Deterministically Counting Satisfying Assignments for Constant-Depth Circuits with Parity Gates, with Implications for Lower Bounds

Ninad Rajgopal[1]
Department of Computer Science, University of Oxford, Oxford, United Kingdom
ninad.rajgopal@cs.ox.ac.uk

Rahul Santhanam[2]
Department of Computer Science, University of Oxford, Oxford, United Kingdom
rahul.santhanam@cs.ox.ac.uk

Srikanth Srinivasan
Department of Mathematics, IIT Bombay, Mumbai, India
srikanth@math.iitb.ac.in

Abstract

We give a deterministic algorithm for counting the number of satisfying assignments of any $\mathrm{AC}^0[\oplus]$ circuit C of size s and depth d over n variables in time $2^{n-f(n,s,d)}$, where $f(n,s,d) = n/O(\log(s))^{d-1}$, whenever $s = 2^{o(n^{1/d})}$. As a consequence, we get that for each d, there is a language in E^{NP} that does not have $\mathrm{AC}^0[\oplus]$ circuits of size $2^{o(n^{1/(d+1)})}$. This is the first lower bound in E^{NP} against $\mathrm{AC}^0[\oplus]$ circuits that beats the lower bound of $2^{\Omega(n^{1/2(d-1)})}$ due to Razborov and Smolensky for large d. Both our algorithm and our lower bounds extend to $\mathrm{AC}^0[p]$ circuits for any prime p.

2012 ACM Subject Classification Theory of computation → Computational complexity and cryptography

Keywords and phrases circuit satisfiability, circuit lower bounds, polynomial method, derandomization

Digital Object Identifier 10.4230/LIPIcs.MFCS.2018.78

1 Introduction

In circuit complexity, we are interested in understanding the power and weaknesses of various circuit models. This understanding can take various forms for any given circuit class $\mathfrak{C}$. One indication of a deeper understanding is to be able to show *lower bounds* against $\mathfrak{C}$, i.e., prove that some "explicit" function cannot be computed by small circuits in $\mathfrak{C}$. Other indications come from efficient or at least non-trivial solutions for various *meta-algorithmic* tasks involving $\mathfrak{C}$, such as satisfiability algorithms for $\mathfrak{C}$, learning algorithms for $\mathfrak{C}$ or pseudo-random generators useful against $\mathfrak{C}$. There are some formal connections between lower bounds and efficient solvability of meta-algorithmic tasks for classes $\mathfrak{C}$ satisfying some natural closure properties, for example, an equivalence between pseudo-random generators against $\mathfrak{C}$ and average-case lower bounds in linear exponential time against $\mathfrak{C}$ [18], an implication from

[1] This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement No. 615075.
[2] This work was supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2014)/ERC Grant Agreement No. 615075.

non-trivial satisfiability algorithms for $\mathfrak{C}$ to lower bounds in non-deterministic exponential time against $\mathfrak{C}$ [26], and an implication from non-trivial learning algorithms for $\mathfrak{C}$ to lower bounds in probabilistic exponential time against $\mathfrak{C}$ [20].

For the class of Boolean circuits in general, our understanding is very limited in these terms. We have no super-linear lower bounds against general Boolean circuits for any explicit function, nor do we have non-trivial solutions for any of the meta-algorithmic tasks mentioned. The situation is far better for more restricted classes of circuits, especially circuits of *constant depth* where the gates have unbounded fan-in, both with respect to lower bounds and to meta-algorithmic questions.

In this paper, we focus on constant-depth circuits with AND, OR and PARITY gates (or more generally, AND, OR and MODp gates for some prime p). This is a class that has been intensively studied. Razborov [21] and Smolensky [23] showed super-polynomial size lower bounds against this class for very simple functions such as the MAJORITY function and the MODq function where q is a prime different from 2. As we discuss in Section 2, non-trivial randomized algorithms for deciding satisfiability of circuits from this class are implicit in [27, 16]. Recently, [5] gave quasi-polynomial time algorithms for learning $\mathrm{AC}^0[\oplus]$ circuits in the membership query model.

However, there are still several gaps in our understanding of these $\mathrm{AC}^0[\oplus]$ circuits. With regard to lower bounds, we still do not have *tight* lower bounds for functions like Majority or MODq. The Razborov-Smolensky approximation method yields size lower bounds of the form $2^{\Omega(n^{1/2(d-1)})}$ for Majority against $\mathrm{AC}^0[\oplus]$ circuits of depth d over n variables. The best known upper bound is $2^{\tilde{O}(n^{1/(d-1)})}$ using a standard divide-and-conquer strategy. Closing this quadratic gap in the exponent between upper and lower bounds has been a long-standing open question in the complexity theory of constant-depth circuits. Note that, in contrast, tight bounds are known up to constant factors in the exponent when only AND and OR gates are allowed - Parity is known to have complexity $2^{\Theta(n^{1/(d-1)})}$ in this simpler model [1, 9, 29, 11].

With regard to meta-algorithmic tasks, despite the learning breakthrough of [5] mentioned previously, the situation for pseudo-random generators (PRGs) and satisfiability algorithms is still unclear. While we have super-polynomial worst-case lower bounds against $\mathrm{AC}^0[\oplus]$ circuits, we do not have good average-case lower bounds, and as a consequence, do not have good PRGs. In the case of satisfiability algorithms, randomized algorithms improving non-trivially over brute force search are implicit in previous work, but good deterministic algorithms were unknown prior to this work.

Deterministic satisfiability algorithms are important for a couple of reasons. First, they indicate an improved structural understanding of the circuit class in question, often requiring new techniques to design. Second, they imply circuit lower bounds via the connection of Williams [26] - such an implication is not known from randomized algorithms.[3]

Note that even under standard derandomization assumptions, it is unclear how to get a non-trivial deterministic satisfiability algorithm from a non-trivial randomized satisfiability algorithm. The reason is that derandomization inherently incurs a quadratic slowdown. The deterministic simulation of a randomized algorithm running in time T will take time at least T^2 when a PRG is used to do the derandomization, as the range of the PRG will have size at least T. This quadratic slowdown is unaffordable in the parametric regime

[3] One-sided error randomized algorithms, and more generally, co-non-deterministic algorithms for satisfiability for the circuit class also imply lower bounds via the result of Williams [26]. However, it is easy to check that the previous randomized algorithms for $\mathrm{AC}^0[\oplus]$ were two-sided error algorithms.

under consideration here - we are interested in algorithms running in time $2^{n-g(n)}$, for some $g(n) = o(n)$. Hence the determinization procedure cannot be black-box, but must rather use refined structural information about the circuit class in question.

The main result of this paper is a deterministic algorithm for counting satisfying assignments to $\mathrm{AC}^0[\oplus]$ circuits that improves non-trivially over brute force search.

▶ **Theorem 1** (Main theorem). *The following holds for some absolute constant $\varepsilon_0 > 0$. There is a deterministic algorithm that given an $\mathrm{AC}^0[\oplus]$ circuit C over n variables such that C has depth at most d and size $s \leq 2^{(\varepsilon_0 n)^{1/d}}$, counts the number of satisfying assignments to C in time 2^{n-t} where $t = t(n, s, d) = n/O(\log s)^{d-1}$.*

▶ Remark. It is easy to generalize our results to work with $\mathrm{AC}^0[\mathrm{Mod}_p]$ circuits for any fixed prime p.

