq d - d_0$. ■

[21]Specifically, we choose $\epsilon > 0$ such that $\frac{\epsilon}{1+\epsilon} = 1 - \frac{1}{1+\epsilon} = (1 - 1/k) \cdot e^{-(d/d_0 - 2)/k}$ (i.e., $\epsilon = \frac{1}{1 - (1-1/k) \cdot e^{-(d/d_0-2)/k}} - 1 = \frac{e^{(d/d_0-2)/k}}{e^{(d/d_0-2)/k} - (1-1/k)} - 1 > 0$, where the inequality is by the hypothesis that d is sufficiently large). It follows that $\left\lceil k \cdot \ln\left(\frac{(1-1/k)\cdot(1+\epsilon)}{\epsilon}\right)\right\rceil + 1 = \lceil d/d_0 \rceil - 1$, and hence $d - d_0 \leq d_\epsilon < d$.

We now instantiate Corollary 20 for specific monoid problems and circuit classes: Assuming that some monoid problem can be solved by polynomial-sized circuits from some class $\mathcal{C}$, we deduce that the problem can be solved by extremely sparse $\mathcal{C}$-circuits. The main "gap" that needs to be handled is that Corollary 20 only deduces that the promise problem can be solved by extremely sparse circuits; we will need the circuits to also verify that the input is a valid encoding of elements from Σ, and output 0^ℓ otherwise. This is easy when the circuit class is complete; moreover, in $\mathcal{TC}^0$ this can be done with essentially no overhead:

Corollary 21 *(W_{S_5} and W5-STCONN in $\mathcal{TC}^0$). Let Π be either the W_{S_5} problem or the W5-STCONN problem. Assume that Π can be computed by $\mathcal{TC}^0$ circuits of depth d_0 and size n^k. Then, for every sufficiently large constant $d \in \mathbb{N}$ it holds that Π can be computed by $\mathcal{TC}^0$ circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

Note that the conclusion of Corollary 21 implies, in particular, that for any $c' < e^{1/(k \cdot d_0)}$, for every sufficiently large $d \in \mathbb{N}$ it holds that W_{S_5} can be computed by $\mathcal{TC}^0$ circuits of depth d and size $n^{1+(c')^{-d}}$ (since $(c')^{-d} > 2e^3 \cdot c^{-d}$).

Proof of Corollary 21. Our hypothesis now is stronger than the hypothesis required for instantiating Corollary 20, since we assume that the monoid problem *without a promise* can be solved in $\mathcal{TC}^0$ (with some depth d_0 and size n^k). Thus, as our starting point we can use the construction of the circuit from the proof of Theorem 19.

We add to this construction another circuit V, in parallel to the original circuit, that is a conjunction of n/ℓ conditions, each condition verifying that the i^{th} block of ℓ bits in the input indeed represents an element from Σ. Then, we modify each of the ℓ output gates of the original circuit such that it conjuncts with the output of V. [22] Hence, if the input is not a valid representation of elements from Σ, the output is 0^ℓ. Note that the number of wires in V and between V and the outputs is $O_\ell(n)$; thus, the number of wires in the entire construction is still $O(n^{1+\epsilon})$. ■

Similarly, when the circuit class is $\mathcal{ACC}^0$ it is easy to verify that an input is a valid encoding of elements from Σ. We thus get the following corollary:

Corollary 22 *(W_{S_5} and W5-STCONN in $\mathcal{ACC}^0$). Let Π be either the W_{S_5} problem or the W5-STCONN problem. Assume that Π can be computed by $\mathcal{ACC}^0$ circuits of depth d_0 and size n^k. Then, for every sufficiently large constant $d \in \mathbb{N}$ it holds that Π can be computed by $\mathcal{ACC}^0$ circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

Proof. Similarly to the proof of Corollary 21, we start from the circuit C of depth $d' = d - 1$ from the proof of Corollary 20 (i.e., the construction from the proof of Theorem 19), and add to it a linear-sized circuit, denoted V, that verifies that the input is a valid encoding of elements from the monoid. Since now we cannot modify the output gates to conjunct with V (since the output gates may be modular gates), we add a top layer of $\ell = |\Sigma$ gates to the construction, such that each top gate computes the AND of V and of a corresponding output gate of C. The final circuit is of depth $d = d' + 1$ and size $n^{1+2e^{2+1/k} \cdot c^{-(d-1)}} \le n^{1+2e^3 \cdot c^{-d}}$. ■

[22] That is, assume that an output gate g is an LTF that accepts if and only if the weighted sum of its inputs is at least $\theta \in \mathbb{R}$. Denote by M the sum of absolute values of the weights of g. Then, we replace g by g' that assigns weight $(M+1)^2$ to V (and the same weights as in g to the original inputs), and accepts if and only if the weighted sum of its inputs is at least $(M+1)^2 + \theta$.

Finally, turning to $\mathcal{CC}^0$ and to the AND function, note that when $|\Sigma|$ is a power of two, the promise is trivial (since any bit-string of length $\log(|\Sigma|)$ is a valid encoding of an element from Σ). Thus, we get the following corollary:

Corollary 23 *(AND and $\mathcal{CC}^0$). If AND can be computed by $\mathcal{CC}^0$ circuits of depth d_0 and size n^k, then for every sufficiently large constant $d \in \mathbb{N}$ it holds that AND can be computed by $\mathcal{CC}^0$ circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

As explained after the statement of Corollary 21, the conclusions of Corollaries 22 and 23 imply that for every sufficiently large $d \in \mathbb{N}$, the corresponding function can be computed by depth-d circuits of size $n^{1+(c')^{-d}}$, where $c' < e^{1/(k \cdot d_0)}$.

4.2 Growing monoids: MAJ and Boolean matrix multiplication

In this section we prove our results regarding computing MAJ in $\mathcal{ACC}^0$ (i.e., Theorem 6), and computing the Boolean iterated matrix multiplication problem in constant-depth circuit classes.

As a starting point we want to extend the self-reduction of the monoid problem (i.e., Theorem 19) to monoids of size that is growing with the input size. That is, fixing an associative length-preserving binary operation $\{\times : (\{0,1\}^r)^2 \to \{0,1\}^r\}_{r \in \mathbb{N}}$, for every two integers $m \geq \ell \in \mathbb{N}$, the corresponding monoid problem $\mathsf{MON}^{\times}_{m,\ell}$ is defined on input length $n = m \cdot \ell$ as follows: An input $x \in \{0,1\}^n$ is parsed as $(\sigma_1, ..., \sigma_m)$ where $\sigma_i \in \{0,1\}^\ell$ for all $i \in [m]$, and the ℓ-bit output is $f(x) = \prod_{i \in [m]} \sigma_i$.

We show a strong self-reduction for the foregoing problem: For $d \gg d_0$, we reduce the task of computing $\mathsf{MON}^{\times}_{m,\ell}$ in depth d and size $(m \cdot \ell)^{1+\exp(-d)}$ to the task of computing $\mathsf{MON}^{\times}_{m',\ell}$, for various values of $m' = m^{\Omega(1)}$, in depth d_0 and size $(m' \cdot \ell)^k$.

Theorem 24 *(strong self-reduction for the monoid problem with monoids of growing size). Let $\mathcal{C}$ be a typical constant-depth circuit class, and let $\{\times : (\{0,1\}^r)^2 \to \{0,1\}^r\}_{r \in \mathbb{N}}$ be an associative operation. Let $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(m) = m^{o(1)}$. Assume that there exist constants $d_0, k \in \mathbb{N}$ such that for every constant $\rho > 0$ and any sufficiently large $m \in \mathbb{N}$ it holds that $MON^{\times}_{m^\rho,\ell(m)}$ can be be solved by a family of $\mathcal{C}$-circuits of depth d_0 and size n^k, where $n = m^\rho \cdot \ell(m)$. Then, for every sufficiently large constant $d \in \mathbb{N}$ it holds that $MON^{\times}_{m,\ell(m)}$ can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$ and $n = m \cdot \ell(m)$.*

In high-level, the proof of Theorem 24 is as follows. In order to compute $\mathsf{MON}^{\times}_{m,\ell}$, we will use the tree construction from the proof of Theorem 19 to reduce the problem to the problems of computing $\mathsf{MON}^{\times}_{m^\epsilon,\ell}$ for various small constants $\epsilon > 0$. Note that in the problems $\mathsf{MON}^{\times}_{m^\epsilon,\ell}$ in the target of the reductions, the input length is smaller than the original input length (i.e., the input length is m^ϵ instead of m) but the monoid is still of its same original size $\ell = \ell(m)$. Now, by our hypothesis, there exist fixed $d_0, k \in \mathbb{N}$ such that for every input length m^ϵ, the problem $\mathsf{MON}^{\times}_{m^\epsilon,\ell}$ is solvable by circuits of depth d_0 and size $(m^\epsilon \cdot \ell)^k$. The crucial point in the proof is that $\ell(m) = m^{o(1)}$, and therefore the size of the latter circuits is less than $(2m^\epsilon)^k$; thus, the bound on the overall size of the circuit for $\mathsf{MON}^{\times}_{m,\ell}$ remains essentially the same as in the proof of Theorem 19. We defer the full details to Appendix A.

Recall that the Boolean Iterated Matrix Multiplication problem (BIMM) can be thought of as an instance of the monoid problem with monoids of growing size (the associative operation here is matrix multiplication). Therefore, the version of Theorem 2 for $\mathsf{BIMM}_{n,n^{o(1)}}$ follows as a corollary of Theorem 24. Also, we can now prove the following theorem regarding MAJ in $\mathcal{ACC}^0$, whose contrapositive is Theorem 6:

Theorem 25 *($\mathsf{MAJ} \in \mathcal{ACC}^0$ iff MAJ is in extremely sparse $\mathcal{ACC}^0$). If $\mathsf{MAJ} \in \mathcal{ACC}^0$, then there exists a constant $c > 1$ such that for every sufficiently large constant $d \in \mathbb{N}$ it holds that MAJ can be solved by an $\mathcal{ACC}^0$ circuit family with depth d and $n^{1+c^{-d}}$ wires.*

Proof. We first define an instance of the monoid problem with monoids of growing size, denoted $\mathsf{SUM}_{m,\ell}$, where the associative operation is summation of m integers, represented by in binary by ℓ bits, modulo 2^ℓ. Formally, for any $m \in \mathbb{N}$ and $\ell \in [m]$, let $\mathsf{SUM}_{m,\ell} : \{0,1\}^{m\cdot\ell} \to \{0,1\}^\ell$ be the following function: Interpreting any input $x \in \{0,1\}^{m\cdot\ell}$ as a list $a^{(1)},...,a^{(m)}$, where each $a^{(i)} \in \{0,1\}^\ell$ is the binary representation of an integer $\sigma^{(i)} \in \{0,...,2^\ell - 1\}$, we define $\mathsf{SUM}_{m,\ell}(x)$ to be the binary representation of $\sum_{i\in[m]} \sigma^{(i)} \bmod 2^\ell$. It is well-known that $\mathsf{SUM}_{m,m} \in \mathcal{TC}^0$; in fact:

Fact 25.1. *There exist $k, d_0 \in \mathbb{N}$ such that for every sufficiently large $m \in \mathbb{N}$ and every $\ell \in [m]$ it holds that $\mathsf{SUM}_{m,\ell}$ can be computed by a $\mathcal{TC}^0$ circuit of size m^k and depth d_0.*

Proof. As shown by Chandra, Stockmeyer and Vishkin [CSV84], there exist $k, d_0 \in \mathbb{N}$ such that $\mathsf{SUM}_{m,m}$ can be computed in $\mathcal{TC}^0$ of size m^k and depth d_0. [23] We reduce $\mathsf{SUM}_{m,\ell}$ to $\mathsf{SUM}_{m,m}$ by padding each ℓ-bit integer to be of m bits and by truncating the output to ℓ bits, while noting that this reduction does not increase the value of m. $\square$

Now, assume that $\mathsf{MAJ} \in \mathcal{ACC}^0$, which implies that $\mathcal{ACC}^0 = \mathcal{TC}^0$. By Fact 25.1, there exist $k, d_0 \in \mathbb{N}$ such that for every sufficiently large $m \in \mathbb{N}$ and every $\ell \in [m]$ it holds that $\mathsf{SUM}_{m,\ell}$ can be computed by an $\mathcal{ACC}^0$ circuit of size $m^k < (m \cdot \ell)^k$ and depth d_0. In particular, for every $\rho > 0$ and sufficiently large $m \in \mathbb{N}$ and $\ell = 2 \cdot \log(m)$, we have that $\mathsf{SUM}_{m^\rho,\ell}$ can be computed by an $\mathcal{ACC}^0$ circuit of size $(m^\rho \cdot \ell)^k$ and depth d_0. Therefore, by Theorem 24 it holds that for some $c_0 > 1$ (i.e., any $1 < c_0 < e^{1/(k\cdot d_0)}$ suffices) and for infinitely many $d \in \mathbb{N}$, the problem $\mathsf{SUM}_{m,2\log m}$ can be solved by a depth-d $\mathcal{ACC}^0$ circuit with $(2 \cdot m \cdot \log(m))^{1+c_0^{-d}}$ wires.

