Quantum algorithms and approximating polynomials for composed functions with shared inputs

Mark Bun[1], Robin Kothari[2], and Justin Thaler[3]

[1]Boston University mbun@bu.edu

[2]Microsoft Quantum robin.kothari@microsoft.com

[3]Georgetown University justin.thaler@georgetown.edu

We give new quantum algorithms for evaluating composed functions whose inputs may be shared between bottom-level gates. Let f be an m-bit Boolean function and consider an n-bit function F obtained by applying f to conjunctions of possibly overlapping subsets of n variables. If f has quantum query complexity $Q(f)$, we give an algorithm for evaluating F using $\tilde{O}(\sqrt{Q(f)\cdot n})$ quantum queries. This improves on the bound of $O(Q(f)\cdot\sqrt{n})$ that follows by treating each conjunction independently, and our bound is tight for worst-case choices of f. Using completely different techniques, we prove a similar tight composition theorem for the approximate degree of f.

By recursively applying our composition theorems, we obtain a nearly optimal $\tilde{O}(n^{1-2^{-d}})$ upper bound on the quantum query complexity and approximate degree of linear-size depth-d AC^0 circuits. As a consequence, such circuits can be PAC learned in subexponential time, even in the challenging agnostic setting. Prior to our work, a subexponential-time algorithm was not known even for linear-size depth-3 AC^0 circuits.

As an additional consequence, we show that $\mathsf{AC}^0\circ\oplus$ circuits of depth $d+1$ require size $\widetilde{\Omega}(n^{1/(1-2^{-d})})\geq\omega(n^{1+2^{-d}})$ to compute the Inner Product function even on average. The previous best size lower bound was $\Omega(n^{1+4^{-(d+1)}})$ and only held in the worst case (Cheraghchi et al., JCSS 2018).

A preliminary version of this manuscript appeared in ACM-SIAM Symposium on Discrete Algorithms (SODA), 2019 [BKT19]. That version did not contain the lower bound for $\mathsf{AC}^0\circ\oplus$ circuits computing the Inner Product function.

1 Introduction

In the query, or black-box, model of computation, an algorithm aims to evaluate a known Boolean function $f : \{0,1\}^n \to \{0,1\}$ on an unknown input $x \in \{0,1\}^n$ by reading as few bits of x as possible. One of the most basic questions one can ask about query complexity, or indeed any complexity measure of Boolean functions, is how it behaves under *composition*. Namely, given functions f and g, and a method of combining these functions to produce a new function h, how does the query complexity of h depend on the complexities of the constituent functions f and g?

The simplest method for combining functions is *block composition*, where the inputs to f are obtained by applying the function g to independent sets of variables. That is, if $f : \{0,1\}^m \to \{0,1\}$ and $g : \{0,1\}^k \to \{0,1\}$, then the block composition $(f \circ g) : \{0,1\}^{m \cdot k} \to \{0,1\}$ is defined by $(f \circ g)(x_1, \ldots, x_m) = f(g(x_1), \ldots, g(x_m))$ where each x_i is a k-bit string. In most reasonable models of computation, one can evaluate $f \circ g$ by running an algorithm for f, and using an algorithm for g to compute the inputs to f as needed. Thus, the query complexity of $f \circ g$ is at most the product of the complexities of f and g.[1]

For many query models, including those capturing deterministic and quantum computation, this is known to be tight. In particular, letting $Q(f)$ denote the bounded-error quantum query complexity of a function f, it is known that $Q(f \circ g) = \Theta(Q(f) \cdot Q(g))$ for all Boolean functions f and g [HLŠ07, Rei11]. This result has the flavor of a direct sum theorem: When computing many copies of the function g (in this case, as many as are needed to generate the necessary inputs to f), one cannot do better than just computing each copy independently.

1.1 Quantum algorithms for shared-input compositions

While we have a complete understanding of the behavior of quantum query complexity under block composition, little is known for more general compositions. What is the quantum query complexity of a composed function where inputs to f are generated by applying g to overlapping sets of variables? We call these more general compositions *shared-input compositions*. Not only does answering this question serve as a natural next step for improving our understanding of quantum query complexity, but it may lead to more unified algorithms and lower bounds for specific functions of interest in quantum computing. Many of the functions that have played an influential role in the study of quantum query complexity can be naturally expressed as compositions of simple functions with shared inputs, including k-distinctness, k-sum, surjectivity, triangle finding, and graph collision.

In this work, we study shared-input compositions between an arbitrary function f and the function $g = \mathsf{AND}$. If $f : \{0,1\}^m \to \{0,1\}$, then we let $h : \{0,1\}^n \to \{0,1\}$ be any function obtained by generating each input to f as an AND over some subset of (possibly negated) variables from $x_1, \ldots, x_n$, as depicted in Figure 1.

Of course, one can compute the function h by ignoring the fact that the AND gates depend on shared inputs, and instead regard each gate as depending on its own set of copies of the input variables. Using the quantum query upper bound for block compositions, together with

[1] In some "reasonable models," such as those with bounded error, one must take care to ensure that errors in computing each copy of g do not propagate, but we elide these issues for this introduction. Addressing this concern typically adds at most a logarithmic overhead.