In terms of parameters, the savings over brute force search matches the savings in the randomized algorithm of [14] for AC^0 circuits when the circuit size is $n^{1+\Omega(1)}$. Thus any further improvement in savings in our result would give a corresponding improvement for AC^0 algorithms, and moreover via the connection of Williams [26] to circuit lower bounds, a strong improvement would give super-polynomial formula size lower bounds for non-deterministic exponential time.

A minor caveat is that we require the circuit size to be $2^{O(n^{1/d})}$ in Theorem 1, for technical reasons. One would expect that the analysis can be extended to circuit size up to $2^{n^{1/(d-1)}}$.

We use Theorem 1 to get better lower bounds against $\mathrm{AC}^0[\oplus]$ circuits than known before by using the connection of Williams [26]. In fact, we need a refinement of the connection due to [4]. Our lower bound holds for a language in E^{NP}. An intriguing open question is to use the more refined structural information about $\mathrm{AC}^0[\oplus]$ circuits exploited in the proof of Theorem 1 to prove a similar lower bound for more explicit problems or even for MAJORITY.

▶ **Theorem 2.** *For any positive integer d, there is a language in E^{NP} which does not have $\mathrm{AC}^0[\oplus]$ circuits of depth d and size $2^{o(n^{1/(d+1)})}$.*

2 Proof Outline for the Main Theorem

Our starting point is a *randomized* algorithm for the problem of checking satisfiability of an $\mathrm{AC}^0[\oplus]$ circuit C that runs in time 2^{n-m} where $m = n/O(\log s)^{d-1}$, and s, d represent the size, depth of C respectively (we also assume that s is suitably upper bounded, but we ignore it in this section). This algorithm is essentially due to Williams [27] and Lokshtanov, Paturi, Tamaki, Williams and Yu [16], though it does not appear explicitly in either of these papers.

The idea is to use a result of Razborov [21] that essentially says that small $\mathrm{AC}^0[\oplus]$ circuits C can be "approximated" by polynomials of small degree. More formally, there is a randomized algorithm that, when given a circuit C of size s over n variables, produces a (random) polynomial $\boldsymbol{P} \in \mathbb{F}_2[x_1, \ldots, x_n]$ of degree $O(\log s)^{d-1}$ that agrees with the value of a circuit C on any given input with good probability (say 0.9). Along with a fast polynomial evaluation algorithm [25], this immediately yields an *enumeration* algorithm for C (i.e. an algorithm to output the truth table of C) that runs in time $\mathrm{poly}(n)2^n + \mathrm{poly}(s)$, which beats (for large enough s) the trivial algorithm that simply evaluates C on each input and hence takes time $s \cdot 2^n$. Repeating the algorithm $\mathrm{poly}(n)$ times and taking the majority vote on each input, we get an enumeration algorithm that works with high probability.

To obtain a randomized satisfiability algorithm that runs in better-than-brute-force time, we use the above idea along with the "blowup-trick" [28, 6, 16]. For any $a \in \{0,1\}^m$, let C_a be the circuit obtained by setting the last m variables of C to a. Note that the satisfiability

of C can be computed by checking the satisfiability of $C' = \bigvee_{a\in\{0,1\}^m} C_a$, and C' is a circuit of larger size ($s \cdot 2^m$) but fewer variables ($n - m$). We now run the enumeration algorithm above on C' to check if it is satisfiable. Since the circuit is larger, the polynomial produced has larger degree: a careful analysis reveals the degree to be $m \cdot O(\log s)^{d-1}$. Setting $m = n/\Theta(\log s)^{d-1}$, we obtain a polynomial of degree $\ll n$, which can be computed in better-than-brute-force time. Running the enumeration algorithm as above gives the required satisfiability algorithm for C, which now runs in time $2^{n-n/\Theta(\log s)^{d-1}}$.

The above algorithm can further be modified to *count* satisfying assignments, by instead defining $C' = \sum_a C_a$ (a sum over $\mathbb{Z}$) instead of using an OR. Now, an additional idea is required since the polynomials $\boldsymbol{P}_a$ approximating each C_a are $\mathbb{F}_2$-polynomials whereas the sum is over the integers (in the satisfiability case above, the OR gate can further be approximated by a constant-degree polynomial using an idea of Razborov [21], but this idea is not available here). What comes to our rescue is an idea of Toda [24] and its subsequent quantitative refinement due to Beigel and Tarui [3] which tells us that we can simulate a sum (over $\mathbb{Z}$) of K many $\mathbb{F}_2$-polynomials of degree at most D as a polynomial (over $\mathbb{Z}$) of degree at most $D \log K$. Using this idea, we are able to obtain a polynomial of degree $m^2 \cdot O(\log s)^{d-1}$. Overall, this yields an algorithm with slightly worse running time $2^{n-\sqrt{n/\Theta(\log s)^{d-1}}}$.

A partial derandomization of these algorithms was obtained by Chan and Williams [6] in the case that the $\mathrm{AC}^0[\oplus]$ circuits are k-CNFs and generalized by Lokshtanov et al. [16] to the case of ANDs of degree-k polynomials.[4] Chan and Williams [6] observed that Razborov's random construction of polynomials could be suitably derandomized using *ε-biased spaces* [17]. Using this idea (and more work), it was shown that the number of satisfying assignments to a set of degree k-polynomials in n variables could be computed in time $2^{n-n/\Theta(k)}$, which meets the running time of the satisfiability algorithm mentioned above in this special case.

However, it is unclear how to extend the ideas to the setting of general $\mathrm{AC}^0[\oplus]$ circuits since these results used a very special property of the randomized polynomial construction for a single OR gate (and, dually, a single AND gate): namely, that there is a *constant-degree* polynomial whose bias perfectly predicts whether the input to an OR-gate is a 1 input or a 0 input.[5] Unfortunately, such a strong property is not known for general $\mathrm{AC}^0[\oplus]$ circuits: the best we can hope for is to construct a polynomial that with high probability, say $1-\varepsilon$, equals the output of the circuit on any given input, but then we have to pay for this precision in terms of the degree of the polynomial constructed. Further, it is not clear how to derandomize this general inductive construction.

We start by derandomizing the higher-depth random polynomial construction. Once again, ε-biased spaces play a crucial role, and we need to further use derandomized sampling using expanders for a near-optimal derandomization. Using this along with the idea of Beigel and Tarui [3] would yield a deterministic algorithm for counting satisfying assignments in time $2^{n-\sqrt{n/\Theta(\log s)^{d-1}}}$.

However, we further improve the running time to $2^{n-n/\Theta(\log s)^{d-1}}$, matching the running time of the randomized algorithm for checking satisfiability. The principal idea here is to observe (by looking inside the Razborov construction) that the polynomials $\boldsymbol{P}_a$ computed for approximating the individual circuits C_a mentioned above have a very special form: each $\boldsymbol{P}_a$

[4] Note that any k-clause is in particular a degree-k polynomial and hence the latter result generalizes the former.