Finally, we reduce MAJ on n input bits to $\mathsf{SUM}_{n,2\cdot\log(n)}$ as follows. We first pad each input bit $\sigma \in \{0,1\}$ to a binary representation of an integer $x_\sigma = 0...0\sigma \in \{0,1\}^{2\cdot\log(m)}$. Then, we compute $\mathsf{SUM}_{n,2\cdot\log(n)}$ using a circuit with $(2 \cdot n \cdot \log(n))^{1+c_0^{-d}} = n^{1+\exp(-d)}$ wires. Lastly, we output the OR of the $\log(n) + 1$ most important bits of $\mathsf{SUM}_{n,2\cdot\log(n)}$, which only adds one more layer to the depth and $\log(n) + 1$ wires. $\blacksquare$

4.3 The BFE function

We now present a strong self-reduction for the BFE function. Recall that in this function the input is $x = (\Phi, z)$, where $\Phi \in \Sigma^{2^\ell - 1}$ is a description of a formula over the basis $\Sigma = \{0, 1, \wedge, \vee, \oplus\}$ that is a complete binary tree of depth ℓ and $z \in \{0,1\}^{2^\ell}$ is an assignment to the formula, and the output $\mathsf{BFE}(x)$ is the evaluation of Φ at z.

Let us present the proof idea, in high-level. Our goal is to evaluate a given formula Φ at a given assignment z by a circuit of size $n^{1+\epsilon}$ and depth $O(\log(1/\epsilon))$, under the assumption that there exists a circuit for BFE over m input symbols of size m^k.

[23]They actually showed that iterated addition of m integers, each represented in binary by m bits, is in $\mathcal{TC}^0$; the statement that $\mathsf{SUM}_{m,m} \in \mathcal{TC}^0$ follows by truncating the result to the last m bits.

Following [AK10], we will construct a circuit whose structure imitates the formula structure: Our circuit will consist of $O(\log(1/\epsilon))$ layers such that in layer i there will be n_i circuits, each computing a sub-formula of Φ that takes its inputs from sub-formulas in layer $i+1$. We abstract this construction as a tree. The top node represents a sub-circuit that computes the top sub-formula in Φ that has $n^{(1+\epsilon)/k}$ inputs (i.e., the top sub-formula in Φ of depth $\frac{1+\epsilon}{k} \cdot \log(n)$). Denoting the latter formula by Φ', note that in contrast to the construction in Theorem 19, the circuit corresponding to the top node now depends not only on the outputs of the $n^{(1+\epsilon)/k}$ nodes in the layer beneath it, but also on the symbols in the input that represent the labels of Φ'. Nevertheless, since Φ' is a *formula*, the number of inputs to Φ' is essentially identical to the number of labels in Φ'. Thus, the input size to the circuit corresponding to the top node is still $O(n^{(1+\epsilon)/k})$, and thus the circuit size is $O(n^{1+\epsilon})$. Continuing on, and using the same tree structure as in the proof of Theorem 19, at each level $i \geq 2$ we need to compute n_i sub-formulas that will serve as inputs to level $i-1$, and we compute each of these sub-formulas by a circuit for BFE that has $O((n^{(1+\epsilon)}/n_i)^{1/k})$ inputs. We have that $n_i = n^{\alpha_i}$, where $\alpha_i = (1+\epsilon) \cdot \left(1 - (1-1/k)^{i-1}\right)$, and thus at level $d \approx k \cdot \ln(1/\epsilon)$ we can finish the construction.

We now present the formal statements and proofs. Similarly to Section 4.1, we first consider the promise problem version of BFE: In this promise problem, every input $x \in \{0,1\}^{4 \cdot 2^{\ell}-3}$ is guaranteed to be of the form $x = (\Phi, z)$, where $\Phi \in \Sigma^{2^{\ell}-1} \subseteq \{0,1\}^{3 \cdot (2^{\ell}-1)}$ is an encoding of a formula and $z \in \{0,1\}^{2^{\ell}}$ represents an assignment to the formula.

Theorem 26 *(strong self-reduction for BFE). Let $\mathcal{C}$ be a typical constant-depth circuit class, and assume that for two constants $d_0, k \in \mathbb{N}$, the promise problem version of BFE can be solved by a family of $\mathcal{C}$-circuits of depth d_0 and size n^k. Then, for any sufficiently large constant $d \in \mathbb{N}$, the promise problem version of BFE problem can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k \cdot d_0)}$.*

Proof. For any $m \in \{4 \cdot 2^{\ell} - 3 : \ell \in \mathbb{N}\}$, let $C_m : \{0,1\}^m \to \{0,1\}$ be a circuit of size m^k and depth d_0 that solves the promise problem version of BFE on inputs of length m. For any $\epsilon > 0$, let $d_1 = \left\lceil k \cdot \ln\left(\frac{(1-1/k)\cdot(1+\epsilon)}{\epsilon}\right)\right\rceil + 1$. We will construct a circuit $\{0,1\}^n \to \{0,1\}$ for the promise problem version of BFE on inputs of length n with depth $d = d_1 \cdot d_0$ and with $O\left(n^{1+\epsilon}\right) < n^{1+2e^2 \cdot e^{-d/(k \cdot d_0)}}$ wires.

We think of an input $x \in \{0,1\}^n$, where $n = 4 \cdot r - 3$ and $r = 2^{\ell}$, as $x = (\Phi, z)$, where $\Phi \in \{0,1\}^{3 \cdot (r-1)}$ represents the formula and $z \in \{0,1\}^r$ represents the assignment. For $\ell_i \in \{0, ..., \ell - 1\}$, denote by Φ_{ℓ_i} the set of sub-formulas of Φ of distance ℓ_i from the top node in Φ. Also denote by Φ_{ℓ} the input gates of the formula (at distance ℓ from the top node). Our goal is to construct a circuit that outputs $\Phi(z)$.

As in the proof of Theorem 19, define a sequence $0 = \alpha_1 < \alpha_2 < ... < \alpha_{d_1} < 1$ such that for $i \in [d_1 - 1]$ it holds that $\alpha_{i+1} = \frac{1+\epsilon}{k} + (1-1/k) \cdot \alpha_i = (1+\epsilon) \cdot \left(1 - (1-1/k)^i\right)$. Our circuit consists of d_1 layers, where layer $i = 1$ is the top layer, and layer $i = d_1$ is the bottom layer above the input x. Each layer $i \in [d_1]$ computes the evaluations of the sub-formulas in Φ_{ℓ_i} at the assignment z, where $\ell_i = \lceil \alpha_i \cdot \ell \rceil$; note that there are $r_i = 2^{\ell_i} < 2 \cdot r^{\alpha_i}$ such sub-formulas. To do so, we use r_i copies of C_{n_i}, where $n_i = 4 \cdot r_{i+1}/r_i - 3$: Each of the r_i copies gets as input the evaluations at z of r_{i+1}/r_i sub-formulas in $\Phi_{\ell_{i+1}}$, which are computed by layer $i+1$; as well as the descriptions of the $r_{i+1}/r_i - 1$ symbols in Φ (each of which is represented by three bits) that are needed to compute the corresponding sub-formula.

Since each layer $i \in [d_1]$ computes the evaluations of the sub-formulas in Φ_{ℓ_i} at z, and we have that $\ell_1 = 0$, the output of the circuit is $\Phi(z)$. Now, in each layer $i = \{0, ..., d_1 - 1\}$ there are at most $2 \cdot r^{\alpha_i} < 2 \cdot n^{\alpha_i}$ copies of C_{n_i}, where $n_i = 4 \cdot r_{i+1}/r_i - 3 < 8 \cdot r^{\alpha_{i+1} - \alpha_i} < 8 \cdot n^{\alpha_{i+1}-\alpha_i}$. Thus, the total number of wires in each layer $i = \{0, ..., d_1 - 1\}$ is at most $2 \cdot n^{\alpha_i} \cdot (8 \cdot n^{\alpha_{i+1}-\alpha_i})^k = O\big(n^{1+\epsilon}\big)$. In layer d_1, we have $2 \cdot r^{\alpha_{d_1}} < 2 \cdot n^{\alpha_{d_1}}$ copies of $C_{n_{d_1}}$ where $n_{d_1} = 4 \cdot r/r_{d_1} < 4 \cdot n^{1-\alpha_{d_1}}$. Since $\alpha_{d_1} \geq 1 - \epsilon/(k-1)$, the total number of wires in layer d_1 is less than $8 \cdot n^{\alpha_{d_1} + k\cdot(1-\alpha_{d_1})} = O\left(n^{k-(k-1)\cdot\alpha_{d_1}}\right) = O\left(n^{1+\epsilon}\right)$.

Finally, the construction above yields a circuit of depth $d = d_1 \cdot d_0$ for every $\epsilon > 0$, and we can extend this construction to values of d that are not multiples of d_0 just as in the proof of Corollary 20, at the cost of deteriorating the size bound from $n^{1+2e^2 \cdot e^{-d/(k\cdot d_0)}}$ to $n^{1+2e^{2+1/k} \cdot e^{-d/(k\cdot d_0)}} < n^{1+2e^3 \cdot e^{-d/(k\cdot d_0)}}$. ■

As in Section 4.1, to instantiate Theorem 26 for the non-promise version of BFE, we just need the constructed circuit to also verify that the input is a valid encoding of a Boolean formula and an assignment. This is indeed easy in $\mathcal{TC}^0$:

Corollary 27 *(BFE and $\mathcal{TC}^0$). If BFE can be computed by $\mathcal{TC}^0$ circuits of depth d_0 and size n^k, then for every sufficiently large constant $d \in \mathbb{N}$ it holds that BFE can be computed by $\mathcal{TC}^0$ circuits of depth d and size $n^{1+2e^3 \cdot c^{-d}}$, where $c = e^{1/(k\cdot d_0)}$.*

Proof. Let $\Sigma = \{0, 1, \mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$. Given an input $x \in \{0,1\}^n$, where $n = 3 \cdot (2^{\ell+1} - 1)$, we think of the input as $x = (\Phi, z)$ where $\Phi \in \{0,1\}^{3\cdot(2^\ell - 1)}$ and $z \in \{0,1\}^{3\cdot 2^\ell}$. We will use the circuit C from the proof of Theorem 26, augmented by two auxiliary circuits: One circuit V will verify that Φ and z are valid encodings (i.e., Φ encodes symbols from Σ and z encodes symbols from $\{0,1\} \subseteq \Sigma$), and another circuit T will translate z into a $\{0,1\}$-assignment that C can use. Specifically, we invoke the circuit C on Φ and on $T(z)$, and conjunct the output of C with V.

The circuits V and T can be implemented using $O(n)$ wires. Also, the addition of T between the inputs layer and C creates an overhead of a single layer in the depth (i.e., if each translation of an element in Σ into a bit is implemented by a CNF, the top AND gate of the CNF can be merged into the LTF above it in C). Therefore, the size of the resulting circuit is $O\left(n^{1+2e^{2+1/k} \cdot c^{-(d-1)}}\right) \leq n^{1+2e^3 \cdot c^{-d}}$. ■

Similarly, when the circuit class is $\mathcal{ACC}^0$ it is easy to verify that an input is a valid encoding of a Boolean formula and of an assignment; thus:

Corollary 28 *(BFE and $\mathcal{ACC}^0$). If BFE can be computed by $\mathcal{ACC}^0$ circuits of depth d_0 and size n^k, then for every sufficiently large constant d it holds that BFE can be computed by $\mathcal{ACC}^0$ circuits of depth d and size $n^{1+2e^4 \cdot c^{-d}}$, where $c = e^{1/(k\cdot d_0)}$.*

Proof. Similarly to the proof of Corollary 27, we start from the circuit C of depth $d' = d - 3$ from the proof of Theorem 26, and add to C a circuit T that translates z into a $\{0,1\}$-assignment and a circuit V that verifies that Φ and z are valid encodings, where both T and V have $O(n)$ wires. Adding a top gate that conjuncts the output of C with the output of V, and a bottom layer of CNFs that implement T, we obtain a circuit of depth $d = d' + 3$ and of size $n^{1+2e^{2+1/k} \cdot c^{-(d-3)}} \leq n^{1+2e^4 \cdot c^{-d}}$. ■

As explained after the statement of Corollary 21, the conclusions in Corollaries 27 and in 28 imply that for every sufficiently large constant d it holds that BFE can be computed by depth-d circuits of size $n^{1+(c')^{-d}}$, where $c' < e^{1/(k\cdot d_0)}$.