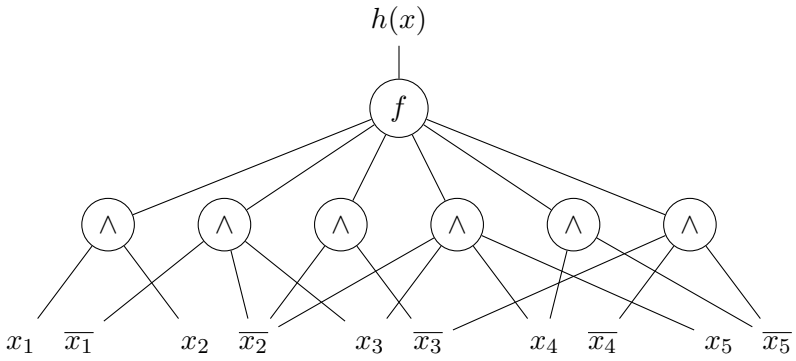

Figure 1: A depth-2 circuit $h : \{0,1\}^5 \to \{0,1\}$ where the top gate is a function $f : \{0,1\}^6 \to \{0,1\}$ and the bottom level gates are AND gates on a subset of the input bits and their negations. More generally, we consider $h : \{0,1\}^n \to \{0,1\}$, with top gate $f : \{0,1\}^m \to \{0,1\}$.

the fact that $Q(\mathsf{AND}_n) = \Theta(\sqrt{n})$ [Gro96, BBBV97], one obtains

$$Q(h) = O(Q(f) \cdot Q(\mathsf{AND}_n)) = O(Q(f) \cdot \sqrt{n}). \tag{1}$$

Observe that this bound on $Q(h)$ is non-trivial only if $Q(f) \ll \sqrt{n}$. A priori, one may conjecture that this bound is tight in the worst case for shared-input compositions. After all, if the variables overlap in some completely arbitrary way with no structure, it is unclear from the perspective of an algorithm designer how to use the values of already-computed AND gates to reduce the number of queries needed to compute further AND gates. It might even be the case that every pair of AND gates shares very few common input bits, suggesting that evaluating one AND gate yields almost no information about the output of any other AND gate. This intuition even suggests a path for proving a matching lower bound: Using a random wiring pattern, combinatorial designs, etc., construct the set of inputs to each AND gate so that evaluating any particular gate leaks almost no useful information that could be helpful in evaluating the other AND gates.

In this work, we show that this intuition is wrong: the overlapping structure of the AND gates can *always* be exploited algorithmically (so long as $Q(f) \ll n$).

Results. Our main result shows that a shared-input composition between a function f and the AND function always has substantially lower quantum query complexity than the block composition $f \circ \mathsf{AND}_n$. Specifically, instead of having quantum query complexity which is the product $Q(f) \cdot \sqrt{n}$, a shared-input composition has quantum query complexity which is, up to logarithmic factors, the geometric mean $\sqrt{Q(f) \cdot n}$ between $Q(f)$ and the number of input variables n. This bound is nontrivial whenever $Q(f)$ is significantly smaller than n.

Theorem 1. *Let $h : \{0,1\}^n \to \{0,1\}$ be computed by a depth-2 circuit where the top gate is a function $f : \{0,1\}^m \to \{0,1\}$ and the bottom level gates are* AND *gates on a subset of the input bits and their negations (as depicted in Figure 1). Then we have*

$$Q(h) = O\Big(\sqrt{Q(f) \cdot n} \cdot \log^2(mn)\Big). \tag{2}$$

Note that Theorem 1 is nearly tight for every possible value of $Q(f) \in [n]$.[2] For a parameter $t \leq n$, consider the block composition (i.e., the composition with disjoint inputs) $\mathsf{PARITY}_t \circ \mathsf{AND}_{n/t}$. Since $Q(\mathsf{PARITY}_t) = \lceil t/2 \rceil$ [BBC+01], this function has quantum query complexity

$$Q\left(\mathsf{PARITY}_t \circ \mathsf{AND}_{n/t}\right) = \Theta\left(t \cdot \sqrt{n/t}\right) = \Theta\left(\sqrt{Q(\mathsf{PARITY}_t) \cdot n}\right), \tag{3}$$

matching the upper bound provided by Theorem 1 up to log factors. This shows that Theorem 1 cannot be significantly improved in general.

The proof of Theorem 1 makes use of an optimal quantum algorithm for computing f and Grover's search algorithm for evaluating AND gates. Surprisingly, it uses no other tools from quantum computing. The core of the argument is entirely classical, relying on a recursive gate and wire-elimination argument for evaluating AND gates with overlapping inputs.

At a high level, the algorithm in Theorem 1 works as follows. The overall goal is to query enough input bits such that the resulting circuit is simple enough to apply the composition upper bound $Q(f \circ g) = O(Q(f)Q(g))$. To apply this upper bound and obtain the claimed upper bound in Theorem 1, we require $Q(g)$ to be $O(\sqrt{n/Q(f)})$. Since g is just an AND gate on some subset of inputs, this means we want the fan-in of each AND gate in our circuit to be $O(n/Q(f))$. If we call AND gates with fan-in $\omega(n/Q(f))$ "high fan-in" gates, then the goal is to eliminate all high fan-in gates. Our algorithm achieves this by judiciously querying input bits that would eliminate a large number of high fan-in gates if they were set to 0.

Besides the line of work on the quantum query complexity of block compositions, our result is also closely related to work of Childs, Kimmel, and Kothari [CKK12] on read-many formulas. Childs et al. showed that any formula on n inputs consisting of G gates from the de Morgan basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ can be evaluated using $O(G^{1/4} \cdot \sqrt{n})$ quantum queries. In the special case of DNF formulas, our result coincides with theirs by taking the top function f to be the OR function. However, even in this special case, the result of Childs et al. makes critical use of the top function being OR. Specifically, their result uses the fact that the quantum query complexity of the OR function is the square root of its formula size. Our result, on the other hand, applies without making any assumptions on the top function f. This level of generality is needed when using Theorem 1 to understand *circuits* (rather than just formulas) of depth 3 and higher, as discussed in Section 1.3.