[5] Briefly, the Razborov polynomial for the OR function on input bits $x_1, \ldots, x_s$ is as follows. Choose $a_1, \ldots, a_s \in \mathbb{F}_2$ independently and uniformly at random and compute $\boldsymbol{\ell}(x) = \sum_i a_i x_i$. Now, note that if $\mathrm{OR}(x_1, \ldots, x_s) = 1$, then $\ell(x)$ computes a uniformly random element of $\mathbb{F}_2$ and otherwise, $\ell(x) = 0$ with probability 1.

is a majority of $\ell = O(m)$ many polynomials $\boldsymbol{P}_{a,1}, \ldots, \boldsymbol{P}_{a,\ell}$ of degree $O(\log s)^{d-1}$ each. We use this and some basic Fourier analysis of Boolean functions to write $\boldsymbol{P}_a$ as a real-valued sum of polynomials of degree at most $O(\log s)^{d-1}$; this idea is inspired by a recent result of Chen and Papakonstantinou [7], who use it to give an improved depth-reduction result for constant-depth circuits with Mod_q gates for composite q. The advantage of doing this is that the degree blow-up in the randomized #SAT algorithm outline above is restricted to applying the idea of Beigel and Tarui, which means that the degree drops to $m \cdot O(\log s)^{d-1}$. Setting m suitably, we now obtain a deterministic algorithm running in time $2^{n-n/O(\log s)^{d-1}}$.

3 Preliminaries

We will consider polynomials over the fields $\mathbb{R}$ and $\mathbb{F}_2$. We identify $\mathbb{F}_2$ with $\{0,1\}$ in the natural way. We use $\binom{n}{\leq k}$ to denote $\sum_{i=0}^{k} \binom{n}{i}$. All algorithms will be implemented in the standard Turing machine with RAM model.

We recall that any function $f : \{0,1\}^n \to R$ for any commutative ring R has a unique representation as a multilinear polynomial. Given two such polynomials representing possibly different functions f_1 and f_2, we can compute the multilinear polynomial corresponding to their product by multiplying the polynomials f_1 and f_2 and "multilinearizing" by replacing each copy of x_i^2 by x_i. In particular, this idea yields the following easy algorithm.

▶ **Fact 3.** *Let R be either $\mathbb{F}_2$ or $\mathbb{Z}$. There is a deterministic algorithm which, when given as input multilinear polynomials $f_1, \ldots, f_t \in R[x_1, \ldots, x_n]$ (as a sum of monomials) of degree $d_1, \ldots, d_t$ such that $\sum_i d_i \leq D$, computes the multilinear polynomial corresponding to the product $f = f_1 \cdots f_t$. The algorithm runs in time $\mathrm{poly}(\binom{n}{\leq D})$ when $R = \mathbb{F}_2$ and time $\mathrm{poly}(\binom{n}{\leq D}, B)$ where B is the bit-complexity of the coefficients of $f_1, \ldots, f_t$ when $R = \mathbb{Z}$.*

3.1 Polynomials over $\mathbb{F}_2$ and Probabilistic polynomials

▶ **Definition 4** (Probabilistic Polynomials)**.** We recall [21, 23] that a *Probabilistic polynomial* from $\mathbb{F}_2[x_1, \ldots, x_n]$ is a random multilinear polynomial $\mathbf{P}$ (chosen according to some distribution) from $\mathbb{F}_2[x_1, \ldots, x_n]$. We say that $\mathbf{P}$ has degree at most D if the distribution of $\mathbf{P}$ is supported on polynomials of degree at most D (or equivalently $\Pr_{\mathbf{P}}[\deg(\mathbf{P}) \leq D] = 1$).

We say that $\mathbf{P}$ is an ε-error probabilistic polynomial for a function $f : \{0,1\}^n \to \{0,1\}$ if for each $a \in \{0,1\}^n$, we have $\Pr_{\mathbf{P}}[\mathbf{P}(a) \neq f(a)] \leq \varepsilon$.

3.2 Polynomials over $\mathbb{R}$ and Modulus-amplification

We recall the following basic facts about writing Boolean functions as multilinear polynomials over the the *reals*. See, e.g. O'Donnell [19] for proofs.

▶ **Fact 5.** *Let $f : \{0,1\}^\ell \to \{0,1\}$ be any Boolean function.*

1. *f can written as a unique* real-valued *linear combination*

$$f(x) = \sum_{S \subseteq [\ell]} \alpha_S \chi_S(x)$$

where $\chi_S(x) = \bigoplus_{i \in S} x_i$ (note that we interpret $\chi_S(x) \in \{0,1\}$ as a real number and the sum above is taken over $\mathbb{R}$).
2. *For each S, $\alpha_S \in [-1,1]$ and is moreover an integral multiple of $2^{-(\ell+1)}$.*
3. *There is a deterministic algorithm $\mathcal{C}$ which, given as input f (via its truth table), computes all the above α_S's in time $2^{O(\ell)}$.*

We also define the Fourier l_1 norm of f as $\|f\|_1 = \sum_S |\alpha_S|$.

The following is a useful *Modulus-amplification* lemma due to Beigel and Tarui [3]. This particular version is from the work of Chan and Williams [6].

▶ **Lemma 6** (Beigel-Tarui [3]). *For every positive integer t, the degree $2t-1$ polynomial $F_t(y) \in \mathbb{Z}[y]$ defined by*

$$F_t(y) = 1 - (1-y)^t \sum_{j=0}^{t-1} \binom{t+j-1}{j} y^j$$

has the property that for all $b \in \mathbb{Z}$,

- *if $b \equiv 0 \pmod 2$, then $F_t(b) \equiv 0 \pmod{2^t}$, and*
- *if $b \equiv 1 \pmod 2$, then $F_t(b) \equiv 1 \pmod{2^t}$.*

Many satisfiability algorithms for circuits are based on evaluating multivariate polynomials efficiently over grids. The following lemma can be found in, e.g., [25].

▶ **Lemma 7** (Fast Polynomial Evaluation). *There is a deterministic algorithm* FPE, *which given as input a multilinear polynomial $P \in \mathbb{Z}[x_1, \ldots, x_n]$ as a sum of monomials, computes the values $(P(a))_{a \in \{0,1\}^n}$ in time $\mathrm{poly}(n, B) \cdot 2^n$ where B is an upper bound on the bit complexity of the coefficients of P.*

3.3 Small-biased sets

We need the notion of *ε-biased sets* [17, 2], which are a standard tool in the derandomization literature.

▶ **Definition 8** (ε-biased sets [17]). For $\varepsilon \in \left(0, \frac{1}{2}\right)$, a set $S \subseteq \{0,1\}^n$ of n-dimensional vectors is ε-biased if for all non-zero $v \in \{0,1\}^n$,

$$\Pr_{w \in S}\{\langle v, w\rangle = 0 \pmod 2\} \in \left(\frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon\right)$$

There are many explicit constructions for ε-biased sets. We use the following construction:

▶ **Theorem 9** ([2]). *There is a deterministic algorithm that, given as input n and $\varepsilon \in (0, 1/2)$, produces an ε-biased set $S \subseteq \{0,1\}^n$ of size $O(n^2/\varepsilon^2)$. The algorithm runs in time* $\mathrm{poly}(n/\varepsilon)$.