5 Bootstrapping derandomization of $\mathcal{TC}^0$ and $\mathcal{ACC}^0$

In this section we prove Theorem 4, and the bootstrapping result for derandomization of $\mathcal{ACC}^0$ (which was mentioned in the end of Section 1.2.1). In Section 5.1 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits and $\mathcal{TC}^0$ circuits that encode balanced codes. In Section 5.2 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits and $\mathcal{TC}^0$ circuits that compute averaging samplers. Finally, in we use the averaging samplers to prove Theorem 4 (in Section 5.3) and the bootstrapping result for derandomization of $\mathcal{ACC}^0$ (in Section 5.4).

5.1 A balanced code in uniform sparse $\mathcal{CC}^0[2]$ and $\mathcal{TC}^0$

In Section 5.1.1 we construct uniform sparse $\mathcal{CC}^0[2]$ circuits that compute *range detectors*, which are basic building blocks that will be useful in constructing the encoding circuit. Then, in Section 5.1.2 we use the range detectors to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that encode codes with constant relative distance. In Section 5.1.3 we use the foregoing construction to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that encode balanced codes. Finally, in Section 5.1.4 we show how to convert the latter circuits into uniform extremely sparse $\mathcal{TC}^0$ circuits that encode balanced codes.

5.1.1 Range detectors in uniform sparse $\mathcal{CC}^0[2]$ from lossless expanders

As pointed out in [Gál+13], the task of constructing range detectors reduces to the task of constructing unbalanced *lossless expanders*: The latter are bipartite graphs with n "input" vertices and m "output" vertices such that each input vertex has degree d, and each set S of input vertices of size at most w has at least $(1-\epsilon)\cdot d\cdot|S|$ distinct neighbors among the output vertices. In particular, for each set S of size at most w, at least $(1-2\epsilon)\cdot d\cdot|S|$ of its neighbors are connected to a *single* vertex in S. [24]

The reason that lossless expanders are useful for constructing range detectors is the following. Assume that we construct a depth-one circuit $C:\{0,1\}^n\to\{0,1\}^m$ of parity gates by wiring according to a lossless expander. Then, for every input $x\in\{0,1\}^n$ with Hamming weight between $w/2$ and w, we have that $S_x=\{i\in[n]:x_i=1\}$ satisfies $|S_x|\leq w$. Since the graph is an expander, at least $(1-2\epsilon)\cdot d\cdot|S_x|$ output gates are connected to a *single* coordinate $i\in[n]$ such that $x_i=1$; hence, the Hamming weight of the output $C(x)$ is at least $(1-2\epsilon)\cdot d\cdot|S_x|=\Omega(d\cdot w)$. Also, since $|S_x|\leq w$, the Hamming weight of $C(x)$ is at most $d\cdot w$. Thus, if we use a lossless expander with $m=O(d\cdot w)$, we obtain an $(n,m,w,\Omega(m),m/2)$ range detector.

For our purposes we need a lossless expander with *small* input-degree d (to get a sparse circuit) and $m=O(d\cdot w)$. We will use the explicit construction of lossless expanders by Capalbo *et al.* [Cap+02]. To be consistent with their notation, we use uppercase letters instead of lowercase ones (i.e., the number of inputs is N, the number of outputs is M, and the weight is W). The input-degree that is obtained using their construction is $D=\exp(\text{poly}\log\log(N))$, which is sufficiently small for our purposes.

Proposition 29 *(constructing sparse range detectors using [Cap+02]). There exist two constants $\eta,\mu>0$ such that the following holds. There exists a deterministic polynomial-time algorithm that gets as input 1^N, where N is a power of two, and $W\leq\eta\cdot N$, and outputs the following depth-one circuit that consists of parity gates. The circuit has N input bits and*

[24]This is because $(1-\epsilon)\cdot d\cdot|S|$ edges are needed to obtain $(1-\epsilon)\cdot d\cdot|S|$ distinct neighbors of S, which leaves only $\epsilon\cdot d\cdot|S|$ edges that can "donate" a second edge to a neighbor of S.

$M = O\left(W \cdot \exp(\operatorname{poly}\log\log(N))\right)$ *output gates, and* $N \cdot \exp(\operatorname{poly}\log\log(N))$ *wires, and the circuit computes an* $(N, M, W, \mu \cdot M, M/2)$*-range detector.*

Proof. We use the lossless expander construction of [Cap+02, Thm 7.3], instantiated with the following parameters: $n = \log(N)$, and $\epsilon = 0.01$, and $t \in \mathbb{N}$ such that $t + \alpha \cdot \log^3(t/\epsilon) = n - \log(W) - \beta$, where $\alpha, \beta \geq 1$ are positive constants that will be specified later. [25] Their construction yields a lossless expander with the following parameters:

1. The number of input vertices is N.
2. The number of output vertices is $M = 2^{n-t} = 2^{\beta} \cdot W \cdot 2^{\alpha \cdot \log^3(t/\epsilon)}$.
3. The degree of input vertices is $D = 2^d = 2^{\gamma \cdot \log^3(t/\epsilon)} < \exp(\operatorname{poly}\log\log(N))$ (where $\gamma > 1$ is a universal constant from [Cap+02]).
4. All sets $S \subseteq [N]$ of size at most $K = 2^{k_{max}}$ have at least $0.99 \cdot D \cdot |S|$ neighbors, where $K = 2^{n-t-d-\log(1/\epsilon)-\delta} \geq W$ (in the calculation, $\delta > 1$ is a universal constant from [Cap+02], and we choose $\alpha = \gamma$ and $\beta \geq \log(1/\epsilon) + \delta$).

The neighbor function $E : [N] \times [D] \to [M]$ is computable in time $\operatorname{poly}\log(N)$, which means that the circuit (with parity gates) corresponding to the expander can be constructed in time $\operatorname{poly}(N)$. By the preceding discussion, to prove that the circuit is an $(N, M, W, \Omega(M), M/2)$-range detector it remains to verify that $M \geq 2 \cdot D \cdot W$ and $M = O(D \cdot W)$. The first inequality holds because $\beta \geq 1$, and the second inequality holds because β is bounded by a universal constant. ■

Proposition 29 gives range detectors for all weights w up to $\Omega(n)$. Constructing range detectors for weight $w = \Omega(n)$ is much simpler: The identity mapping preserves the constant relative weight, and so we just need to handle the edge case of $w > n/2$ (recall that we want the output weight to be at most $(1 - \Omega(1)) \cdot m$).[26]

Proposition 30 *(constructing a layer of range detectors). For some universal constant $\rho > 0$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a power of two, and outputs the following $\log(n)$ depth-one circuits that consist of parity gates. For $i = 1, ..., \log(n)$, the i^{th} circuit computes an $(n, m, w, \rho \cdot m, (1 - \rho) \cdot m)$-range detector, where $w = 2^i$ and $m = O(w \cdot \exp(\operatorname{poly}\log\log(n))$ and the circuit has at most $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires.*

Proof. Let $\eta, \mu > 0$ be the two constants from Proposition 29. For every $i \leq \log(n) - \log(1/\eta)$, we use Proposition 29 to get an $(n, m, w, \eta \cdot m, m/2)$-range detector with $m = O(n \cdot \exp(\operatorname{poly}\log\log(n))$ and $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires. For $\log(n) - \log(1/\eta) < i < \log(n)$, we just use the identity mapping to get an $(n, m = n, w, \eta \cdot m, m/2)$-range detector with n wires.

For $i = \log(n)$, we map n input bits to $m = (3/2) \cdot n$ output bits as follows: The first n output bits are the identity mapping of the input bits, and each of the last $n/2$

[25] Specifically, in the proof α will be a specific universal constant, and β will be lower-bounded by another universal constant. We will thus increase β (by a constant) such that $n - \log(W) - \beta$ will touch the set $\{t + \alpha \cdot \log^3(t/\epsilon)\}_{t \in \mathbb{N}}$. The fact $n - \log(W) - \beta \geq 1 + \alpha \cdot \log^3(1/\epsilon)$ follows from the hypothesis that $W \leq \eta \cdot N$ (where η is sufficiently small and depends on α and on the lower bound for β).

[26] In this paper we will not actually use the fact that the output weight is at most $(1 - \Omega(1)) \cdot m$, but we nevertheless present this construction for completeness.

output bits is connected to a consecutive pair of input bits (i.e., for $i \in [n]$, the i^{th} input bit is connected to the i^{th} output bit; and for $i \in [n/2]$, the $(n+i)^{th}$ output bit is connected to input bits $2i-1$ and $2i$). Note that for any input $x \in \{0,1\}^n$ of Hamming weight at least $n/2$, the Hamming weight of the output is at least $n/2 = m/3$. Now, if the Hamming weight of x is at most $(3/4) \cdot n$, then the Hamming weight of the output is at most $n = (5/6) \cdot m$; and on the other hand, if the Hamming weight of x is more than $(3/4) \cdot n$, then $\Pr_{i \in [n/2]}[x_{2i-1} \oplus x_{2i} = 0] \geq \Pr_{i \in [n/2]}[x_{2i-1} = x_{2i} = 1] \geq 1/2$, which implies that the Hamming weight of the output is at most $m - n/4 = (5/6) \cdot m$. Thus, this yields an $(n, m, w = n, m/2, (5/6) \cdot m)$-range detector with $2n$ wires. The proposition follows by taking $\rho = \min\{\mu, \eta, 1/6\}$. ■

5.1.2 A code with constant relative distance in uniform sparse $\mathcal{CC}^0[2]$

Our goal now is to use the range detectors to construct an encoding circuit. To do so, we will put the range detectors as a layer of gates above the inputs, and use an additional top layer to combine the range detectors into a code. The construction of the top layer follows the construction in [Gál+13, p. Clm. 34], with different parameters.

Proposition 31 *(uniform extremely sparse $\mathcal{CC}^0[2]$ circuits for encoding an "almost-good" code). For some universal constant $\rho > 0$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a power of two, and outputs a depth-two circuit of that consists of parity gates, and satisfies the following:*

1. *The circuit computes the encoding function of a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance $\rho > 0$, where $\hat{n} = n \cdot \exp(\operatorname{poly}\log\log(n))$.*
2. *The circuit has $n \cdot \exp(\operatorname{poly}\log\log(n))$ gates and $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires.*

Proof. We first use Proposition 30 to obtain $\log(n)$ range detectors with parameters as in the proposition; the middle layer of the circuit consists of these range detectors. Note that for every non-zero $x \in \{0,1\}^n$ it holds that for some $i \in [\log(n)]$, the i^{th} range detector maps x to $m_i = O(2^i \cdot \exp(\operatorname{poly}\log\log(n)))$ outputs such that between $\rho \cdot m_i$ and $(1-\rho) \cdot m_i$ of the outputs are set to one.

Towards constructing the top layer, we first construct a top layer of n_0 gates, where $n_0 = n \cdot \exp(\operatorname{poly}\log\log(n)) \geq \max_{i \in [\log(n)]}\{m_i\}$, that satisfies the following property: For every non-zero $x \in \{0,1\}^n$, a constant fraction of the n_0 top gates are connected to a gate in the middle layer that is set to one. This property does not suffice for the complete construction, since a top gate might touch an *even* number of gates in the middle layer that are set to one (i.e., their parity will be zero); later on we will replace the top n_0 gates by $\hat{n}$ parity gates in a manner that will solve this problem.

For now, imagine a top layer of n_0 gates. For $i \in [\log(n)]$, we connect the i^{th} range detector to these top gates, while ensuring that if a constant fraction of the outputs of the range detector are set to one, then a constant fraction of the top gates touch an output gate of the range detector that is set to one. Specifically, we connect the layers such that each top gate is connected to exactly one output of the range detector, and the degree of the outputs of the range detector is at least $\lfloor n_0/m_i \rfloor$.[27] Let S be the set of output gates of the range detector that are set to one. If $|S| \geq \rho \cdot m_i$, then the number

[27]That is, we split the n_0 top gates into m_i blocks of size either $\lceil n_0/m_i \rceil$ or $\lfloor n_0/m_i \rfloor$, and connect each of the m_i gates in the middle to all top gates in a corresponding block.

of wires outgoing from S is at least $\rho \cdot m_i \cdot \lfloor n_0/m_i \rfloor$, which implies that the number of top gates connected to S is at least $\rho \cdot m_i \cdot \lfloor n_0/m_i \rfloor = \Omega(\rho \cdot n_0)$.

After connecting all range detectors to the top layer of n_0 gates in this manner, the degree of each top gate is exactly $\log(n)$. To obtain the actual top layer (with $\hat{n}$ parity gates instead of n_0 gates), we replace each of the n_0 gates, denoted g, with $O(\log(n))$ parity gates that compute a good code on the middle gates that feed into g. (This can be any code with constant rate and constant relative distance, since we do not require that these sub-circuits over $\log(n)$ bits will be sparse.) Thus, if a constant fraction of the previous n_0 top gates were connected to a gate in the middle layer that is set to one, then a constant fraction of the actual $\hat{n}$ top parity gates are set to one.