1.2 Approximate degree of shared-input compositions

We also study shared-input compositions under the related notion of approximate degree. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, an *$\varepsilon$-approximating polynomial* for f is a real polynomial $p : \{0,1\}^n \to \mathbb{R}$ such that $|p(x) - f(x)| \leq \varepsilon$ for all $x \in \{0,1\}^n$. The *ε-approximate degree* of f, denoted $\deg_\varepsilon(f)$, is the least degree among all ε-approximating polynomials for f. We use the term *approximate degree* without qualification to refer to choice $\varepsilon = 1/3$, and denote it $\widetilde{\deg}(f) = \deg_{1/3}(f)$.

A fundamental observation due to Beals et al. [BBC+01] is that any T-query quantum algorithm for computing a function f implicitly defines a degree-$2T$ approximating polynomial

[2] Theorem 1 is not tight for every function f, of course. For example if f is an AND on many inputs, the composed function will have quantum query complexity $O(\sqrt{n})$ but the upper bound of Theorem 1 can be larger than this.

for f. Thus, $\widetilde{\deg}(f) \leq 2Q(f)$. This relationship has led to a number of successes in proving quantum query complexity lower bounds via approximate degree lower bounds, constituting a technique known as the polynomial method in quantum computing. Conversely, quantum algorithms are powerful tools for establishing the existence of low-degree approximating polynomials that are needed in other applications to theoretical computer science. For example, the deep result that every de Morgan formula of size s has quantum query complexity, and hence approximate degree, $O(\sqrt{s})$ [FGG08, CCJYM09, ACR+10, Rei11] underlies the fastest known algorithm for agnostically learning formulas [KKMS08, Rei11] (See Section 1.4 and Section 5 for details on this application). It has also played a major role in the proofs of the strongest formula and graph complexity lower bounds for explicit functions [Tal17].

Results. We complement our result on the quantum query complexity of shared-input compositions with an analogous result for approximate degree.

Theorem 2. *Let $h : \{0,1\}^n \to \{0,1\}$ be computed by a depth-2 circuit where the top gate is a function $f : \{0,1\}^m \to \{0,1\}$ and the bottom level gates are* AND *gates on a subset of the input bits and their negations (as depicted in Figure 1). Then*

$$\deg_\varepsilon(h) = O\left(\sqrt{\deg_\varepsilon(f) \cdot n \cdot \log m} + \sqrt{n \log(1/\varepsilon)}\right). \tag{4}$$

In particular, $\widetilde{\deg}(h) = O\left(\sqrt{\widetilde{\deg}(f) \cdot n \log m}\right)$.

Note that our result for approximate degree is incomparable with Theorem 1, even for bounded error, since both sides of the equation include the complexity measure under consideration.

Like Theorem 1, Theorem 2 can be shown to be tight by considering the block composition of PARITY with AND, since $\widetilde{\deg}(\mathsf{PARITY}_t \circ \mathsf{AND}_{n/t}) = \Theta\left(\sqrt{\widetilde{\deg}(\mathsf{PARITY}_t) \cdot n}\right)$ [She13b, She11b].

Our proof of Theorem 2 abstracts and generalizes a technique introduced by Sherstov [She18], who very recently proved an $O(n^{3/4})$ upper bound on the approximate degree of an important depth-3 circuit of nearly quadratic size called Surjectivity [She18]. Despite the similarity between Theorem 2 and Theorem 1, and the close connection between approximating polynomials and quantum algorithms, the proof of Theorem 2 is completely different from Theorem 1, making crucial use of properties of polynomials that do not hold for quantum algorithms.[3] In our opinion, this feature of the proof of Theorem 2 makes Theorem 1 for quantum algorithms even more surprising.

We remark that a different proof of the $O(n^{3/4})$ upper bound for the approximate degree of Surjectivity was discovered in [BKT18], who also showed a matching lower bound. It is also possible to prove Theorem 2 by generalizing the techniques developed in that work, but the techniques of [She18] lead to a shorter and cleaner analysis.

[3] Any analysis capable of yielding a sublinear upper bound on the approximate degree of Surjectivity requires moving beyond quantum algorithms, as its quantum query complexity is known to be $\Omega(n)$ [BM12, She15].

1.3 Application: Evaluating and approximating linear-size AC^0 circuits

The circuit class AC^0 consists of constant-depth, polynomial-size circuits over the de Morgan basis $\{\mathsf{AND}, \mathsf{OR}, \mathsf{NOT}\}$ with unbounded fan-in gates. The full class AC^0 is known to contain very hard functions from the standpoint of both quantum query complexity and approximate degree. The aforementioned Surjectivity function is in depth-3 AC^0 and has quantum query complexity $\Omega(n)$ [BM12, She15], while for every positive constant $\delta > 0$, there exists a depth-$O(\log(1/\delta))$ AC^0 circuit with approximate degree $\Omega(n^{1-\delta})$ [BT17].

Nevertheless, AC^0 contains a number of interesting subclasses for which nontrivial quantum query and approximate degree upper bounds might still hold. Here, we discuss applications of our composition theorem to understanding the subclass LC^0, consisting of AC^0 circuits of linear size.