For any subspace W of $\{0,1\}^n$, define the indicator function $\mathbf{1}_W : \{0,1\}^n \to \{0,1\}$ as $\mathbf{1}_W(z) = 1$ if and only if the vector $z \in W$. From an Observation in O'Donnell's book [19], we see that

▶ **Observation 10** (Proposition 3.11 of [19]). *Let W be a subspace of $\{0,1\}^n$ and $W_\perp$ be its orthogonal complement such that $dim(W_\perp) = k$. Then, the constant term in the Fourier expansion of the indicator function $\mathbf{1}_W$ is $\frac{1}{2^k}$. Moreover, $\|\mathbf{1}_W\|_1 = 1$.*

Essentially, the proof for Observation 10 is based on the fact that a vector $z \in W$ if and only if the dot product of z with every basis vector of $W_\perp$ is 0.

De, Etesami, Trevisan and Tulsiani [8] observed that ε-biased spaces also fool functions with small Fourier l_1-norm.

▶ **Lemma 11** (Lemma 2.5 of [8]). *Let S be an ε-biased set. For every function $f : \{0,1\}^n \to \mathbb{R}$ we have,*

$$\left| \mathop{\boldsymbol{E}}_{y \in S}[f(y)] - \mathop{\boldsymbol{E}}_{x \sim U_n}[f(x)] \right| \leq \varepsilon \|f\|_1$$

where y is picked uniformly at random from S and U_n is the uniform distribution over $\{0,1\}^n$.

From Observation 10 and Lemma 11, we state the following useful corollary.

▶ **Corollary 12.** *Let W be a subspace of $\{0,1\}^n$, such that the co-dimension of W is k and let S be an ε-biased set. Then*

$$\Pr_{x \in S}\{x \in W\} \in \left(\frac{1}{2^k} - \varepsilon, \frac{1}{2^k} + \varepsilon\right)$$

where x is picked uniformly at random from S.

Intuitively speaking, Corollary 12 states that an ε-biased set also "fools" conjunctions of parities.

3.4 Expanders

Proofs of the following well-known facts may be found in the monograph of Hoory, Linial and Wigderson [13].

Given an undirected Δ-regular multigraph G, we denote by $A(G)$ its adjacency matrix and $\tilde{A}(G) = (1/\Delta)A(G)$ its *normalized* adjacency matrix. Then, we have the following.

- The all-1s vector $v \in \mathbb{R}^n$ is an eigenvector of $\tilde{A}(G)$ with eigenvalue 1.
- G is connected if and only if v is the only such eigenvector (up to scalar multiplication).

▶ **Definition 13** (Expanders [13])**.** An undirected multigraph G is an (N, Δ, λ) *expander* if it is a Δ-regular connected graph on N vertices (which we will identify with $[N]$), and all the eigenvalues (counted with multiplicity) of $\tilde{A}(G)$ other than 1 are bounded by λ in absolute value.

Let $(G_n)_{n\geq 1}$ be a sequence of expander graphs with G_n being an $(f(n), \Delta, \lambda)$-expander for some increasing function $f : \mathbb{N} \to \mathbb{N}$ and constants Δ and λ. We say that $(G_n)_{n\geq 1}$ is *explicit* if there is a deterministic algorithm that, when given as input n, produces the graph G_n in time $\mathrm{poly}(n, f(n))$.

We use Reingold, Vadhan and Wigderson's [22] explicit construction for an expander graph.

▶ **Theorem 14** ([22])**.** *For every fixed $\lambda > 0$ there exists a constant $\Delta > 0$ and an explicit sequence of expander graphs $(G_n)_{n\geq 1}$, where G_n is a $(2^n, \Delta, \lambda)$ expander graph, for some large enough constant Δ. Further, we can assume that Δ is a power of 2.*

We will need the following expander-based Chernoff bound due to Gillman [10]. The version below is due to Healy [12].

▶ **Theorem 15** ([10, 12])**.** *Let G be an (N, D, λ)-graph and let $S \subseteq [N]$ be a subset of the vertices of G such that $|S| = \beta N$. Consider the natural ℓ-step random walk on G defined by choosing a uniformly random vertex $u_1 \in [N]$ and repeatedly choosing random neighbours $\ell - 1$ times to obtain a (random) sequence $(u_1, \ldots, u_\ell)$ of vertices of G. Let X_S denote the number of $i \in [\ell]$ such that $u_i \in S$. For any fixed $\rho \in (0,1)$, we have*

$$\Pr_{u_1,\ldots,u_\ell}[|X_S - \beta\ell| \geq \rho\ell] \leq 2\exp\left(-\frac{1}{4}\rho^2(1-\lambda)\ell\right).$$

4 The #SAT algorithm

In this section we prove Theorem 1. We start with a deterministic algorithmic version of a lemma of Razborov [21] regarding approximating $\mathrm{AC}^0[\oplus]$ circuits by low-degree polynomials. Using this version, we then state formally the #SAT algorithm and analyze it.

4.1 Derandomized construction of probabilistic polynomials for $\mathbf{AC^0}[\oplus]$

The following lemma is an algorithmic version of a result of Razborov [21] (see also Kopparty-Srinivasan [15] for the dependence on ε). It can be viewed as a derandomization of a randomized algorithm due to Williams [27].

▶ **Lemma 16.** *For any $\varepsilon > 0$, an $\mathrm{AC}^0[\oplus]$ circuit C over n variables of depth d and size at most s has an ε-error probabilistic polynomial $\mathbf{P}$ from $\mathbb{F}_2[x_1, \ldots, x_n]$ of degree at most $D = (O(\log s)^{d-1} \cdot \log(1/\varepsilon))$. Moreover $\mathbf{P} = \mathrm{Maj}(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$, where $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ are probabilistic polynomials of degree $D_1 = O(\log s)^{d-1}$ and $\ell = O(\log(1/\varepsilon))$.*

Moreover, there is a deterministic procedure $\mathcal{S}$, which when given as input the circuit C, the parameter ε, and a uniformly random Boolean string $\boldsymbol{\sigma}$ of length $r = O(\log(s/\varepsilon))$, produces a random sample of the polynomials $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ as sums of monomials. The procedure $\mathcal{S}$ runs in time $\mathrm{poly}\left(\ell, s, \binom{n}{\leq D_1}\right)$.

Before we go into the proof of Lemma 16, we prove a weaker version that will be useful in the proof of Lemma 16. This result is a higher-depth analogue of a result of Chan and Williams [6] which itself may be viewed as a derandomization of the construction of Razborov [21] for depth-1 circuits.

▶ **Lemma 17.** *For every $\mathrm{AC}^0[\oplus]$ circuit C over n variables of depth d and size s, there exists a probabilistic polynomial $\mathbf{P}'$ with error at most $\frac{1}{4}$ with degree at most $D_1 = O(\log s)^{d-1}$. Further, there is a deterministic algorithm $\mathcal{S}_1$ that produces a random sample of $\mathbf{P}'$ as a sum of monomials given as input s, C and $O(\log s)$ random bits. The algorithm $\mathcal{S}_1$ runs in time* $\mathrm{poly}\left(s, \binom{n}{\leq D_1}\right)$.

Proof. Let C be the circuit input to the sampling algorithm. Let $m = s\log(40s), \varepsilon = 1/(20s)$ and $S \subseteq \{0,1\}^m$ be an ε-biased set of size $\mathrm{poly}(s)$ given by Theorem 9. Fix an input $a \in \{0,1\}^n$.