The number of wires between the top layer and the middle layer is at most $\hat{n} \cdot \log(n) = n_0 \cdot O(\log(n)) \cdot \log(n) = n \cdot \exp(\operatorname{poly}\log\log(n))$, and the number of wires between the middle layer and the inputs layer is also bounded by $n \cdot \exp(\operatorname{poly}\log\log(n))$. Also, the top layer has more gates than the middle layer, and thus the circuit has $O(n \cdot \exp(\operatorname{poly}\log\log(n)))$ gates. ■

5.1.3 A balanced code in uniform sparse $\mathcal{CC}^0[2]$

In this section we use an additional sparse circuit to amplify the distance of the code in Proposition 31 from $\Omega(1)$ to $1/2 - \epsilon$. This is done relying on the well-known strategy of Naor and Naor [NN93], which uses random walks on an expander on the vertex-set $[\hat{n}]$. Specifically, we use the following construction:

Proposition 32 *(amplifying the distance of a code by a sparse circuit; see [Tel17b, Prop. 6.6]). For some universal constant $r_0 > 1$, there exists a polynomial-time algorithm that is given as input $1^{\hat{n}}$, a constant $\rho > 0$, and $\epsilon = \epsilon(\hat{n}) > 0$, and outputs a circuit C such that:*

1. *The circuit C maps $\hat{n}$ input bits to $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$ output bits.*
2. *For every $\hat{x} \in \{0,1\}^{\hat{n}}$ with relative Hamming weight at least ρ, the relative Hamming weight of $\bar{x} = C(\hat{x})$ is between $1/2 - \epsilon$ and $1/2$.*
3. *The circuit C has depth one, and each output bit of C is a linear function of $O(\log(1/\epsilon)/\rho)$ input bits.*[28]

Combining Propositions 31 and 32, we get the following:

Theorem 33 *(a balanced code in superlinear $\mathcal{CC}^0$). For some universal constant $r_1 > 1$ there exists a polynomial-time algorithm that is given as input 1^n and $\epsilon = \epsilon(n)$, and outputs a $\mathcal{CC}^0[2]$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1}$ such that every codeword has relative Hamming weight $1/2 \pm \epsilon$.*
2. *The circuit C has depth two and at most $n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1} \cdot \log(1/\epsilon)$ wires.*

[28] In [Tel17b, Prop. 6.6] the circuit is a depth-two $\mathcal{TC}^0$ circuit rather than a depth-one $\mathcal{CC}^0[2]$ circuit. However, in the $\mathcal{TC}^0$ circuit every output bit is just the parity of $O(\log(1/\epsilon)/\rho)$ input bits, so we can construct an equivalent depth-one $\mathcal{CC}^0[2]$ circuit.

Proof. Let $\rho > 0$ be any sufficiently small constant. We first use the algorithm from Proposition 31 to construct a depth-two $\mathcal{CC}^0[2]$ circuit $C_0 : \{0,1\}^n \to \{0,1\}^{\hat{n}}$, where $\hat{n} = n \cdot \exp(\operatorname{poly}\log\log(n))$, that computes a linear code with relative distance ρ using $n \cdot \exp(\operatorname{poly}\log\log(n))$ wires. Now, we use the algorithm from Proposition 32 to construct a depth-one $\mathcal{CC}^0[2]$ circuit $C_1 : \{0,1\}^{\hat{n}} \to \{0,1\}^{\bar{n}}$, where $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$, that maps any $\hat{x} \in \{0,1\}^{\hat{n}}$ of relative weight at least ρ to $\bar{x} \in \{0,1\}^{\bar{n}}$ of relative weight at least $1/2 - \epsilon$, using $O(\bar{n} \cdot \log(1/\epsilon))$ wires.

We now combine C_0 and C_1. Note that the naive combination has depth three, but that the top layer has fan-in $O(\log(1/\epsilon))$ and that the middle layer has fan-in $O(\log(n))$. Thus, we can collapse the two layers, and obtain a depth-two circuit in which the top layer has fan-in $O(\log(n) \cdot \log(1/\epsilon))$. Overall, we obtain a circuit with at most $n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\epsilon)^{r_0/\rho} \cdot \log(1/\epsilon)$ wires. ■

Relying on the Johnson bound (see, e.g., [AB09, Thm. 19.23]), Theorem 33 also yields a list-decodable code:

Corollary 34 *(a list-decodable code in superlinear $\mathcal{CC}^0$). For some universal constant $r_2 > 1$, there exists a polynomial-time algorithm that is given as input 1^n and $\delta = \delta(n)$, and outputs a $\mathcal{CC}^0[2]$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\delta)^{r_2}$ such that in any Hamming ball of radius $1/2 - \delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $O(1/\delta^2)$ codewords.*

2. *The circuit C has depth two and at most $n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\delta)^{r_2} \cdot \log(1/\delta)$ wires.*

5.1.4 A balanced code in extremely sparse $\mathcal{TC}^0$

In this section we convert our construction of uniform $\mathcal{CC}^0[2]$ circuits into uniform extremely sparse $\mathcal{TC}^0$ circuits. We do this by replacing each parity gate by a $\mathcal{TC}^0$ circuit, using the constructions of uniform circuits by [BBL92; PS94], in each of the constructions in the previous sections. We start with the construction of encoding circuits for a code with constant relative distnce:

Proposition 35 *(uniform extremely sparse $\mathcal{TC}^0$ circuits for encoding an "almost-good" code). For some universal constants $\rho > 0$ and $c_0 > 1$, there exists a deterministic polynomial-time algorithm that gets as input 1^n, where n is a sufficiently large power of two, and a constant $d \geq 2$, and outputs a $\mathcal{TC}^0$ circuit that satisfies the following:*

1. *The circuit computes the encoding function of a linear code $\{0,1\}^n \to \{0,1\}^{\hat{n}}$ with constant relative distance $\rho > 0$, where $\hat{n} = n \cdot \exp(\operatorname{poly}\log\log(n))$.*

2. *The circuit has depth $d + 2$ and at most $n^{1+c_0 \cdot \phi^{-d}+o(1)}$ wires, where $\phi = \frac{1+\sqrt{5}}{2}$.*

Proof. Observe that in the construction in the proof of of Proposition 31, each of the gates in the middle layer (i.e., the gates that compute the range detectors) may compute the parity of an arbitrary number of input bits, but each gate in the top layer only computes the parity of $\log(n)$ gates in the middle layer.

We first replace each parity gate in the middle layer by a depth-d $\mathcal{TC}^0$ circuit for parity, using the construction in [PS94, Thm. 1]. For every gate g, denote by n_g

the fan-in of g; then, the $\mathcal{TC}^0$ circuit that computes g has $n_g^{1+c_0\cdot\phi^{-d}}$ wires, for some $c_0 > 1$. Thus, the overall number of wires that are contributed by the $\mathcal{TC}^0$ circuits that compute the middle layer is at most

$$\sum_g n_g^{1+c_0\cdot\phi^{-d}} \leq \left(\sum_g n_g\right)^{1+c_0\cdot\phi^{-d}} \leq (n \cdot \exp(\operatorname{poly}\log\log(n)))^{1+c_0\cdot\phi^{-d}} ,$$

which is $n^{1+c_0\cdot\phi^{-d}+o(1)}$.

Now, we replace each gate in the top layer, which computes the parity of $\log(n)$ bits, by a $\mathcal{TC}^0$ circuit of depth two with $O(\log^2(n))$ wires (using the standard construction with a quadratic number of wires; see, e.g., [BBL92, Sec. 1]). The number of $\mathcal{TC}^0$ circuits that we use (i.e., the number of top gates) is $n \cdot \exp(\operatorname{poly}\log\log(n))$, and each of them contributes $O(\log^2(n))$ wires, so the overall contribution of this layer to the number of wires is $n^{1+o(1)}$. ■

In Proposition 32, we amplify the distance of the code from $\Omega(1)$ to $1/2 - \epsilon$ by a single layer of parity gates, each of which has fan-in at most $O(\log(1/\epsilon))$. By converting each such parity gate to a depth-two $\mathcal{TC}^0$ circuit, we obtain the following:

Proposition 36 *(amplifying the distance of a code by a sparse $\mathcal{TC}^0$ circuit). For some universal constant $r_0 > 1$, there exists a polynomial-time algorithm that is given as input $1^{\hat{n}}$, a constant $\rho > 0$, and $\epsilon = \epsilon(\hat{n}) > 0$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C maps $\hat{n}$ input bits to $\bar{n} = \hat{n} \cdot (1/\epsilon)^{r_0/\rho}$ output bits.*
2. *For every $\hat{x} \in \{0,1\}^{\hat{n}}$ with relative Hamming weight at least ρ, the relative Hamming weight of $\bar{x} = C(\hat{x})$ is between $1/2 - \epsilon$ and $1/2$.*
3. *Each output bit of C is a linear function of $O(\log(1/\epsilon)/\rho)$ input bits.*
4. *The circuit C has depth two and $O\left(\hat{n} \cdot (1/\epsilon)^{r_0/\rho} \cdot \log^2(1/\epsilon)\right)$ wires.*

Now, by combining Propositions 35 and 36, we get the following:

Theorem 37 *(a balanced code in extremely sparse $\mathcal{TC}^0$). For some universal constants $c_0 > 1$ and $r_1 > 1$, there exists a polynomial-time algorithm that is given as input 1^n and $\epsilon = \epsilon(n)$ and a constant $d \geq 2$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1}$ such that every codeword has relative Hamming weight $1/2 \pm \epsilon$.*
2. *The circuit C has depth $d + 4$ and at most $n^{1+c_0\cdot\phi^{-d}+o(1)} + n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\epsilon)^{r_1} \cdot \log^2(1/\epsilon)$ wires, where $\phi = \frac{1+\sqrt{5}}{2}$.*

Corollary 38 *(a list-decodable code in sparse $\mathcal{TC}^0$). For some universal constants $c_0 > 1$ and $r_2 > 1$, there exists a polynomial-time algorithm that is given as input 1^n and $\delta = \delta(n)$ and a constant $d \geq 2$, and outputs a $\mathcal{TC}^0$ circuit C such that:*

1. *The circuit C computes a linear code that maps messages of length n to codewords of length $\bar{n} = n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\delta)^{r_2}$ such that in any Hamming ball of radius $1/2 - \delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $O(1/\delta^2)$ codewords.*
2. *The circuit C has depth $d + 4$ and at most $n^{1+c_0\cdot\phi^{-d}+o(1)} + n \cdot \exp(\operatorname{poly}\log\log(n)) \cdot (1/\delta)^{r_2} \cdot \log^2(1/\delta)$ wires, where $\phi = \frac{1+\sqrt{5}}{2}$.*

5.2 An averaging sampler in uniform sparse $\mathcal{CC}^0[2]$ and $\mathcal{TC}^0$

Let us recall the notion of weak combinatorial designs, which was introduced by Raz, Reingold, and Vadhan [RRV02], and state their result that Trevisan's extractor [Tre01] can be instantiated with weak designs instead of standard combinatorial designs.

Definition 39 *(weak designs). For positive integers $m, \ell, t \in \mathbb{N}$ and an integer $\rho > 1$, an (m, ℓ, t, ρ)* weak design *is a collection of sets $S_1, ..., S_m \subseteq [t]$ such that for every $i \in [m]$ it holds that $|S_i| = \ell$ and $\sum_{j<i} 2^{|S_i \cap S_j|} < (m-1) \cdot \rho$.*

Theorem 40 *(extractors from weak designs [RRV02, Prop. 10]). Let $m < k < n$ be three integers, and let $\epsilon > 0$. Let $\mathtt{ECC} : \{0,1\}^n \to \{0,1\}^{\bar{n}}$ be a code such that in every Hamming ball of radius $1/2 - \delta$ in $\{0,1\}^{\bar{n}}$ there exist at most $1/\delta^2$ codewords, where $\delta = \epsilon/4m$. Let $S_1, ..., S_m \subseteq [t]$ be an (m, ℓ, t, ρ) weak design with $\ell = \log(\bar{n})$ and $\rho = \frac{k - 3 \cdot \log(m/\epsilon) - t - 3}{m}$. Then, the function $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ that is defined by $Ext(x,z) = (\mathtt{ECC}(x)_{z_{S_1}}, ..., \mathtt{ECC}(x)_{z_{S_m}})$ is a (k, ϵ)-extractor.*

We will need a specific construction of weak designs from [Tel18], in which the intersection parameter $\log(\rho)$ is large, but the universe size t is small (i.e., for sets of size $|S_i| = \ell$ we will require that $\log(\rho) \approx .99 \cdot \ell$ and $t \approx 1.01 \cdot \ell$).