The class LC^0 is one of the most interesting subclasses of AC^0. It has been studied by many authors in various complexity-theoretic contexts, ranging from logical characterizations [KLPT06] to faster-than-brute-force satisfiability algorithms [CIP09, SS12]. LC^0 turns out to be a surprisingly powerful class. For example, the k-threshold function that asks if the input has Hamming weight greater than k is clearly in AC^0 for constant k, by computing the OR of all $\binom{n}{k}$ possible certificates. But this yields a circuit of size $O(n^k)$, which one might conjecture is optimal. However, it turns out that k-threshold is in LC^0 even when k is as large as polylog(n) [RW91]. Another surprising fact is that every regular language in AC^0 can be computed by an AC^0 circuit of almost linear size (e.g., size $O(n \log^* n)$ suffices) [Kou09].

By recursively applying Theorem 1, we obtain the following sublinear upper bound on the quantum query complexity of depth-d LC^0 circuits, denoted by LC^0_d:

Theorem 3. *For all constants $d \geq 0$ and all functions $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d, we have $Q(h) = \widetilde{O}(n^{1-2^{-d}})$.*

Our upper bound is nearly tight for every depth d, as shown in [CKK12].

Theorem 4 (Childs, Kimmel, and Kothari)**.** *For all constants $d \geq 0$, there exists a function $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d with $Q(h) \geq n^{1-2^{-\Omega(d)}}$.*

By recursively applying Theorem 2, we obtain a similar sublinear upper bound for the ε-approximate degree of LC^0_d, even for subconstant values of ε.

Theorem 5. *For all constant $d \geq 0$, and any $\varepsilon > 0$, and all functions $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d, we have*

$$\deg_\varepsilon(h) = \widetilde{O}\Big(n^{1-2^{-d}} \log^{2^{-d}}(1/\varepsilon)\Big). \tag{5}$$

For constant ϵ, we prove a lower bound of the same form with quadratically worse dependence on the depth d.

Theorem 6. *For all constants $d \geq 0$, there exists a function $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d with $\widetilde{\deg}(h) \geq n^{1-2^{-\Omega(\sqrt{d})}}$.*

A lower bound of $\widetilde{\deg}(h) = n^{1-2^{-\Omega(d)}}$ was already known for general AC^0 functions f [BT17, BKT18], but the AC^0 circuits constructed in these prior works are not of linear size. Previously, for any $\ell \geq 1$, [BKT18] exhibited a circuit $C\colon \{0,1\}^n \to \{0,1\}$ of depth at most 3ℓ,

size at most n^2, and approximate degree $\widetilde{\deg}(C) \geq \widetilde{\Omega}(n^{1-2^{-\ell}})$. We show how to transform this quadratic-size circuit C into a linear-size circuit C of depth roughly ℓ^2, whose approximate degree is close to that of C. Our transformation adapts that of [CKK12], but requires a more intricate construction and analysis. This is because, unlike quantum query complexity, approximate degree is not known to increase multiplicatively under block composition.

1.4 Application: Agnostically learning linear-size AC^0 circuits

The challenging agnostic model [KSS94] of computational learning theory captures the task of binary classification in the presence of adversarial noise. In this model, a learning algorithm is given a sequence of labeled examples of the form $(x, b) \in \{0,1\}^n \times \{0,1\}$ drawn from an unknown distribution $\mathcal{D}$. The goal of the algorithm is to learn a hypothesis $h : \{0,1\}^n \to \{0,1\}$ which does "almost as well" at predicting the labels of new examples drawn from $\mathcal{D}$ as does the the best classifier from a known concept class $\mathcal{C}$. Specifically, let the Boolean loss of a hypothesis h be $\mathsf{err}_{\mathcal{D}}(h) = \Pr_{(x,b)\sim\mathcal{D}}[h(x) \neq b]$. For a given accuracy parameter ε, the goal of the learner is to produce a hypothesis h such that $\mathsf{err}_{\mathcal{D}}(h) \leq \min_{c\in\mathcal{C}} \mathsf{err}_{\mathcal{D}}(c) + \varepsilon$.

Very few concept classes $\mathcal{C}$ are known to be agnostically learnable, even in subexponential time. For example, the best known algorithm for agnostically learning disjunctions runs in time $2^{\tilde{O}(\sqrt{n})}$ [KKMS08].[4] Moreover, several hardness results are known. Proper agnostic learning of disjunctions (where the output hypothesis itself must be a disjunction) is NP-hard [KSS94]. Even improper agnostic learning of disjunctions is at least as hard as PAC learning DNF [LBW95], which is a longstanding open question in learning theory.

The best known general result for more expressive classes of circuits is that all de Morgan *formulas* of size s can be learned in time $2^{\tilde{O}(\sqrt{s})}$ [KKMS08, Rei11] (Section 5.1 contains a detailed overview of prior work on agnostic and PAC learning). Both of the aforementioned results make use of the well-known *linear regression* framework of [KKMS08] for agnostic learning. This algorithm works whenever there is a "small" set of "features" $\mathcal{F}$ (where each feature is a function mapping $\{0,1\}^n$ to $\mathbb{R}$) such that each concept in the concept class $\mathcal{C}$ can be approximated to error ε in the ℓ_∞ norm by a linear combination of features in $\mathcal{F}$. (See Section 5 for details.) If every function in a concept class $\mathcal{C}$ has approximate degree at most d, then one obtains an agnostic learning algorithm for $\mathcal{C}$ with running time $2^{\tilde{O}(d)}$ by taking $\mathcal{F}$ to be the set of all monomials of degree at most d. Applying this algorithm using the approximate degree upper bound of Theorem 5 yields a subexponential time algorithm for agnostically learning LC^0_d.