Fix some enumeration $g_1, \ldots, g_s$ of all the gates of C. Let $h \in \{g_1, \ldots, g_s\}$ be the output gate of C and let $C_1, \ldots, C_r$ ($r \leq s$) be the depth-$(d-1)$ sub-circuits of C feeding into h. First we construct a probabilistic polynomial of degree $O(\log s)$ for each gate $g \neq h$ in the circuit, following which we construct a constant degree probabilistic polynomial for the gate h. Composing these polynomials together gives the probabilistic polynomial for the circuit C of degree $O(\log s)^{d-1}$.

Fix any gate $g \neq h$ in C. If g is a NOT gate or a $\oplus$ gate, we can easily get a polynomial P_g of degree 1 which always agrees with the function computed by g and needs no random bits. Therefore, let g be an OR gate in C (a dual construction works for AND gates). Let $g_{i_1}, \ldots, g_{i_k}$ be the gates that are inputs to g and $v_g \in \{0,1\}^s$ such that $v_g[i] = 1$ iff $i \in \{i_1, \ldots, i_k\}$ and $g_i = 1$ (on the fixed input a). Observe that g outputs 1 iff v_g is not the zero vector.

Let $t = \log(40s)$. Construct the vectors $u_1, \ldots, u_t \in \{0,1\}^m$, such that for each u_p, $1 \leq p \leq t$, we divide u_p into t blocks, such that the p^{th} block contains v_g and all the other bits of u_p are set to 0. The dimension of the vector space $V \subseteq \mathbb{F}_2^m$ spanned by $\{u_1, \ldots, u_t\}$ is equal to t if v_g is a non-zero vector. Let $\boldsymbol{y} \in S$ be picked uniformly at random from S. We split $\boldsymbol{y}$ into t blocks $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t$ of size s each. W.l.o.g. consider the vector u_1. The inner product $\langle u_1, \boldsymbol{y}\rangle$ which is exactly equal to the inner product $\langle v_g, \boldsymbol{y}_1\rangle$, can be calculated by hardwiring into a MOD_2 gate all the input gates of g for which the corresponding bit in $\boldsymbol{y}$ equals 1. In other words, the inner product $\langle u_1, \boldsymbol{y}\rangle$ is represented by the polynomial $q_1 = \sum_{j \in \{i_1, \ldots, i_k\}} g_j \cdot (\boldsymbol{y}_1)_j$ in the inputs $g_{i_1}, \ldots, g_{i_k}$. Repeating this construction for all t

vectors $u_1, \ldots, u_t$, we get polynomials $q_1^{\boldsymbol{y}}, \ldots, q_t^{\boldsymbol{y}}$. Finally, we compute the disjunction of these t terms using the polynomial $P_g^{\boldsymbol{y}} = 1 - \prod_{p=1}^{t}(1 - q_p^{\boldsymbol{y}})$, which is of degree $t = O(\log s)$ in the input variables of g.

We now analyze the behaviour of $P_g^{\boldsymbol{y}}$ on a fixed input to the gate g. If the gate g outputs 0, then for any $y \in S$, we get $\langle u_p, y\rangle = 0$ for every $1 \le p \le t$. In particular, $P_g^{\boldsymbol{y}}$ outputs 0 with probability 1. On the other hand, if g outputs 1, we see that $P_g^{\boldsymbol{y}}$ errs on this input iff for each $1 \le p \le t$, $q_p^{\boldsymbol{y}} = 0$. Let $V_\perp$ be the orthogonal complement to $V = \mathrm{span}(\{u_1, \ldots, u_t\})$. Note that the codimension of the vector space $V_\perp$ is t. We now use Corollary 12 to see that, if g outputs 1, a uniformly random element $\boldsymbol{y}$ from S belongs to $V_\perp$, or in other words, yields $\langle u_p, \boldsymbol{y}\rangle = 0$ for every $1 \le p \le t$, with probability at most $\frac{1}{2^t} + \varepsilon$. Thus, $P_g^{\boldsymbol{y}}$ disagrees with the output of g on *any fixed input* with probability at most $\frac{1}{2^t} + \varepsilon = \frac{1}{40s} + \frac{1}{20s} = \frac{3}{40s}$.

We pick a *single* $\boldsymbol{y}$ uniformly at random from S and then use it (for each OR and AND gate) to get the probabilistic polynomial $P_g^{\boldsymbol{y}}$ for each gate $g \neq h$ in C. For each $j \in [r]$, composing the polynomials representing the gates in C_j, we obtain a probabilistic polynomial $\mathbf{Q}_j$ of degree at most $O(\log s)^{d-1}$. Following this, we use the first $t_1 = 3$ blocks of $\boldsymbol{y}$ in a similar fashion to get a probabilistic polynomial $\mathbf{P}_h$ of degree $O(1)$ for the function computed by the output gate h with error at most $\frac{1}{2^{t_1}} + \varepsilon$. Now, for any input $a \in \{0,1\}^n$, $\mathbf{P}' = \mathbf{P}_h(\mathbf{Q}_1, \ldots, \mathbf{Q}_r)$ satisfies

$$\begin{aligned}
\Pr_{\mathbf{P}'}[C(a) \neq \mathbf{P}'(a)] &\le \Pr_{\mathbf{Q}_1,\ldots,\mathbf{Q}_r}[\exists j \in [r] : C_j(a) \neq \mathbf{Q}_j(a)] \\
&\quad + \Pr_{\mathbf{P_h}}[h(C_1(a), \ldots, C_r(a)) \neq \mathbf{P}_h(C_1(a), \ldots, C_r(a))] \\
&\le s \cdot \frac{3}{40s} + \frac{1}{2^{t_1}} + \varepsilon \\
&\le \frac{3}{40} + \frac{1}{8} + \frac{1}{20s} \le \frac{1}{4}
\end{aligned}$$

where the second inequality uses a union bound over the (at most s) gates g in C.

This gives us a probabilistic polynomial $\mathbf{P}'$ of degree $O(\log s)^{d-1}$ for the circuit C, with error at most $\frac{1}{4}$ and we need $O(\log |S|) = O(\log s)$ random bits to get this sample. This polynomial is multilinear and by expanding the monomials at each step we can ensure multilinearity of the intermediate polynomials. Using Fact 3 we see that the running time of the sampling algorithm is given by poly $\left(s, \binom{n}{\le D_1}\right)$. ◀

Proof of Lemma 16. Let $k = O(\log s)$ be the number of random bits needed by the algorithm $\mathcal{S}_1$ from Lemma 17. Consider an explicit $(2^k, \Delta, \lambda)$ expander graph G on $V = \{0,1\}^k$ given by Theorem 14, where Δ is a large enough constant given by the construction and $\lambda = \frac{1}{2}$. The graph G can be constructed in time poly$(2^k) =$ poly(s).

Let $\ell = 200 \log\left(\frac{2}{\varepsilon}\right)$. We define the algorithm $\mathcal{S}$ to be the following deterministic procedure which takes as input the circuit C, the parameter ε and a random string $\boldsymbol{\sigma}$ of length $r = k + (\ell - 1) \cdot \log \Delta$ and produces a random sample of the probabilistic polynomials $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$, where each of the $\mathbf{P}_i$ is an instantation of the probabilistic polynomial constructed in Lemma 17.