Lemma 41 *(constructing weak designs [Tel17b, Lem. 6.2]). There exists an algorithm that gets as input $m \in \mathbb{N}$ and $\ell \in \mathbb{N}$ and $\rho \in \mathbb{N}$ such that $\log(\rho) = (1 - \alpha) \cdot \ell$, where $\alpha \in (0, 1/4)$, and satisfies the following. The algorithm runs in time $\mathrm{poly}(m, 2^\ell)$ and outputs an (m, ℓ, t, ρ) weak design, where $t = \lceil (1 + 4\alpha) \cdot \ell \rceil$.*

We now instantiate Theorem 40 with the code from Corollary 34 and the weak designs from Lemma 41 in order to construct uniform sparse $\mathcal{CC}^0[2]$ circuits that computes the following averaging sampler: The sampler gets an input of length n, and two parameters $0 < \gamma \ll \beta < 1$, and constructs a sampler that outputs $m = n^\gamma$ bits and has accuracy $1/m$ and error $2^{n^\beta - n}$.

Theorem 42 *(an averaging sampler in superlinear $\mathcal{CC}^0$). For any sufficiently large constant $r > 1$, there exists a polynomial-time algorithm that gets as input 1^n and two parameters $\beta = \beta(n) > 3/4$ and $\gamma = \gamma(n) < \frac{\beta - 3/4}{r}$, and outputs a $\mathcal{CC}^0[2]$ circuit C that satisfies the following:*

1. *The circuit C gets input $x \in \{0,1\}^n$ and outputs $2^t < n^{5 - 4(\beta - r\gamma)} \cdot \exp(\mathrm{poly}\log\log(n))$ strings of length $m = n^\gamma$.*
2. *The function $Samp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that $Samp(x,i) = C(x)_i$ is an averaging sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$.*
3. *The depth of C is three and its number of wires is $n^{5 - 4(\beta - r\gamma) + \gamma} \cdot \exp(\mathrm{poly}\log\log(n))$.*

In particular, if $\beta \geq 1 - r\gamma$, then the number of outputs of C is $2^t \leq n^{1 + 8r \cdot \gamma} \cdot \exp(\mathrm{poly}\log\log(n))$, and its number of wires is at most $n^{1 + (8r+1) \cdot \gamma} \cdot \exp(\mathrm{poly}\log\log(n))$.

Proof. Let $r_2 > 1$ be the constant from Corollary 34, let $r_3 = 3 \cdot r_2$, and let $r \geq r_3 + 2$. We first use Corollary 34 with the parameter value $\delta = \epsilon/4m$ to construct a depth-two circuit C_0 that encodes its input $x \in \{0,1\}^n$ to a codeword $\bar{x}$ of length $\bar{n} = n \cdot \exp(\mathrm{poly}\log\log(n)) \cdot (1/\delta)^{r_2}$. Then, we use Lemma 41 to construct an (m, ℓ, t, ρ) weak design $S_1, ..., S_m \subseteq [t]$ with the following parameters: For $\alpha = 1 - \beta + r \cdot \gamma < 1/4$

(the inequality is since $\beta > 3/4$ and $\gamma < (\beta - 3/4)/r$), we construct a design with $\ell = \log(\bar{n})$ and $\rho = 2^{(1-\alpha)\cdot\ell}$ and $t = \lceil (1+4\alpha)\cdot\ell \rceil$. Now, define a function $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ as in Theorem 40; that is, for $x \in \{0,1\}^n$ and $z \in \{0,1\}^t$, the m-bit string $Ext(x,z)$ is the projection of $\bar{x}$ to the coordinates $z_{S_1}, ..., z_{S_m}$. The circuit C outputs the 2^t strings corresponding to $\{Ext(x,z)\}_{z\in\{0,1\}^t}$, where each output string is a projections of m bits of $\bar{x}$.

Let $k = n^\beta$. By our choice of α, we have that $\rho = 2^{(1-\alpha)\cdot\ell} < k/2m < \frac{k-3\cdot\log(m/\epsilon)-t-3}{m}$.[29] Thus, relying on Theorem 40, the function Ext is an $(n^\beta, \epsilon = 1/m)$-extractor, and also (by Proposition 18) a sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$. The number of wires in C_0 is at most $n \cdot \exp(\text{poly}\log\log(n)) \cdot m^{r_3}$, and the number of wires between $\bar{x}$ and the outputs is $2^t \cdot m = 2^{\lceil (1+4\alpha)\cdot\log(\bar{n})\rceil} \cdot n^\gamma \leq n^{5-4\beta+(4r+1)\gamma} \cdot \exp(\text{poly}\log\log(n))$. Hence, the total number of wires in the sampler is at most $n^{5-4\beta+(4r+1)\gamma} \cdot \exp(\text{poly}\log\log(n))$. ■

We now construct averaging samplers in uniform extremely sparse $\mathcal{TC}^0$. Similarly to Theorem 42, we use Theorem 40 and Lemma 41, but this time with the code construction in $\mathcal{TC}^0$ from Corollary 38. The sampler will get an input of length n, and for two constants $0 < \gamma \ll \beta < 1$, the sampler will output $m = n^\gamma$ bits and have accuracy $1/m$ and error $2^{n^\beta - n}$.

Theorem 43 *(an averaging sampler in extremely sparse $\mathcal{TC}^0$). For a universal constant $c_0 > 1$ and any sufficiently large constant $r > 1$, there exists a polynomial-time algorithm that gets as input 1^n and three constants $d \geq 2$ and $\beta > 3/4$ and $\gamma < \frac{\beta - 3/4}{r}$, and outputs a $\mathcal{TC}^0$ circuit C that satisfies the following:*

1. *The circuit C gets input $x \in \{0,1\}^n$ and outputs $2^t < n^{(1+4r\cdot\gamma)\cdot(5-4\beta)}$ strings of length $m = n^\gamma$.*

2. *The function $Samp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ such that $Samp(x,i) = C(x)_i$ (i.e., $Samp(x,i) \in \{0,1\}^m$ is the i^{th} output string of $C(x)$) is an averaging sampler with accuracy $\epsilon = 1/m$ and error $2^{n^\beta - n}$.*

3. *The depth of C is $d+5$ and its number of wires is $n^{1+c_0\cdot\phi^{-d}+o(1)} + O\left(n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}\right)$, where $\phi = \frac{1+\sqrt{5}}{2}$.*

In particular, if $\gamma \leq 1/(24r\cdot\phi^d)$ and $\beta \geq 1 - \phi^{-d}/2$, then both the number of outputs of C (i.e., 2^t) and the number of wires in C are bounded by $n^{1+c_0\cdot\phi^{-d}+o(1)}$.

Proof. The parametrization of the sampler in this proof is similar to that in the proof of Theorem 42. Specifically, let $r_2 > 1$ and $c_0 > 1$ be the constants from Corollary 38, let $r_3 = 3 \cdot r_2$, and let $r \geq r_3 + 2$. We use Corollary 38 with $\delta = \epsilon/4m$ to construct a circuit C_0 of depth $d+4$ that encodes its input $x \in \{0,1\}^n$ to a codeword $\bar{x}$ of length $\bar{n} = n \cdot \exp(\text{poly}\log\log(n)) \cdot (1/\delta)^{r_2} < n^{1+o(1)} \cdot m^{r_3}$. Then, for $\alpha = 1 - \beta + r\cdot\gamma < 1/4$, we use Lemma 41 to construct an (m, ℓ, t, ρ) weak design $S_1, ..., S_m \subseteq [t]$ with $\ell = \log(\bar{n})$ and $\rho = 2^{(1-\alpha)\cdot\ell}$ and $t = \lceil (1+4\alpha)\cdot\ell \rceil$. Let $Ext : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$

[29]To see that $(1-\alpha)\cdot\ell < \log(k/2m)$, note the following. Since $\ell = \log(\bar{n}) < (1+o(1))\cdot\log(n) + r_3\cdot\log(m)$, we have that $(1-\alpha)\cdot\ell \leq (\beta - r\cdot\gamma)\cdot((1+o(1))\cdot\log(n) + r_3\cdot\log(m)) < (\beta - r\cdot\gamma + o(1))\cdot\log(n) + r_3\cdot\log(m)$. On the other hand, we have that $\log(k/2m) = \beta\cdot\log(n) - \log(2m)$. Thus, it suffices to prove that $(r\cdot\gamma - o(1))\cdot\log(n) - r_3\cdot\log(m) \geq \log(2m)$, which holds since $n = m^{1/\gamma}$ and $r - r_3 > 1$.

such that for $x \in \{0,1\}^n$ and $z \in \{0,1\}^t$, the m-bit string $Ext(x,z)$ is the projection of $\bar{x}$ to the coordinates $z_{S_1},...,z_{S_m}$. The circuit C outputs the 2^t strings $\{Ext(x,z)\}_{z \in \{0,1\}^t}$.

Let $k = n^{\beta}$, and note that $\rho = 2^{(1-\alpha)\cdot\ell} < k/2m < \frac{k-3\cdot\log(m/\epsilon)-t-3}{m}$. Relying on Theorem 40 and Proposition 18, the function Ext is a sampler with accuracy $\epsilon = 1/m$ and error $2^{n^{\beta}-n}$. The depth of C is $d+5$ (since the depth of C_0 is $d+4$, and the 2^t outputs are projections of $\bar{x}$). Finally, the number of wires in C_0 is at most $n^{1+c_0\cdot\phi^{-d}+o(1)} + n^{1+o(1)} \cdot m^{r_3} < n^{1+c_0\cdot\phi^{-d}+o(1)} + n^{1+\gamma\cdot r}$, and the number of wires between $\bar{x}$ and the outputs is $2^t \cdot m = 2^{\lceil(1+4\alpha)\cdot\log(\bar{n})\rceil} \cdot n^{\gamma} < n^{(1+r\cdot\gamma)\cdot(5-4\beta+4r\cdot\gamma)} < n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}$. Hence, the total number of wires in the sampler is $n^{1+c_0\cdot\phi^{-d}+o(1)} + O\left(n^{(1+r\cdot\gamma)\cdot(5-4\beta)+8r\cdot\gamma}\right)$. ■

5.3 Bootstrapping derandomization of $\mathcal{TC}^0$

Let us now formally state Theorem 4 and prove it using the averaging sampler from Theorem 43.

Theorem 44 *(Theorem 4, restated). For $\phi = \frac{1+\sqrt{5}}{2}$, there exists a universal constant $c_0 > 1$ such that for any sufficiently large constant $r > 1$ the following holds.*

Let $d \in \mathbb{N}$. Assume that for some $d' \geq d+7$ there exists an algorithm that gets as input a $\mathcal{TC}^0$ circuit $C' : \{0,1\}^n \to \{0,1\}$ with depth d' and $n^{1+c_1\cdot\phi^{-d'}}$ wires, where $c_1 = c_0 \cdot \phi^{d+5} + 1/r$, runs in time $T(n)$, and for $\beta = 1 - \phi^{-d'}$ satisfies the following: If C' rejects all but at most $2^{n^{\beta}}$ of its inputs, then the algorithm rejects C', and if C' accepts all but at most $2^{n^{\beta}}$ of its inputs, then the algorithm accepts C'.

Then, there exists an algorithm that for every $k \in \mathbb{N}$, when given as input a $\mathcal{TC}^0$ circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, runs in time $T(m^{24r\cdot k\cdot\phi^{d'}})$, and satisfies the following: If C accepts at least $2/3$ of its inputs then the algorithm accepts C, and if C rejects at least $2/3$ of its inputs then the algorithm rejects C.

Recall that in Theorem 4 the hypothesis is that for $c < \phi$ (e.g., $c = 1.61$) and every d', the quantified derandomization algorithm will be able to handle circuits with depth d' and $n^{1+c^{-d'}}$ wires. This hypothesis is stronger than the hypothesis in Theorem 44, since for any fixed $d \in \mathbb{N}$ and sufficiently large d' it holds that $c^{-d'} > c_1 \cdot \phi^{-d'}$.

Proof of Theorem 44. Let $c_0 > 1$ be the universal constant from Theorem 43, and assume that $r > 1$ is sufficiently large to satisfy the hypothesis of Theorem 43. We construct an algorithm that gets as input a $\mathcal{TC}^0$ circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, and acts as follows. For $\gamma = 1/(24r \cdot k \cdot \phi^{d'})$ and $n = m^{1/\gamma}$, the algorithm constructs a circuit $C' : \{0,1\}^n \to \{0,1\}$ of depth d' with $n^{1+c_1\cdot\phi^{-d'}}$ wires such that the following holds: If C rejects at least a $2/3$ fraction of its inputs, then C' rejects all but at most $2^{n^{\beta}}$ inputs; and if C accepts at least a $2/3$ fraction of its inputs, then C' accepts all but $2^{n^{\beta}}$ of its inputs. Then, the algorithm invokes the quantified derandomization algorithm for C', which runs in time $T(n) = T\left(m^{24r\cdot k\cdot\phi^{d'}}\right)$, to decide whether the acceptance probability of C is at least $2/3$ or at most $1/3$.