Theorem 7. *The concept class of n-bit functions computed by LC^0 circuits of depth d can be learned in the distribution-free agnostic PAC model in time $2^{\widetilde{O}(n^{1-2^{-d}})}$. More generally, size-$s$ AC^0_d circuits can be learned in time $2^{\widetilde{O}(\sqrt{n}s^{1/2-2^{-d}})}$.*

Prior to our work, no subexponential time algorithm was known even for agnostically learning LC^0_3. Moreover, since our upper bound on the approximate degree of LC^0 circuits is nearly tight, new techniques will be needed to significantly surpass our results, and in particular, learn *all* of LC^0 in subexponential time. (Note that standard techniques [She11a] automatically generalize the lower bound of Theorem 6 from the feature set of low-degree monomials to *arbitrary feature sets*. See Section 5.2 for details.)

[4] Throughout this manuscript, $\tilde{O}$ and $\tilde{\Omega}$ notation hides factors polylogarithmic in the input size n.

1.5 Application: New Circuit Lower Bounds

An important frontier problem in circuit complexity is to show that the well-known Inner Product function cannot be computed by $\mathsf{AC}^0 \circ \oplus$ circuits of polynomial size. Here, $\mathsf{AC}^0 \circ \oplus$ refers to AC^0 circuits augmented with a layer of parity gates at the bottom (i.e., closest to the inputs). Servedio and Viola [SV12] identified this open problem as a first step toward proving matrix rigidity lower bounds, itself a notorious open problem in complexity theory, and Akavia et al. [ABG+14] connected the problem to the goal of constructing highly efficient pseudorandom generators.[5] Average-case versions of this question have also been posed, even just for DNFs with a layer of parity gates at the bottom [CS16, ER21]. Unfortunately, the best known lower bounds against $\mathsf{AC}^0 \circ \oplus$ circuits computing Inner Product are quite weak. The state of the art result [CGJ+16] for any constant depth $d > 4$ is that Inner Product cannot be computed by any depth-$(d+1)$ $\mathsf{AC}^0 \circ \oplus$ circuit of size $O(n^{1+4^{-(d+1)}})$. We show that Theorem 5 implies an improved (if still unsatisfying) lower bound of $\widetilde{\Omega}(n^{1/(1-2^{-d})}) = n^{1+2^{-d}+\Omega(1)}$. More significantly, unlike prior work our lower bound holds even against circuits that compute the Inner Product function on slightly more than half of all inputs. Below, when we refer to the depth of an $\mathsf{AC}^0 \circ \oplus$ circuit, we count the layer of parity gates toward the depth. For example, we consider a DNF of parities to have depth 3.

Theorem 8. *For any constant integer $d \geq 4$, any depth-$(d+1)$ $\mathsf{AC}^0 \circ \oplus$ circuit computing the Inner Product function on n bits on greater than a $1/2 + n^{-\log n}$ fraction of inputs has size $\widetilde{\Omega}(n^{1/(1-2^{-d})}) = n^{1+2^{-d}+\Omega(1)}$.*

This application is new and does not appear in the conference version of this paper [BKT19]. The idea of our proof is to use the approximate degree upper bound for LC^0_d circuits of Theorem 5 to show that any small $\mathsf{AC}^0 \circ \oplus$ circuit has non-trivial (i.e., $\gg 2^{-n}$) correlation under the uniform distribution with some parity function. Yet it is well-known that the Inner Product function has correlation at most 2^{-n} with any parity function. As we show, this rules out the possibility that a small $\mathsf{AC}^0 \circ \oplus$ circuit computes the Inner Product function, even on slightly more than half of all inputs.

1.6 Discussion and future directions

Summarizing our results, we established shared-input composition theorems for quantum query complexity (Theorem 1) and approximate degree (Theorem 2), roughly showing that for compositions between an arbitrary function f and the function $g = \mathsf{AND}$, it is always possible to leverage sharing of inputs to obtain algorithmic speedups. We applied these results to obtain the first sublinear upper bounds on the quantum query complexity and approximate degree of LC^0_d.

Generalizing our composition theorems. Although considering the inner function $g = \mathsf{AND}$ is sufficient for our applications to LC^0, an important open question is to generalize our results to larger classes of inner functions. The proof of our composition theorem for approximate degree actually applies to any inner function g that can be exactly represented

[5] Superpolynomial lower bounds are known for $\mathsf{AC}^0 \circ \oplus$ circuits computing the Majority function [Raz87] (in fact, even for $\mathsf{AC}^0[2]$ circuits, which are AC^0 circuits augmented with parity gates at any layer). However, these techniques do not apply to the Inner Product function, which does have small $\mathsf{AC}^0[2]$ circuits.

as a low-weight sum of ANDs (for example, it applies to any strongly unbalanced function g, meaning that $|g^{-1}(1)| = \text{poly}(n)$). Extending this further would be a major step forward in our understanding of how quantum query complexity and approximate degree behave under composition with shared inputs.