1. Perform a length ℓ random walk in G using the bits of $\boldsymbol{\sigma}$ to obtain $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_\ell \in V = \{0,1\}^k$. I.e. first choose $\boldsymbol{u}_1$ uniformly at random from V and for each $i \in \{2, \ldots, \ell\}$, let $\boldsymbol{u}_i$ be a random neighbour of $\boldsymbol{u}_{i-1}$ in the graph G.
2. For each $1 \le i \le \ell$, use $\boldsymbol{u}_i$ as the input string of random bits to algorithm $\mathcal{S}_1$ from Lemma 17 to obtain a random sample $\mathbf{P}_i$ of the 1/4-error probabilistic polynomial $\mathbf{P}'$ for C.

The number of random bits used by $\mathcal{S}$ to obtain a random sample of $\mathbf{P}_1, \ldots, \mathbf{P}_\ell$ is $r = O(k+\ell) = O(\log s + \log\left(\frac{1}{\varepsilon}\right)) = O\left(\log(s/\varepsilon)\right)$. At each step of the random walk we spend poly $\left(s, \binom{n}{\leq D_1}\right)$ time to get a sample and thus, the overall running time of $\mathcal{S}$ is poly $\left(\ell, s, \binom{n}{\leq D_1}\right)$.

Now, define the probabilistic polynomial $\mathbf{P} = \mathrm{Maj}(\mathbf{P}_1, \ldots, \mathbf{P}_\ell)$. Since, the majority of ℓ bits is a polynomial of degree at most ℓ, the degree of $\mathbf{P}$ is $O(\log s)^{d-1} \cdot \log(1/\varepsilon)$.

To show that $\mathbf{P}$ is a probabilistic polynomial with error ε for the circuit C, fix an input $a \in \{0,1\}^n$. For any $u \in V$, let P^u be the polynomial sampled by the algorithm $\mathcal{S}_1$ when given u as input. Let B be the set of vertices $u \in V$ such that $P^u(a) \neq C(a)$. Note that as $\mathcal{S}_1$ samples a $1/4$-error probabilistic polynomial for C, we must have $|B| \leq |V|/4$. Now, we have $\mathbf{P}(a) \neq C(a)$ iff a majority of the vertices on the random walk sampled by $\mathcal{S}$ belong to B. Using Theorem 15 we show that this event happens with a probability at most ε.

Let X_B be the random variable which denotes the number of $i \in [\ell]$ such that $i \in B$. Using Theorem 15 with the settings $\beta = \frac{|B|}{|V|} \leq \frac{1}{4}, \lambda = \frac{1}{2}, \rho = \frac{1}{4}$ and $\ell = 200\log(2/\varepsilon)$, we see that

$$\Pr_{u_1,\ldots,u_\ell}\left\{\left|X_B - \ell/4\right| > \ell/4\right\} \leq 2\exp\left(-\frac{1}{4}\rho^2(1-\lambda)\cdot 200\log\left(\frac{2}{\varepsilon}\right)\right) < 2\cdot 2^{-\log\left(\frac{2}{\varepsilon}\right)} < \varepsilon \quad \blacktriangleleft$$

4.2 The algorithm and its analysis

We begin by describing the #SAT algorithm $\mathcal{A}$.

Algorithm $\mathcal{A}$.

The algorithm $\mathcal{A}$ has the following desired input-output behaviour.

Input: An $\mathrm{AC}^0[\oplus]$ circuit C over n variables of size at most s and depth at most d. Recall that $s \leq 2^{(\varepsilon_0 n)^{1/d}}$ for some absolute constant $\varepsilon_0 > 0$ (to be chosen below). We assume $s \geq n$.

Desired Output: The number of satisfying assignments of C.

Notation. Let $m = \gamma n/(\log s)^{d-1}$ for a suitable absolute constant $\gamma > 0$ that will be fixed below. Let $\varepsilon = 1/2^{10m}$. For s and ε as defined above, choose $r = O(\log(s/\varepsilon))$ suitably so that the sampling algorithm $\mathcal{S}$ from Lemma 16 works as stated.

For each $\sigma \in \{0,1\}^r$, let $(P_1^\sigma, \ldots, P_\ell^\sigma)$ be the output of the algorithm $\mathcal{S}$ on string σ (note that the probabilistic polynomials $(\boldsymbol{P}_1, \ldots, \boldsymbol{P}_\ell)$ from Lemma 16 are exactly the polynomials $(P_1^{\boldsymbol{\sigma}}, \ldots, P_\ell^{\boldsymbol{\sigma}})$ for a uniformly random $\boldsymbol{\sigma}$ and hence $\boldsymbol{P} = \mathrm{Maj}(P_1^{\boldsymbol{\sigma}}, \ldots, P_\ell^{\boldsymbol{\sigma}})$).

For fixed $\sigma \in \{0,1\}^r$ and $c \in \{0,1\}^m$, let $P_i^{\sigma,c} \in \mathbb{F}_2[x_1, \ldots, x_{n-m}]$ be defined by $P_i^{\sigma,c} = P_i^\sigma(x_1, \ldots, x_{n-m}, c_1, \ldots, c_m)$ (i.e. the last m variables of P_i^σ are fixed to bits of c). Let $P^{\sigma,c} = \mathrm{Maj}(P_1^{\sigma,c}, \ldots, P_\ell^{\sigma,c})$.

For $S \subseteq [\ell]$, let $P_S^{\sigma,c} = \bigoplus_{i\in S} P_i^{\sigma,c}$. Let $Q_S^{\sigma,c}$ be the polynomial with integer coefficients obtained by treating the $\mathbb{F}_2$-coefficients of $P_S^{\sigma,c}$ as integers. Note that for each $b \in \{0,1\}^{n-m}$, $P_S^{\sigma,c}(b) \equiv Q_S^{\sigma,c}(b) \pmod 2$.

1. Using the algorithm $\mathcal{C}$ from Fact 5, compute[6] integers $k_S \in \{2^{-(\ell+1)}, \ldots, 2^{(\ell+1)}\}$ for each $S \subseteq [\ell]$ such that $\mathrm{Maj}(z_1, \ldots, z_\ell) = \frac{1}{2^{\ell+1}} \sum_{S\subseteq[\ell]} k_S \bigoplus_{i\in S} z_i$.

[6] There is actually an explicit description of the integers k_S (see, e.g., O'Donnell [19]) using which each k_S can each be computed in time poly(ℓ) as opposed to the $2^{O(\ell)}$ time taken by the algorithm $\mathcal{C}$. However, the algorithm we give here doesn't need this and works for any Boolean function in place of Maj.