To construct C', let $d_{Samp} = d' - d - 5 \geq 2$; we first use Theorem 43 to construct a $\mathcal{TC}^0$ circuit $Samp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ that is an averaging sampler with the following properties: The input length is n, the output length is $m = n^{\gamma}$, the accuracy is $\epsilon = n^{\Omega(1)} < 1/100$, and the error is $\delta = 2^{n^{\beta}-n}$; the depth of $Samp$ is $d_{Samp}+5$, and by the "in particular" part of Theorem 43 (relying on the facts that $\beta = 1 - \phi^{-d'} >$

$1-\phi^{-d_{Samp}}/2$ and that $\gamma = 1/(24r \cdot k \cdot \phi^{d'}) < 1/(24r \cdot \phi^{d_{Samp}})$, the number of wires in $Samp$ is bounded by $n^{1+c_0 \cdot \phi^{-d_{Samp}}+o(1)}$. The circuit C' first computes the sampler $Samp$, then evaluates C in parallel on each of the $2^t < n^{1+c_0\cdot\phi^{-d_{Samp}}+o(1)}$ outputs of the sampler, and finally computes the majority of the 2^t evaluations of C. That is, $C'(x) = MAJ_{z\in\{0,1\}^t}[C(Samp(x,z))]$. The circuit C' is of depth $(d_{Samp}+5)+d+1$, but the gates in one of its layers (i.e., the output layer of the sampler) are just projections of the gates in the layer beneath it; therefore, we can collapse one layer, and obtain an equivalent circuit of depth $d' = d_{Samp}+d+5$. The number of wires in C' is at most

$$\begin{aligned} n^{1+c_0\cdot\phi^{-d_{Samp}}+o(1)} + 2^t \cdot m^k + 2^t &< n^{1+c_0\cdot\phi^{-d_{Samp}}+o(1)} + n^{1+c_0\cdot\phi^{-d_{Samp}}+1/(24r\cdot\phi^{d'})+o(1)} \\ &< n^{1+\left(c_0\cdot\phi^{d+5}+1/r\right)\cdot\phi^{-d'}}, \end{aligned}$$

where we relied on the facts that $m^k = n^{1/(24r\cdot\phi^{d'})}$ and that $\phi^{-d_{Samp}} = \phi^{d+5}\cdot\phi^{-d'}$.

Finally, note that for any $x \in \{0,1\}^n$ such that $\Pr_{z\in\{0,1\}^t}[C(Samp(x,z))=1] \in \Pr[C(\mathbf{u}_n)=1]\pm\epsilon$, we have that $C'(x)$ outputs the most frequent value of C. Since the error of the sampler is $\delta = 2^{n^\beta - n}$, the number of inputs $x\in\{0,1\}^n$ such that $\Pr_{z\in\{0,1\}^t}[C(Samp(x,z))=1] \notin \Pr[C(\mathbf{u}_n)=1]\pm\epsilon$ is at most 2^{n^β}. Thus, the circuit C' outputs the most frequent value of C on all but at most 2^{n^β} inputs $x\in\{0,1\}^n$. ■

Relying on known relaxations of Williams' "algorithmic method" (see [Wil13; SW13; BV14; FS16; MW18]), we obtain the following corollary of Theorem 44:

Corollary 45 *(quantified derandomization of sparse $\mathcal{TC}^0$ implies lower bounds for $\mathcal{TC}^0$). There exists a constant $\epsilon > 0$ such that the following holds. Let $c > 1$ be any fixed constant smaller than $\frac{1+\sqrt{5}}{2}$. Assume that if $\mathcal{NEXP} \subseteq \mathcal{TC}^0$, then for every $d \in \mathbb{N}$ there exists a* non-deterministic *machine that gets as input a $\mathcal{TC}^0$ circuit $C : \{0,1\}^n \to \{0,1\}$ with depth d and $n^{1+c^{-d}}$ wires, and also n^ϵ* bits of non-uniform advice, *runs in time $2^{n^{o(1)}}$, and solves the following problem: If C accepts all of its inputs, then there exist non-deterministic choices that cause the machine to accept C; and if C rejects all but $B(n) = 2^{n^{1-c^{-d}}}$ of its inputs, then the machine rejects C regardless of the non-determinism. Then, $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$.*

5.4 Bootstrapping derandomization of $\mathcal{ACC}^0$

In this section we prove that a algorithm for quantified derandomization of $\mathcal{ACC}^0$ circuits with $n^{1+\Omega(1)}$ wires and with a subexponential $B(n)$ that runs in time $2^{n^{o(1)}}$ would yield a corresponding algorithm for standard derandomization of $\mathcal{ACC}^0$ with "one-sided error" that runs in time $2^{n^{o(1)}}$.

The proof strategy is similar to that of the proof of Theorem 4. Specifically, we construct an algorithm that gets as input an $\mathcal{ACC}^0$ circuit $C : \{0,1\}^m \to \{0,1\}$, and outputs an $\mathcal{ACC}^0$ circuit $C' : \{0,1\}^n \to \{0,1\}$ such that if C has acceptance probability one then C' has acceptance probability one, and if C has acceptance probability at most half then C' rejects all but $B(n)$ of its inputs. To do so, the algorithm first constructs a uniform sparse $\mathcal{CC}^0[\oplus]$ circuit that computes an averaging sampler; this is done relying on the construction that was presented in Section 5.2. Then, for $n = \text{poly}(m)$, the algorithm constructs $C' : \{0,1\}^n \to \{0,1\}$ that first uses its input to sample inputs for C using the foregoing averaging sampler, then evaluates C on the inputs in the sample, and finally outputs the conjunction of the latter evaluations of C.[30]

Note that our transformation of C to C' only involves adding $\oplus$ gates and a single AND gate (at the top). Thus, our construction actually works not only for $\mathcal{ACC}^0$ circuits, but for essentially any circuit class whose gates can compute the AND and $\oplus$ functions (e.g., it also works for $\mathcal{AC}^0[\oplus]$). Specifically:

Theorem 46 *(a bootstrapping result for quantified derandomization of $\mathcal{ACC}^0$). Let $\mathcal{C}$ be any typical circuit class whose gates can compute the AND and $\oplus$ functions. Let $d \in \mathbb{N}$, and let $\gamma_d < 1/4r$, where $r > 1$ is a universal constant. Assume that there exists an algorithm that gets as input a $\mathcal{C}$-circuit $C' : \{0,1\}^n \to \{0,1\}$ with depth $d' = d+3$ and $O\left(n^{1+(8r+2)\cdot\gamma_d}\right) \cdot \exp(\text{poly}\log\log(n))$ wires, runs in time $T(n)$, and for $\beta = 1 - r \cdot \gamma_d$ satisfies the following: If C' rejects all but at most 2^{n^β} of its inputs, then the algorithm rejects C', and if C' accepts all of its inputs, then the algorithm accepts C'. Then, there exists an algorithm that for every $k \in \mathbb{N}$, when given as input a $\mathcal{C}$-circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, runs in time $T(m^{k/\gamma_d})$, and satisfies the following: If C accepts all of its inputs then the algorithm accepts C, and if C rejects at least half of its inputs then the algorithm rejects C.*

We comment that Theorem 46 holds also when γ_d is a *sub-constant* function of n (rather than a constant that depends only on d), albeit with a more complicated expression for the running time of the algorithm for standard derandomization in the conclusion of the theorem. We defer the discussion of this point until after the proof.

Proof of Theorem 46. Assume that $r > 1$ is sufficiently large to satisfy the hypothesis of Theorem 42. Our algorithm gets as input a $\mathcal{C}$-circuit $C : \{0,1\}^m \to \{0,1\}$ with depth d and m^k wires, and constructs a corresponding $\mathcal{C}$-circuit C', as follows.

Let n be the minimal integer such that $n^{\gamma_d(n)/k} \geq m$. The algorithm first uses Theorem 42 to construct a depth-three $\mathcal{CC}^0[2]$ circuit $Samp : \{0,1\}^n \times \{0,1\}^t \to \{0,1\}^m$ that is an averaging sampler with the following properties: For $\gamma' = \gamma_d(n)$ and $\beta' = 1 - r \cdot \gamma'$, the input length is $n = m^{k/\gamma'}$, the output length is m, the accuracy is $\epsilon = n^{\Omega(1)} < 1/2$, and the error is $\delta = 2^{n^{\beta'} - n}$; by the "in particular" part of Theorem 42, the number of wires in $Samp$ is bounded by $n^{1+(8r+1)\cdot\gamma'}$. Now, the circuit C' first computes the sampler $Samp$, then evaluates C in parallel on each of the $2^t < n^{1+8r\cdot\gamma'}$ outputs of the sampler, and finally computes the conjunction of the 2^t evaluations of C. That is, $C'(x) = \text{AND}_{z\in\{0,1\}^t}\left[C(Samp(x,z))\right]$.

The circuit C' is of depth $d+4$, but the gates in one of its layers (i.e., the output layer of $Samp$) just compute projections of the layer beneath it, so the algorithm can collapse this layer to obtain an equivalent circuit of depth $d' = d+3$. The number of wires in C' is $O\left(n^{1+(8r+2)\cdot\gamma'}\right)$. Also, we only added gates that compute AND and $\oplus$ functions, and therefore $C' \in \mathcal{C}$. Lastly, if C accepts all of its inputs then C' also accepts all of its inputs, and if C rejects at least half of its inputs, then C' accepts all but at most $2^{n^{\beta'}} > 2^{n^\beta}$ of its inputs (the inequality is since $2^{n^{\beta'}} = 2^{n^{1-r\cdot\gamma'}} > 2^{n^{1-r\cdot\gamma_d(n)}}$, relying again on the fact that $\gamma_d(n) > \gamma'$). The latter statement is since if C rejects at least half of its inputs, then for all but $\delta \cdot 2^n = 2^{n^{\beta'}}$ of the inputs $x \in \{0,1\}^n$ to $Samp$

[30]The reason that we use a top AND gate, instead of a sub-circuit for approximate majority, is that the known constructions for the latter circuit are of polynomial size (i.e., are not super-linear). Indeed, this is the reason that our result is limited to derandomization with "one-sided error".

it holds that $\Pr_{z\in\{0,1\}^t}[C(Samp(x,z))=1] \geq \Pr[C(\mathbf{u}_n)=1]-\epsilon > 0$; and for every $x\in\{0,1\}^n$ such that the latter holds we have that $C'(x)=1$.

Therefore, our algorithm can now invoke the quantified derandomization algorithm for C', which runs in time $T(n)$, to decide whether the acceptance probability of C is 1 or at most 1/2. ■

As mentioned after the statement of Theorem 46, the theorem holds also when γ_d is a sub-constant function of n, where the only change is that the running time of the algorithm for standard derandomization (in the conclusion of the theorem) will be larger. Specifically, given a function $\gamma_d(n)$ in the hypothesis, in the proof we set n to be the minimal integer such that $n^{\gamma_d(n)/k} \geq m$; then, the running time of the algorithm for standard derandomization will be $T(n)$ (rather than $T(m^{k/\gamma_d})$ as in Theorem 46). Note that when $\gamma_d(n)=o(1)$, the algorithm in the hypothesis of the theorem only needs to handle circuits with $n^{1+o(1)}$ wires; in particular, when $\gamma_d(n)=\frac{\text{poly}\log\log(n)}{\log(n)}$, the algorithm only needs to handle circuits of size $n\cdot\exp(\text{poly}\log\log(n))$.

6 Restrictions for $\mathcal{TC}^0$ circuits "just beyond" parity

In this section we present an open problem whose resolution would imply that $\mathcal{NEXP}\not\subseteq\mathcal{TC}^0$ (relying on Corollary 5). The open problem will focus on finding a representation of a "simple" function that approximates a given extremely sparse $\mathcal{TC}^0$ circuit $C:\{0,1\}^n\to\{0,1\}$ in a large subset $S\subseteq\{0,1\}^n$. To properly define the latter notion, let us first define the notion of a sampling circuit for a set $S\subseteq\{0,1\}^n$.