While our paper considers the composition scenario where the top function is arbitrary and the bottom function is AND, the opposite scenario is also interesting. Here the top function is AND_m and the bottom functions are $f_1, \ldots, f_m$, each acting on the same set of n input variables. Now the question is whether we can do better than the upper bound obtained using results on block composition that treat all the input variables as being independent. More concretely, for such a function F, the upper bound that follows from block composition is $Q(F) = O(\sqrt{m} \max_i Q(f_i))$. However, this upper bound cannot be improved in general, because the Surjectivity function is an example of such a function. Here the bottom functions f_i check if the input contains a particular range element i, and the upper bound obtained from this argument is $O(n)$, which matches the lower bound [BM12, She15]. Surprisingly, this lower bound only holds for quantum query complexity, as we know that the approximate degree of Surjectivity is $\tilde{\Theta}(n^{3/4})$. We do not know if the upper bound obtained from block composition can be improved for approximate degree.

Quantum query complexity of LC^0 and DNFs. For quantum query complexity, we obtain the upper bound $Q(\mathsf{LC}^0_d) = \tilde{O}(n^{1-2^{-d}})$, nearly matching the lower bound $Q(\mathsf{LC}^0_d) = n^{1-2^{-\Omega(d)}}$ from [CKK12]. However, the bounds do not match for any fixed value of d. The lack of matching lower bounds can be attributed to the fact that the Surjectivity function, which is known to have linear quantum query complexity, is computed by a quadratic-size depth-3 circuit, rather than a quadratic-size depth-2 circuit (i.e., a DNF). If one could prove a linear lower bound on the quantum query complexity of some quadratic-size DNF, the argument of [CKK12] would translate this into a $\tilde{\Omega}(n^{1-2^{-d}})$ lower bound for LC^0_d, matching our upper bound. Unfortunately, no linear lower bound on the quantum query complexity of *any* polynomial size DNFs is known; we highlight this as an important open problem (the same problem was previously been posed by Troy Lee with different motivations [Lee12]).

Open Problem 1. Is there a polynomial-size DNF with $\widetilde{\Omega}(n)$ quantum query complexity?

The quantum query complexity of depth-2 LC^0, or linear-size DNFs also remains open. The best upper bound is $O(n^{3/4})$, but the best lower bound is $\Omega(n^{0.555})$ [CKK12]. Any improvement in the lower bound would also imply, in a black-box way, an improved lower bound for the Boolean matrix product verification problem. Improving the lower bound all the way to $\Omega(n^{3/4})$ would imply optimal lower bounds for all of LC^0 using the argument in [CKK12]. We conjecture that there is a linear-size DNF with quantum query complexity $\Omega(n^{3/4})$, matching the known upper bound.

Approximate degree of LC^0 and DNFs. For approximate degree, we obtain the upper bound $\widetilde{\deg}(\mathsf{LC}^0_d) = \tilde{O}(n^{1-2^{-d}})$, and prove a new lower bound of $\widetilde{\deg}(\mathsf{LC}^0_d) = n^{1-2^{-\Omega(\sqrt{d})}}$. The reason our approximate degree lower bound approaches n more slowly than the quantum query lower bound from [CKK12] is that, while the quantum query complexity of AC^0 is known to be $\Omega(n)$, such a result is not known for approximate degree. This remains an important open problem.

Open Problem 2. Is there a problem in AC^0 with approximate degree $\widetilde{\Omega}(n)$?

Our lower bound argument would translate, in a black-box manner, any linear lower bound on the approximate degree of a general AC^0 circuit into a nearly tight lower bound for LC^0_d.

Alternatively, it would be very interesting if one could improve our approximate degree upper bound for LC^0_d. Even seemingly small improvements to our upper bound would have significant implications. Specifically, standard techniques (see, e.g., [CR96]) imply that for any constant $\delta > 0$, there are approximate majority functions[6] computable by depth-$(2d+3)$ circuits of size $O(n^{1+2^{-d}+\delta})$.[7] This means that, for sufficiently large constant d, if one could improve our upper bound on the approximate degree of LC^0_d from $\widetilde{O}(n^{1-2^{-d}})$ to $\widetilde{O}(n^{1-2^{-d/2.001}})$, one would obtain a sublinear upper bound on the approximate degree of some total function computing an approximate majority. This would answer a question of Srinivasan [FHH+14], and may be considered a surprising result, as approximate majorities are currently the primary natural candidate AC^0 functions that may exhibit linear approximate degree [BKT18].

1.7 Paper organization and notation

This paper is organized so as to be accessible to readers without familiarity with quantum algorithms. Section 2 assumes the reader is somewhat familiar with quantum query complexity and Grover's algorithm [Gro96], but only uses Grover's algorithm as a black box. In Section 2 we show our main result on the quantum query complexity of shared-input compositions (Theorem 1). Section 3 proves our result about the approximate degree of shared-input compositions (Theorem 2). Section 4 uses the results of these sections (in a black-box manner) to upper bound the quantum query complexity and approximate degree of LC^0 circuits, and proves related lower bounds. Section 5 uses the results of Section 4 to obtain algorithms to agnostically PAC learn LC^0 circuits. Section 6 derives our average-case lower bounds on the size of $\mathsf{AC}^0 \circ \oplus$ circuits computing the Inner Product function. This section is new and does not appear in the conference version of this paper [BKT19].