2. For each $c \in \{0,1\}^m$, $\sigma \in \{0,1\}^r$ and $S \subseteq [\ell]$, construct (as a sum of monomials) the polynomial $Q_S^{\sigma,c}(x_1, \ldots, x_{n-m})$ using the algorithm $\mathcal{S}$ from Lemma 16.
3. Construct as a sum of monomials (Fact 3) the multilinear polynomial $R \in \mathbb{Z}[x_1, \ldots, x_{n-m}]$ defined by

$$R(x_1, \ldots, x_{n-m}) = \sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} \sum_{S \subseteq [\ell]} k_S \cdot F_t(Q_S^{\sigma,c}(x_1, \ldots, x_{n-m}))$$

   where F_t is the modulus amplifying polynomial given by Lemma 6 and $t = A \log(s/\varepsilon)$ for a large absolute constant $A > 0$ chosen below.
4. Evaluate $R(b)$ for each $b \in \{0,1\}^{n-m}$ using the algorithm FPE from Lemma 7.
5. Let $R_t(b) = R(b) \pmod{2^t} \in \{0, \ldots, 2^t - 1\}$. Output $\sum_{b \in \{0,1\}^{n-m}} [R_t(b)/2^{\ell+1+r}]$ where $[x]$ denotes the integer closest to x (if x is a half-integer, $[x]$ is defined arbitrarily).

Theorem 1 follows directly from Lemmas 18 and 19 below.

▶ **Lemma 18** (Running time). *For any constant $A > 0$, there exist constants $\gamma > 0$ and $\varepsilon_0 > 0$ such that the algorithm $\mathcal{A}$, on an input circuit C of depth at most d and size at most $s \leq 2^{(\varepsilon_0 n)^{1/d}}$, has running time* $\mathrm{poly}(s) \cdot 2^{n/10} + \mathrm{poly}(n) \cdot 2^{n-m}$.

Proof. We analyse the running time of $\mathcal{A}$ by looking at the running times for each of its individual steps. From Fact 5, we see that Step 1 of the algorithm takes $2^{O(\ell)}$ running time. Since $\ell = O(\log(1/\varepsilon)) = O(m) = O(n/(\log s)^{d-1}) = o(n)$, this step takes at most $2^{o(n)} < 2^{n/10}$ time to run.

For Step 2, we see that the running time is $2^{\ell+m+r} \cdot \mathrm{poly}\left(\ell, s, \binom{n}{\leq D_1}\right)$, as we construct $2^{\ell+m+r}$ many polynomials using the algorithm $\mathcal{S}$, each of which takes $\mathrm{poly}(\ell, s, \binom{n}{\leq D_1})$ time to construct, as seen in Lemma 16. For the parameters we pick, we see that $2^{\ell+m+r} = 2^{o(n)}$ and $\binom{n}{\leq D_1} \leq n^{D_1} = n^{O(\log s)^{d-1}} = 2^{O(n^{(d-1)/d} \log n)}$. Thus, Step 2 takes at most $2^{n/10} \cdot \mathrm{poly}(n, s)$ time.

Step 3 takes time $2^{\ell+m+r} \cdot \mathrm{poly}\left(\ell, \binom{n}{\leq 2tD_1}\right)$, as the modulus amplifying polynomial F_t blows the degree of the polynomial up by a factor of $(2t-1)$ and the number of monomials in the multilinear expansion of the polynomial $R(x_1, \ldots, x_{n-m})$ is $\mathrm{poly}\left(\ell, \binom{n}{\leq 2tD_1}\right)$. To upper bound this running time, let $c' > 0$ be a constant such that the degree parameter D_1 from Lemma 16 is at most $c' \cdot (\log s)^{d-1}$. Then the degree of the polynomial R is at most

$$\begin{aligned} 2tD_1 &\leq 2t \cdot c'(\log s)^{d-1} \\ &= 2A \log(s/\varepsilon) \cdot c'(\log s)^{d-1} \\ &= 2Ac'(\log s)^d + 20Amc'(\log s)^{d-1} \\ &= 2Ac'(\log s)^d + 20A\gamma c' n \end{aligned}$$

where we have used $\log(1/\varepsilon) = 10m$ and $m = \gamma n/(\log s)^{d-1}$.

Fix the constants $\varepsilon_0 = \left(\frac{1}{400Ac'}\right)$ and $\gamma = \frac{1}{4000Ac'}$. This ensures that the $2tD_1 \leq 0.01n$. From this we see that, $\binom{n}{\leq 2tD_1}$ is at most $\left(\frac{ne}{2tD_1}\right)^{2tD_1} \leq \left(\frac{ne}{0.01n}\right)^{0.01n} < 2^{0.09n}$ and we see that step 3 takes at most $2^{n/10}$ time.

Note that each k_S computed in Step 1 is an $(\ell+1)$-bit integer and hence, the bit complexity of the coefficients of R is at most $O(\ell + m + r) \leq n$. From Lemma 7, we see that Step 4 takes $2^{n-m}\,\mathrm{poly}(n)$ time and Step 5 runs in the same time trivially. Thus, the algorithm $\mathcal{A}$ takes a total of $\mathrm{poly}(n, s) \cdot 2^{n/10} + \mathrm{poly}(n) \cdot (2^{n-m})$ to run. ◀

▶ **Lemma 19** (Correctness). *Assume that $A > 0$ is chosen large enough so that $t > m + r + \ell + 10$. Then the algorithm $\mathcal{A}$ above outputs the number of satisfying assignments of C.*

Proof. We need to show that the algorithm $\mathcal{A}$ computes correctly the number of satisfying assignments of C. To this end, define the function C' on $n-m$ input bits by

$$C'(x_1, \ldots, x_{n-m}) = \sum_{c \in \{0,1\}^m} C(x_1, \ldots, x_{n-m}, c_1, \ldots, c_m)$$

where the sum is over $\mathbb{Z}$. It suffices to show that, for every $b \in \{0,1\}^{n-m}$, $[R_t(b)/2^{\ell+1+r}]$ is a correct estimate of $C'(b)$ since this implies that the number of satisfying assignments of C, which is equal to $\sum_{b \in \{0,1\}^{n-m}} C'(b)$, is computed correctly.

From the definition of $R_t(b)$, we see that for any $b \in \{0,1\}^{n-m}$

$$\begin{aligned} R_t(b) &= R(b) \pmod{2^t} \\ &= \left(\sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} \sum_{S \subseteq [\ell]} k_S \cdot F_t(Q_S^{\sigma,c}(b)) \right) \pmod{2^t} \\ &= \left(\sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} \sum_{S \subseteq [\ell]} k_S \cdot \left(F_t(Q_S^{\sigma,c}(b)) \pmod{2^t} \right) \right) \pmod{2^t} \end{aligned}$$

For every $\sigma \in \{0,1\}^r, c \in \{0,1\}^m, S \subseteq [\ell]$ and $b \in \{0,1\}^{n-m}$, we use the property of the modulus amplifying polynomial F_t from Lemma 6, to observe that $F_t(Q_S^{\sigma,c}(b)) \pmod{2^t} = P_S^{\sigma,c}(b) = \oplus_{i \in S} P_i^{\sigma,c}(b)$. This observation, along with Step 1 of the algorithm $\mathcal{A}$ implies that the sum $\sum_{S \subseteq [\ell]} k_S \cdot (F_t(Q_S^{\sigma,c}(b)) \pmod{2^t})$ is the same as $2^{\ell+1}\mathrm{Maj}(P_1^{\sigma,c}(b), \ldots, P_\ell^{\sigma,c}(b)) = 2^{\ell+1} P^{\sigma,c}(b)$. In other words,

$$R_t(b) = \left(2^{\ell+1} \sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} P^{\sigma,c}(b) \right) \pmod{2^t} = 2^{\ell+1} \sum_{c \in \{0,1\}^m} \sum_{\sigma \in \{0,1\}^r} P^{\sigma,c}(b)$$

where the last equality follows from the fact that $t > m + \ell + r + 10$.