Definition 47 *(sampling circuits). A* sampling circuit for a set $S\subseteq\{0,1\}^n$ with error $\epsilon>0$ *is a circuit $U_S:\{0,1\}^r\to\{0,1\}^n$, where $r\geq\log(|S|)$, such that the output distribution of U_S (i.e., $U_S((\mathbf{u}_r))$ is ϵ-close to the uniform distribution over S, in statistical distance.*

Now, consider a circuit $C:\{0,1\}^n\to\{0,1\}$. As mentioned above, we will want to find a representation of "simple" function $\Phi:S\to\{0,1\}$, for some large $S\subseteq\{0,1\}^n$, such that Φ approximates $C\restriction_S$. Specifically, our notion of representation includes both a representation of the function Φ (e.g., a Boolean circuit that computes Φ) and a representation of the set S (i.e., a sampling circuit for S). And our notion of sufficiently simple is that there exists an algorithm that, when given $\Phi:S\to\{0,1\}$ and a sampling circuit for S, can efficiently approximate the bias of Φ. More formally:

Definition 48 *(sufficiently simple functions and sets). Let Γ be a class of pairs (Φ_n,U_{S_n}) where $\Phi_n:S_n\to\{0,1\}$ and U_{S_n} is a sampling circuit for $S_n\subseteq\{0,1\}^n$ with error 1/6. We say that Γ is* sufficiently simple *if there exists an algorithm that, when given as input $(\Phi_n,U_{S_n})\in\Gamma$, runs in time $2^{n^{o(1)}}$ and approximates the bias of Φ_n in S_n (i.e., $\Pr_{x\in S_n}[\Phi_n(x)=1]$), up to error 1/6.*

Let us now elaborate on the discussion that followed Corollary 5, with more specific details. Generalizing a high-level idea of Goldreich and Wigderson [GW14], we claim that quantified derandomization of a circuit $C:\{0,1\}^n\to\{0,1\}$ with $B(n)$ exceptional inputs in time $2^{n^{o(1)}}$ reduces to finding a pair (Φ_n,U_{S_n}) from a sufficiently simple class Γ such that $|S_n|\geq 6\cdot B(n)$ and $C\restriction_{S_n}$ is (1/6)-approximated by Φ_n (i.e., $\Pr_{x\in S_n}[\Phi_n(x)=C(x)]\geq 5/6$). This is the case because $C\restriction_{S_n}$ is biased towards the majority output of C (since C has only $B(n)$ exceptional inputs and $|S_n|\geq 6\cdot B(n)$);

and since, relying on the facts that Γ is sufficiently simple and that Φ_n is $(1/6)$-close to $C\restriction_{S_n}$, we can estimate the bias of $C\restriction_S$ up to error $1/3$ in time $2^{n^{o(1)}}$.

We therefore pose the following open problem. Intuitively, the problem asks to algorithmically find a sufficiently simple pair for a given biased $\mathcal{TC}^0$ circuit of size that is "just beyond" the size required to compute the parity function. More specifically:

Open Problem 49 *(restrictions for $\mathcal{TC}^0$ circuits of size "just beyond" parity). For $c < \frac{1+\sqrt{5}}{2}$, and a sufficiently simple class Γ, construct an algorithm for the following task. The algorithm gets as input $d \in \mathbb{N}$, and a $\mathcal{TC}^0$ circuit $C : \{0,1\}^n \to \{0,1\}$ of depth d with $n^{1+c^{-d}}$ wires that either accepts all its inputs or rejects all but $2^{n^{1-c^{-d}}}$ of its inputs, runs in time $2^{n^{o(1)}}$, and outputs $(\Phi_n, U_{S_n}) \in \Gamma$ such that $|S_n| \geq 6 \cdot 2^{n^{1-c^{-d}}}$ and $C\restriction_S$ is $(1/6)$-approximated by Φ_n.*

Relying on Corollary 5 and on the discussion above, a solution for Open Problem 49 would imply that $\mathcal{NEXP} \not\subseteq \mathcal{TC}^0$. Note that for every circuit C there exists a trivial sufficiently simple pair (Φ_n, U_{S_n}) that satisfies the requirements of the problem; namely, the constant function $\Phi \equiv \sigma$ where $\sigma \in \{0,1\}$ is the majority output of C, and the set $S_n = C^{-1}(\sigma)$ (the set S_n can be sampled by a circuit U_{S_n} with complexity similar to that of C, i.e. U_{S_n} samples inputs and verifies that they are indeed in $C^{-1}(\sigma)$). However, Open Problem 49 requires to *algorithmically* find a suitable sufficiently simple pair (Φ_n, U_{S_n}), while also allowing for less trivial solutions.

Moreover, similarly to Corollary 45, it suffices to solve a relaxed version of Open Problem 49. In particular, it suffices to construct the algorithm under the hypothesis that $\mathcal{NEXP} \subseteq \mathcal{TC}^0$, and the algorithm may use both non-determinism and n^{ϵ} bits of non-uniform advice (where $\epsilon > 0$ is a small universal constant from [FS16]).

Acknowledgements

The first author thanks his advisor, Ryan Williams, for all the support and advice during this work. The second author thanks Ryan Williams for hosting him in MIT, a visit which led to this collaboration. The second author is grateful to his advisor, Oded Goldreich, for his feedback on the manuscript, including in particular the suggestion of the term "bootstrapping results", and for his support and advice in the research and writing process. The second author is partially supported by Irit Dinur's ERC-CoG grant no. 772839.

References

[Aar17] Scott Aaronson. $P \overset{?}{=} NP$. Accessed at http://www.scottaaronson.com/papers/pnp.pdf, June 20, 2017. 2017.

[AB09] Sanjeev Arora and Boaz Barak. *Computational complexity: A modern approach.* Cambridge University Press, Cambridge, 2009.

[Ajt83] M. Ajtai. "Σ_1^1-formulae on finite structures". In: *Annals of Pure and Applied Logic* 24.1 (1983), pp. 1–48.

[AK10] Eric Allender and Michal Koucký. "Amplifying lower bounds by means of self-reducibility". In: *Journal of the ACM* 57.3 (2010), pp. 14, 36.

[Bar89] David A. Mix Barrington. "Bounded-Width Polynomial-Size Branching Programs Recognize Exactly Those Languages in NC^1". In: *Journal of Computer and System Sciences* 38.1 (1989), pp. 150–164.

[BBL92] Paul Beame, Erik Brisson, and Richard Ladner. "The complexity of computing symmetric functions using threshold circuits". In: *Theoretical Computer Science* 100.1 (1992), pp. 253–265.

[BIS90] David A. Mix Barrington, Neil Immerman, and Howard Straubing. "On uniformity within NC^1". In: *Journal of Computer and System Sciences* 41.3 (1990), pp. 274–306.

[BST90] David A. Mix Barrington, Howard Straubing, and Denis Thérien. "Non-uniform Automata over Groups". In: *Information and Computation* 89.2 (1990), pp. 109–132.

[Bus87] Samuel R. Buss. "The Boolean Formula Value Problem Is in ALOGTIME". In: *Proc. 19th Annual ACM Symposium on Theory of Computing (STOC)*. 1987, pp. 123–131.

[BV14] Eli Ben-Sasson and Emanuele Viola. "Short PCPs with projection queries". In: *Proc. 41st International Colloquium on Automata, Languages and Programming (ICALP)*. 2014, pp. 163–173.

[Cap+02] Michael Capalbo et al. "Randomness conductors and constant-degree lossless expanders". In: *Proc. 34th Annual ACM Symposium on Theory of Computing (STOC)*. 2002, pp. 659–668.

[Cha+06] Arkadev Chattopadhyay et al. "Lower bounds for circuits with MOD_m gates". In: *Proc. 47th Annual ACM Symposium on Theory of Computing (STOC)*. 2006.

[Coo85] Stephen A. Cook. "A taxonomy of problems with fast parallel algorithms". In: *Information and Control* 64.1-3 (1985), pp. 2–22.

[CSS16] Ruiwen Chen, Rahul Santhanam, and Srikanth Srinivasan. "Average-case lower bounds and satisfiability algorithms for small threshold circuits". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, 1:1–1:35.

[CSV84] Ashok K. Chandra, Larry J. Stockmeyer, and Uzi Vishkin. "Constant Depth Reducibility". In: *SIAM Journal of Computing* 13.2 (1984), pp. 423–439.

[Ete97] Kousha Etessami. "Counting quantifiers, successor relations, and logarithmic space". In: *Journal of Computer and System Sciences* 54.3 (1997), pp. 400–411.

[For+01] Jürgen Forster et al. "Relations Between Communication Complexity, Linear Arrangements, and Computational Complexity". In: *Proc. 21st Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2001, pp. 171–182.

[FS16] Lance Fortnow and Rahul Santhanam. "New non-uniform lower bounds for uniform classes". In: *Proc. 31st Annual IEEE Conference on Computational Complexity (CCC)*. 2016, Art. No. 19, 14.

[FSS84] Merrick Furst, James B. Saxe, and Michael Sipser. "Parity, circuits, and the polynomial-time hierarchy". In: *Mathematical Systems Theory* 17.1 (1984), pp. 13–27.

[GHR92] Mikael Goldmann, Johan Håstad, and Alexander Razborov. "Majority gates vs. general weighted threshold gates". In: *Proc. 7th Annual Structure in Complexity Theory Conference*. 1992, pp. 2–13.

[GK98] Mikael Goldmann and Marek Karpinski. "Simulating Threshold Circuits by Majority Circuits". In: *SIAM Journal of Computing* 27.1 (1998), pp. 230–246.

[GW13] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Electronic Colloquium on Computational Complexity: ECCC* 20 (2013), p. 152.

[GW14] Oded Goldreich and Avi Widgerson. "On derandomizing algorithms that err extremely rarely". In: *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. 2014, pp. 109–118.

[Gál+13] Anna Gál et al. "Tight bounds on computing error-correcting codes by bounded-depth circuits with arbitrary gates". In: *IEEE Transactions on Information Theory* 59.10 (2013), pp. 6611–6627.

[Haj+93] András Hajnal et al. "Threshold Circuits of Bounded Depth". In: *Journal of Computer and System Sciences* 46.2 (1993), pp. 129–154.

[Hås87] Johan Håstad. *Computational Limitations of Small-depth Circuits*. MIT Press, 1987.

[IPS97] Russell Impagliazzo, Ramamohan Paturi, and Michael E. Saks. "Size-depth tradeoffs for threshold circuits". In: *SIAM Journal of Computing* 26.3 (1997), pp. 693–707.

[KW16] Daniel M. Kane and Ryan Williams. "Super-linear Gate and Super-quadratic Wire Lower Bounds for Depth-two and Depth-three Threshold Circuits". In: *Proc. 48th Annual ACM Symposium on Theory of Computing (STOC)*. 2016, pp. 633–643.

[MV15] Eric Miles and Emanuele Viola. "Substitution-permutation networks, pseudorandom functions, and natural proofs". In: *Journal of the ACM* 62.6 (2015), Art. 46, 29.

[MW18] Cody Murray and Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-Polytime: An Easy Witness Lemma for NP and NQP". In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018.

[NN93] Joseph Naor and Moni Naor. "Small-bias probability spaces: efficient constructions and applications". In: *SIAM Journal of Computing* 22.4 (1993), pp. 838–856.

[NW94] Noam Nisan and Avi Wigderson. "Hardness vs. randomness". In: *Journal of Computer and System Sciences* 49.2 (1994), pp. 149–167.

[OS18] Igor Carboni Oliveira and Rahul Santhanam. "Hardness Magnification for Natural Problems". In: *Electronic Colloquium on Computational Complexity: ECCC* 25 (2018), p. 139.

[PS94] Ramamohan Paturi and Michael E. Saks. "Approximating threshold circuits by rational functions". In: *Information and Computation* 112.2 (1994), pp. 257–272.

[Raz87] Alexander A. Razborov. “Lower bounds on the size of constant-depth networks over a complete basis with logical addition”. In: *Mathematical Notes of the Academy of Science of the USSR* 41.4 (1987), pp. 333–338.

[RR97] Alexander A. Razborov and Steven Rudich. “Natural proofs”. In: *Journal of Computer and System Sciences* 55.1, part 1 (1997), pp. 24–35.

[RRV02] Ran Raz, Omer Reingold, and Salil Vadhan. “Extracting all the randomness and reducing the error in Trevisan’s extractors”. In: *Journal of Computer and System Sciences* 65.1 (2002), pp. 97–128.

[Smo90] Roman Smolensky. “On interpolation by analytic functions with special properties and some weak lower bounds on the size of circuits with symmetric gates”. In: *Proc. 31st Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1990, pp. 628–631.

[SW13] Rahul Santhanam and Ryan Williams. “On medium-uniformity and circuit lower bounds”. In: *Proc. 28th Annual IEEE Conference on Computational Complexity (CCC)*. 2013, pp. 15–23.

[Tel17a] Roei Tell. “Improved Bounds for Quantified Derandomization of Constant-Depth Circuits and Polynomials”. In: *Proc. 32nd Annual IEEE Conference on Computational Complexity (CCC)*. 2017, 18:1 –18:49.

[Tel17b] Roei Tell. “Quantified Derandomization of Linear Threshold Circuits (rev. 2)”. In: *Electronic Colloquium on Computational Complexity: ECCC* 24 (2017), p. 145.

[Tel18] Roei Tell. “Quantified Derandomization of Linear Threshold Circuits”. In: *Proc. 50th Annual ACM Symposium on Theory of Computing (STOC)*. 2018, pp. 855–865.

[Tre01] Luca Trevisan. “Extractors and Pseudorandom Generators”. In: *Journal of the ACM* 48.4 (2001), pp. 860–879.

[Vad12] Salil P. Vadhan. *Pseudorandomness*. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

[Wil11] Ryan Williams. “Non-uniform ACC circuit lower bounds”. In: *Proc. 26th Annual IEEE Conference on Computational Complexity (CCC)*. 2011, pp. 115–125.

[Wil13] Ryan Williams. “Improving Exhaustive Search Implies Superpolynomial Lower Bounds”. In: *SIAM Journal of Computing* 42.3 (2013), pp. 1218–1244.