In this paper we use the $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ notation to suppress logarithmic factors. More formally, $f(n) = \widetilde{O}(g(n))$ means there exists a constant k such that $f(n) = O(g(n)\log^k g(n))$, and similarly $f(n) = \widetilde{\Omega}(g(n))$ means there exists a constant k such that $f(n) = \Omega(g(n)/\log^k g(n))$. For a string $x \in \{0,1\}^n$, we use $|x| = \sum_i x_i$ to denote the Hamming weight of x, i.e., the number of entries in x equal to 1. For any positive integer n, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Given two functions f_m, g_k, let $f_m \circ g_k \colon \{0,1\}^{m\cdot k} \to \{0,1\}$ denote their *block composition*, i.e., $(f_m \circ g_k)(x) = f_m(g_k(x_1), \ldots, g_k(x_m))$, where for every $i \in [m]$, x_i is a k-bit string. For non-negative integers n and k, we use $\binom{n}{\leq k}$ to denote $\sum_{i=0}^{k}\binom{n}{i}$. A basic fact is that $\binom{n}{\leq k} \leq n^k$.

[6] Here, by an approximate majority function, we mean any total function f on n bits for which there exist constants $0 < p < 1/2 < q$ such that $|x| \leq pn \implies f(x) = 0$ and $|x| \geq qn \implies f(x) = 1$.

[7] This precise result has not appeared in the literature; we prove it in Appendix A for completeness.

2 Quantum algorithm for composed functions

2.1 Preliminaries

As described in the introduction, our quantum algorithm only uses variants of Grover's algorithm [Gro96] and is otherwise classical. To make this section accessible to those without familiarity with quantum query complexity, we only state the minimum required preliminaries to understand the algorithm. Furthermore, we do not optimize the logarithmic factors in our upper bound to simplify the presentation. For a more comprehensive introduction to quantum query complexity, we refer the reader to the survey by Buhrman and de Wolf [BdW02].

In quantum or classical query complexity, the goal is to compute some known function $f : \{0,1\}^n \to \{0,1\}$ on some unknown input $x \in \{0,1\}^n$ while reading as few bits of x as possible. Reading a bit of x is also referred to as "querying" a bit of x, and hence the goal is to minimize the number of queries made to the input.

For example, the deterministic query complexity of a function f is the minimum number of queries needed by a deterministic algorithm in the worst case. A deterministic algorithm must be correct on all inputs, and can decide which bit to query next based on the input bits it has seen so far. Another example of a query model is the bounded-error randomized query model. The bounded-error randomized query complexity of a function f, denoted $R(f)$, is the minimum number of queries made by a randomized algorithm that computes the function correctly with probability greater than or equal to $2/3$ on each input. In contrast to a deterministic algorithm, such an algorithm has access to a source of randomness, which it may use in deciding which bits to query.

The bounded-error quantum query complexity of f, denoted $Q(f)$, is similar to bounded-error randomized query complexity, except that the algorithm is now quantum. In particular, this means the algorithm may query the inputs in superposition. Since quantum algorithms can also generate randomness, for all functions we have $Q(f) \leq R(f)$.

An important example of the difference between the two models is provided by the OR_n function, which asks if any of the input bits is equal to 1. We have $R(\mathsf{OR}_n) = \Theta(n)$, because intuitively if the algorithm only sees a small fraction of the input bits and they are all 0, we do not know whether or not the rest of the input contains a 1. However, Grover's algorithm is a quantum algorithm that solves this problem with only $O(\sqrt{n})$ queries [Gro96]. The algorithm is also known to be tight, and we have $Q(\mathsf{OR}_n) = \Theta(\sqrt{n})$ [BBBV97].

There are several variants of Grover's algorithm that solve related problems and are sometimes more useful than the basic version of the algorithm. Most of these can be derived from the basic version of Grover's algorithm (and this sometimes adds logarithmic overhead).

In this work we need a variant of Grover's algorithm that finds a 1 in the input faster when there are many 1s. Let the Hamming weight of the input x be $t = |x|$. If we know t, then we can use Grover's algorithm on a randomly selected subset of the input of size $O(n/t)$, and one of the 1s will be in this set with high probability. Hence the algorithm will have query complexity $O(\sqrt{n/t})$. With some careful bookkeeping, this can be done even when t is unknown, and the algorithm will have expected query complexity $O(\sqrt{n/t})$. More formally, we have the following result of Boyer, Brassard, Høyer, and Tapp [BBHT98].

Lemma 9. *Given query access to a string $x \in \{0,1\}^n$, there is a quantum algorithm that when $t = |x| > 0$, always outputs an index i such that $x_i = 1$ and makes $O(\sqrt{n/t})$ queries in expectation. When $t = 0$, the algorithm does not terminate.*

Note that because we do not know $t = |x|$, we only have a guarantee on the expected query complexity of the algorithm, not the worst-case query complexity. Note also that this variant of Grover's algorithm is a zero-error algorithm in the sense that it always outputs a correct index i with $x_i = 1$ when such an index exists.

In our algorithm we use an amplified version of the algorithm of Lemma 9, which adds a log factor to the query complexity and always terminates after $O(\sqrt{n}\log n)$ queries.

Lemma 10. *Given query access to a string $x \in \{0,1\}^n$, there is a quantum algorithm that*

1. *when $|x| = 0$, the algorithm always outputs "$|x| = 0$",*
2. *when $|x| > 0$, it outputs an index i with $x_i = 1$ with probability $1 - \frac{1}{\mathrm{poly}(n)}$, and*
3. *terminates after $O\left(\sqrt{\frac{n}{|x|+1}}\log n\right)$ queries with probability $1 - \frac{1}{\mathrm{poly}(n)}$.*

Proof. This algorithm is quite straightforward. We simply run $O(\log n)$ instances of the algorithm of Lemma 9 in parallel and halt if any one of them halts. If we reach our budget of $O(\sqrt{n}\log n)$ queries, then we halt and output "$|x| = 0$".