Now, for every $b \in \{0,1\}^{n-m}$ and $c \in \{0,1\}^m$, we have

$$\sum_{\sigma \in \{0,1\}^r} P^{\sigma,c}(b) \begin{cases} \geq 2^r(1-\varepsilon) & \text{if } C(b,c) = 1 \\ \leq 2^r \varepsilon & \text{if } C(b,c) = 0 \end{cases}$$

where $\varepsilon = 2^{-10m}$. Since $C'(b) = \sum_{c \in \{0,1\}^m} C(b,c)$, we now have that for every $b \in \{0,1\}^{n-m}$,

$$\begin{aligned} R_t(b) &\geq 2^{\ell+1} \cdot 2^r(1-\varepsilon) C'(b), \text{ and} \\ R_t(b) &\leq 2^{\ell+1} \cdot (2^r(1-\varepsilon)C'(b) + \varepsilon(2^m - C'(b)) \cdot 2^r) \\ &\leq 2^{\ell+1} \cdot (2^r(1-\varepsilon)C'(b) + \varepsilon 2^m \cdot 2^r)\,. \end{aligned}$$

In particular, since $\varepsilon = 2^{-10m}$, for every $b \in \{0,1\}^{n-m}$, we see that the estimate returned by the algorithm, which is $[R_t(b)/2^{\ell+1+r}]$, is equal to $C'(b)$. ◀

4.3 A Consequence for Lower Bounds

Our #SAT algorithm can be used to obtain improved lower bounds against $\mathrm{AC}^0[\oplus]$ circuits (and more generally, against $\mathrm{AC}^0[p]$ circuits for prime p), using Williams' connection between algorithms and lower bounds. These lower bounds are, however, not very explicit - they hold for a language in E^{NP}.

We first remind the reader of the best explicit lower bounds that are known against $\mathrm{AC}^0[\oplus]$ circuits.

▶ **Theorem 20.** *[21, 23] For each positive integer d, there is a language computable in polynomial time that requires depth-d $\mathrm{AC}^0[\oplus]$ circuits of size $2^{\Omega(n^{1/2(d-1)})}$.*

In fact, there is a fixed explicit language in polynomial time, namely Majority, for which the lower bounds of Theorem 20 hold for all d.

In terms of parameters, the bound in Theorem 20 is weaker than the best known bound for AC^0 circuits as a function of d. Parity is known to require depth-d circuits of size $2^{\Omega(n^{1/(d-1)})}$. The exponent in the bound of Theorem 20 is quadratically smaller. It has been a longstanding open problem to improve the bound in Theorem 20 to match the known bounds for constant-depth circuits without prime modulus circuits. Using the algorithmic method of Williams and its refinements, we are able to use our #SAT algorithm to make progress on this problem.

The following lemma can be shown using the proof technique of Theorem 1.5 in [4].

▶ **Lemma 21.** *[26, 4] Let s be a size function and d a positive integer such that satisfiability can be solved deterministically in time $2^n/n^{\omega(1)}$ on $\mathrm{AC}^0[\oplus]$-circuits of size $O(s(n))$ and depth at most d on n variables. Then there is a language in E^{NP} which does not have $\mathrm{AC}^0[\oplus]$ circuits of depth $d-1$ and size $o(s(n))$.*

We now apply the lemma to get better lower bounds than Theorem 20 in terms of size against $\mathrm{AC}^0[\oplus]$ circuits when the depth is at least 3. A similar lower bound against $\mathrm{AC}^0[p]$ circuits can be shown for prime p using the analogue of Theorem 1 for $\mathrm{AC}^0[p]$ circuits. The following result is simply a re-statement of Theorem 2.

▶ **Theorem 22.** *For any positive integer d, there is a language in E^{NP} which does not have $\mathrm{AC}^0[\oplus]$ circuits of depth d and size $2^{o(n^{1/(d+1)})}$.*

Proof. Pick $\varepsilon < \varepsilon_0$, where ε_0 is the constant in Theorem 1. By Theorem 1, for any size $s \leq 2^{(\varepsilon n)^{1/(d+1)}}$, there is a deterministic algorithm solving satisfiability of $\mathrm{AC}^0[\oplus]$ circuits of size at most s and depth at most $d+1$ in time $2^n/n^{\omega(1)}$. Now using Lemma 21, we have that there is a language in E^{NP} which does not have $\mathrm{AC}^0[\oplus]$ circuits of depth d and size $o(s(n))$, which establishes our claim. ◀

References

1. Miklos Ajtai. Σ^1_1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24(1):1–48, 1983.
2. Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms*, 3(3):289–304, 1992.
3. Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.
4. Eli Ben-Sasson and Emanuele Viola. Short PCPs with projection queries. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, pages 163–173, 2014.
5. Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 10:1–10:24, 2016.
6. Timothy M. Chan and Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016.

7 Shiteng Chen and Periklis A. Papakonstantinou. Depth-reduction for composites. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 99–108. IEEE Computer Society, 2016. URL: `http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7781469`, `doi:10.1109/FOCS.2016.20`.

8 Anindya De, Omid Etesami, Luca Trevisan, and Madhur Tulsiani. Improved pseudorandom generators for depth 2 circuits. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 504–517. Springer, 2010.

9 Merrick Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

10 David Gillman. A Chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

11 Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Symposium on Theory of Computing* (STOC), pages 6–20, 1986.

12 Alexander D Healy. Randomness-efficient sampling within NC. *Computational Complexity*, 17(1):3–37, 2008.

13 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc. (N.S.)*, 43(4):439–561, 2006. `doi:10.1090/S0273-0979-06-01126-8`.

14 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972, 2012.

15 Swastik Kopparty and Srikanth Srinivasan. Certifying polynomials for $\mathrm{AC}^0[\oplus]$ circuits, with applications. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, pages 36–47, 2012.

16 Daniel Lokshtanov, Ramamohan Paturi, Suguru Tamaki, R. Ryan Williams, and Huacheng Yu. Beating brute force for systems of polynomial equations over finite fields. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2190–2202. SIAM, 2017. `doi:10.1137/1.9781611974782.143`.

17 Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

18 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

19 Ryan O'Donnell. *Analysis of Boolean functions*. Cambridge University Press, 2014.

20 Igor Carboni Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 18:1–18:49, 2017.

21 Alexander A Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

22 Omer Reingold, Salil Vadhan, and Avi Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. *Annals of Mathematics*, 155(1):157–187, 2002.

23 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82. ACM, 1987.

24 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 20(5):865–877, 1991. `doi:10.1137/0220053`.

25 Ryan Williams. Guest column: a casual tour around a circuit complexity bound. *SIGACT News*, 42(3):54–76, 2011. doi:10.1145/2034575.2034591.

26 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. *SIAM J. Comput.*, 42(3):1218–1244, 2013.

27 Ryan Williams. New algorithms and lower bounds for circuits with linear threshold gates. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 194–202, 2014.

28 Ryan Williams. Nonuniform ACC circuit lower bounds. *Journal of the ACM (JACM)*, 61(1):2, 2014.

29 Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *Symposium on Foundations of Computer Science* (FOCS), pages 1–10, 1985.