[Yao85] Andrew C-C. Yao. “Separating the Polynomial-time Hierarchy by Oracles”. In: *Proc. 26th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1985, pp. 1–10.

Appendix A Proof of Theorem 24

In this appendix we prove Theorem 24, which asserts the existence of a strong self-reduction for the monoid problem when the size of the monoid grows with the input size. Recall that in the latter problem we fix an associative length-preserving binary operation $\{\times : (\{0,1\}^r)^2 \to \{0,1\}^r\}_{r\in\mathbb{N}}$, and for every two integers $m \geq \ell \in \mathbb{N}$, the corresponding problem $\mathsf{MON}^{\times}_{m,\ell}$ is defined on input length $n = m \cdot \ell$ by $\mathsf{MON}^{\times}_{m,\ell}(x) = \prod_{i\in[m]} \sigma_i$, where $x = (\sigma_1, ..., \sigma_m) \in \{0,1\}^n$ and $\sigma_i \in \{0,1\}^\ell$ for all $i \in [m]$.

Theorem 50 *(strong self-reduction for the monoid problem with monoids of growing size). Let $\mathcal{C}$ be a typical constant-depth circuit class, and let $\{\times : (\{0,1\}^r)^2 \to \{0,1\}^r\}_{r\in\mathbb{N}}$ be an associative operation. Let $\ell : \mathbb{N} \to \mathbb{N}$ such that $\ell(m) = m^{o(1)}$. Assume that there exist constants $d_0, k \in \mathbb{N}$ such that for every constant $\rho > 0$ and any sufficiently large $m \in \mathbb{N}$ it holds that $MON^{\times}_{m^\rho,\ell(m)}$ can be be solved by a family of $\mathcal{C}$-circuits of depth d_0 and size n^k, where $n = m^\rho \cdot \ell(m)$. Then, for infinitely many constants d it holds that $MON^{\times}_{m,\ell(m)}$ can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^2\cdot c^{-d}}$, where $c = e^{1/(k\cdot d_0)}$ and $n = m\cdot\ell(m)$.*

Proof. For any $\epsilon > 0$, let $d_1 = \left\lceil k \cdot \ln\left(\frac{(1-1/k)\cdot(1+\epsilon)}{\epsilon}\right)\right\rceil + 1$. For any sufficiently large $n \in \{m\cdot\ell(m) : m \in \mathbb{N}\}$, we denote $\ell = \ell(m)$, and construct a circuit for $\mathsf{MON}^{\times}_{m,\ell}$ of depth $d = d_1 \cdot d_0$ whose number of wires is $O\big(n^{1+\epsilon}\big) < n^{1+2e^2\cdot e^{-d/(k\cdot d_0)}}$.

As in the proof of Theorem 19, for any $x \in \mathbb{N}$ we denote $\lceil x\rceil_\ell = \ell\cdot\lceil x/\ell\rceil$ and $\lfloor x\rfloor_\ell = \ell\cdot\lfloor x/\ell\rfloor$. Let $0 = \alpha_1 < \alpha_2 < \ldots < \alpha_{d_1} < 1$ such that for every $i = 2,\ldots,d_1$ it holds that $\alpha_i = \frac{1+\epsilon}{k} + (1-1/k)\cdot\alpha_{i-1}$. Our circuit consists of d_1 layers (again, $i=1$ is the top layer and $i = d_1$ is the layer above the inputs) such that each layer $i \in [d_1]$ is of depth d_0 and contains $n_i = \lceil n^{\alpha_i}\rceil$ circuits. Each circuit in layer $i \in [d_1]$ has $\lceil b_i\rceil_\ell$ or $\lfloor b_i\rfloor_\ell$ input bits and ℓ output bits, where $b_i = n_{i+1}/n_i$ (and $b_{d_1} = n/n_{d_1}$), and computes $\mathsf{MON}^{\times}_{\lceil b_i\rceil_\ell,\ell}$ or $\mathsf{MON}^{\times}_{\lfloor b_i\rfloor_\ell,\ell}$; since $b_i = n^{\Omega(1)}$, by our hypothesis there exist such circuits of depth d_0 and size at most $(\lceil b_i\rceil_\ell)^k$. Since $\alpha_1 = 0$, the top layer contains a single circuit; and since $\alpha_i = (1+\epsilon)\cdot\big(1-(1-1/k)^{i-1}\big)$, in layer d_1 we have that $\alpha_{d_1} \geq 1-\epsilon/(k-1)$.

The foregoing circuit is of depth $d_1\cdot d_0$. To see that it computes $\mathsf{MON}^{\times}_{n,\ell}$, note that each layer is connected to all outputs of the layer beneath it, and that the operation $\times$ is associative. To see that the circuit has $O\big(n^{1+\epsilon}\big)$ wires, note that for $i \in [d_1 - 1]$, in the i^{th} layer we have $\lceil n^{\alpha_i}\rceil$ circuits, and the number of inputs bits of each of them is at most $\lceil n_{i+1}/n_i\rceil_\ell < n_{i+1}/n_i + \ell < 2\cdot n_{i+1}/n_i$ (in the last inequality we used the fact that $n_{i+1}/n_i = n^{\Omega(1)}$ and that $\ell = \ell(m) = m^{o(1)} < n^{o(1)}$). Thus, the total number of wires in the i^{th} layer is $O\Big(n^{\alpha_i + k\cdot(\alpha_{i+1}-\alpha_i)}\Big) = O\big(n^{1+\epsilon}\big)$. In level $i = d_1$, each circuit has $\lceil n/n_{d_1}\rceil_\ell < 2\cdot n^{1-\alpha_{d_1}}$ input bits, and thus the total number of wires in level d_1 is $O\Big(n^{\alpha_{d_1}+k\cdot(1-\alpha_{d_1})}\Big) \leq O\Big(n^{k-(k-1)\cdot\alpha_{d_1}}\Big) \leq O\big(n^{1+\epsilon}\big)$. ■

The conclusion of Theorem 50 can be extended to hold for every sufficiently large $d \in \mathbb{N}$ (at the expense of a slightly worse bound on the number of wires), precisely as in the proof of Corollary 20. Thus, we get the following:

Theorem 51 *(Theorem 24, restated). Let $\mathcal{C}, \times, \ell, d_0$ and k be as in the hypothesis of Theorem 50. Then, for every sufficiently large $d \in \mathbb{N}$ it holds that $MON^{\times}_{m,\ell(m)}$ can be solved by a family of $\mathcal{C}$-circuits of depth d and size $n^{1+2e^3\cdot c^{-d}}$, where $c = e^{1/(k\cdot d_0)}$ and $n = m\cdot\ell(m)$.*

Appendix B Minimal-depth tree with a bounded cost

Recall that in Section 2.1 we presented the following problem, whose solution underlies our improved self-reductions for the monoid problem:

> Given a cost function $c : \mathbb{N} \to \mathbb{N}$ and a bound function $B : \mathbb{N} \to \mathbb{N}$, we want to find a minimal d such that there exists a tree T of depth d over n inputs such that $Cost(T) = \sum_{v\in Nodes(T)} c(\deg(v)) \leq B(n)$.

Let us now prove that the solution for this problem when $c(m) = m^k$, which was described in Section 2.1, is indeed optimal. We conduct the analysis under the following two simplifying assumptions:

- We will use a tree in which all nodes in the same layer have the same fan-in. This is since the nodes are connected to disjoint sets of inputs, so any optimization applied to one node should also be applied to all other nodes.
- All fan-ins in each layer will be of the form n^α, for some constant $\alpha \in (0, 1)$. This is because we want a tree with minimal constant depth.

Given the above assumptions, to define a construction we just need to determine the fan-ins of nodes in each layer. For $i = 1, ..., d-1$, denote by n^{α_i} the fan-in of nodes in level i, where $i = 1$ is the root and $i = d-1$ is the next-to-bottom layer; the fan-in of the bottom layer $i = d$ is determined by the other fan-ins, since the bottom layer needs to touch all n inputs. Also denote $\beta_i = \sum_{j<i} \alpha_j$. (For convenience we will treat "empty summations" as the constant 0 and "empty products" as the constant 1.)

We want to determine the minimal d such that there is a setting of $\alpha_1, ..., \alpha_{d-1}$ in which the total cost of the tree is $O(n^{1+\epsilon})$. In particular, we require that:

1. The cost of each layer $i = 1, ..., d-1$ will be at most $n^{1+\epsilon}$. The number of nodes in layer i is n^{β_i}, and the cost of each node is $n^{k \cdot \alpha_i}$; therefore, we require that
$$\forall i \in [d-1], \;\; \beta_i + k \cdot \alpha_i \leq 1 + \epsilon \, . \tag{B.1}$$
2. The cost of the last layer $i = d$ will be at most $n^{1+\epsilon}$. The fan-in of the last layer is $n/n^{\beta_d} = n^{1-\beta_d}$ (since the last layer touches all inputs), so we require that
$$\beta_d + k \cdot (1 - \beta_d) \leq 1 + \epsilon \Longrightarrow \beta_d \geq 1 - \frac{\epsilon}{k-1} \, . \tag{B.2}$$

In other words, we want to find the smallest d such that it is possible to disperse $1 - \frac{\epsilon}{k-1}$ "weight" into $d-1$ "buckets" such that for each "bucket" $i \in [d-1]$ it holds that $\beta_i + k \cdot \alpha_i \leq 1 + \epsilon \Rightarrow \alpha_i \leq \frac{1+\epsilon-\beta_i}{k}$. Indeed, the optimal strategy for this problem is the *greedy* strategy, which for $i = 1, ..., d-1$, sets α_i to the maximal allowed value:

Proposition 52 *(the greedy strategy is optimal). For every fixed d, the maximal β_d is obtained by setting the value of each α_i, for $i = 1, ..., d-1$, to be the maximal value allowed by Eq.* (B.1)*, which is $\alpha_i = \frac{1+\epsilon-\beta_i}{k}$.*

Proof. Fix $d \in \mathbb{N}$. We prove by induction on $i = d-1, ..., 1$ that for every values for $\alpha_1, ..., \alpha_{i-1}$, the maximal value of β_d is obtained by setting each α_j, sequentially for $j = i, ..., d-1$, to the value $\alpha_j = \frac{1+\epsilon-\beta_j}{k}$.

The base case is $i = d-1$, which is immediate: For every fixed β_{d-1} it holds that $\beta_d = \beta_{d-1} + \alpha_{d-1}$, so we better take the maximal α_{d-1} possible. For the induction step, assume that the induction hypothesis holds for $i+1, ..., d-1$, and let us prove that it also holds for i. For any fixed β_i, we ask what is the maximal value β_d^* of β_d as a function of α_i. For any value of α_i, by the induction hypothesis it holds that β_d^* will be obtained by the greedy strategy; thus, when α_i is fixed, for every $j \geq i+1$ define $\alpha_j^* = \frac{1+\epsilon-\beta_j^*}{k}$ and define β_j^* accordingly. It follows that:
$$\beta_d^*(\alpha_i) = \beta_i + \alpha_i + \sum_{j=i+1}^{d-1} \frac{1+\epsilon-\beta_j^*}{k} = C + \alpha_i - \frac{1}{k} \sum_{j=i+1}^{d-1} \beta_j^* \, , \tag{B.3}$$

where $C = \left(\beta_i + \frac{(d-1-i)\cdot(1+\epsilon)}{k}\right)$ is independent of α_i. Our goal will be to show that β_d^* is monotone in α_i, which will complete the proof. To do so, we need to show that the coefficient of α_i in the expression $\sum_{j=i+1}^{d-1} \beta_j^*$ is less than k.

By definition, $\beta_{i+1}^* = \beta_i + \alpha_i$ and for every $j \geq i+2$ we have that $\beta_j^* = \beta_{j-1}^* + \frac{1+\epsilon-\beta_{j-1}^*}{k}$, which solves to

$$\begin{aligned}\beta_j^* &= (1+\epsilon)\cdot\left(1-\left(1-\frac{1}{k}\right)^{j-i-2}\right) + \left(1-\frac{1}{k}\right)^{j-i-1}\cdot(\beta_i+\alpha_i)\\ &= C_j' + (1-1/k)^{j-i-1}\cdot\alpha_i\ ,\end{aligned}$$

where C_j' is independent of α_i. It follows that

$$\begin{aligned}\beta_{i+1}^* + \sum_{j=i+2}^{d-1}\beta_j^* &= C'' + \alpha_i + \sum_{j=1}^{d-i-2}(1-1/k)^j\cdot\alpha_i\\ &= C'' + \left(1 + k\cdot\left(1-(1-1/k)^{d-i-2}\right)\cdot(1-1/k)\right)\cdot\alpha_i\ , \qquad \text{(B.4)}\end{aligned}$$

where C'' is independent of α_i. Since the coefficient of α_i in Eq. (B.4) is smaller than k, the coefficient of α_i in Eq. (B.3) is positive, and hence β_d^* is monotone in α_i. ■