Let us argue that the algorithm has the claimed properties. First, since the algorithm of Lemma 9 does not terminate when $|x| = 0$, our algorithm will correctly output "$|x| = 0$" at the end for such inputs. When $|x| > 0$, we know that the algorithm of Lemma 9 will find an index i with $x_i = 1$ with high probability after $O(\sqrt{n})$ queries. The probability that $O(\log n)$ copies of this algorithm do not find such an i is exponentially small in $O(\log n)$, or polynomially small in n. Finally, our algorithm makes only $O(\sqrt{n}\log n)$ queries when $|x| = 0$ by construction. When $|x| > 0$, we know that the algorithm of Lemma 9 terminates after an expected $O(\sqrt{n/|x|})$ queries, and hence halts with high probability after $O(\sqrt{n/|x|})$ queries by Markov's inequality. The probability that none of $O(\log n)$ copies of the algorithm halt after making $O(\sqrt{n/|x|})$ queries each is inverse polynomially small in n again. □

2.2 Quantum algorithm

We are now ready to present our main result for quantum query complexity, which we restate below.

Theorem 1. *Let $h : \{0,1\}^n \to \{0,1\}$ be computed by a depth-2 circuit where the top gate is a function $f : \{0,1\}^m \to \{0,1\}$ and the bottom level gates are* AND *gates on a subset of the input bits and their negations (as depicted in Figure 1). Then we have*

$$Q(h) = O\Big(\sqrt{Q(f)\cdot n}\cdot \log^2(mn)\Big). \tag{2}$$

While Theorem 1 allows the bottom AND gates to depend on negated variables, it will be without loss of generality in the proof to assume that all input variables are unnegated. This is because we can instead work with the function $h' : \{0,1\}^{2n} \to \{0,1\}$ obtained by treating the positive and negative versions of a variable separately, increasing our final quantum query upper bound by a constant factor.

We now define some notation that will aid with the description and analysis of the algorithm. We know that our circuit h has m AND gates and n input bits x_i. We say an AND gate

has *high fan-in* if the number of inputs to that AND gate is greater than or equal to $n/Q(f)$. Note that if our circuit h has no high fan-in gates, then we are done, because we can simply use the upper bound for block composition, i.e., $Q(f \circ g) = O(Q(f)Q(g))$, to compute h, since we will have $Q(h) = O(Q(f) \times \sqrt{n/Q(f)}) = O(\sqrt{Q(f) \cdot n})$.

Our goal is to reduce to this simple case. More precisely, we will start with the given circuit h, make some queries to the input, and then simplify the given circuit to obtain a new circuit h'. The new circuit will have no high fan-in gates, but will still have $h'(x) = h(x)$ on the given input x. Note that h' and h have the same output only for the given input x, and not necessarily for all inputs.

For any such circuit h, let $S \subseteq [m]$ be the set of all high fan-in AND gates, and let $w(S)$ be the total fan-in of S, which is the sum of fan-ins of all gates in S. In other words, it is the total number of wires incident to the set S. Since the set S only has gates with fan-in at least $n/Q(f)$, we have

$$w(S) \geq n|S|/Q(f). \tag{6}$$

We now present our first algorithm, which is a subroutine in our final algorithm. This algorithm's goal is to take a circuit h, with $|S|$ high fan-in gates and $w(S)$ wires incident on S, and reduce the size of $w(S)$ by a factor of 2. Ultimately we want to have $|S| = w(S) = 0$, and hence if we can decrease the size of $w(S)$ by 2, we can repeat this procedure logarithmically many times to get $|S| = w(S) = 0$.

Lemma 11. *Let $h : \{0,1\}^n \to \{0,1\}$ be a depth-2 circuit where the top gate is a function $f : \{0,1\}^m \to \{0,1\}$ and the bottom level gates are* AND *gates on a subset of the input bits and their negations (as depicted in Figure 1). Let $w(S)$ be the total fan-in of all high fan-in gates in h (i.e., gates with fan-in $\geq n/Q(f)$).*

Then there is a quantum query algorithm that makes $O(\sqrt{Q(f) \cdot n} \log n)$ queries to $x \in \{0,1\}^n$ and outputs a new circuit h' of the same form such that $w(S') \leq w(S)/2$, where $w(S')$ is the total fan-in of all high fan-in gates in h', and such that with probability $1 - \frac{1}{\text{poly}(n)}$ (over the internal randomness of the algorithm) we have $h(x) = h'(x)$ for the query input x.[8]

Proof. The overall structure of the claimed algorithm is the following: We query some well-chosen input bits, and on learning the values of these bits, we simplify the circuit accordingly. If an input bit is 0, then we delete all the AND gates that use that input bit. If an input bit is 1, we delete all outgoing wires from that input bit since a 1-input does not affect the output of an AND gate.

Since the circuit will change during the algorithm, let us define S_0 to be the initial set of high fan-in (i.e., gates with fan-in $\geq n/Q(f)$) AND gates in h.

We also define the degree of an input x_i, denoted $\deg(i)$, to be the number of high fan-in AND gates that it is an input to. Note that this is not the total number of outgoing wires from x_i, but only those that go to high fan-in AND gates, i.e., gates in the set S. With this definition, note that $\sum_{i \in [n]} \deg(i) = w(S)$, for any circuit. We say an input bit x_i is *high degree* if $\deg(i) \geq |S_0|/(2Q(f))$. This value is chosen since it is at least half the average degree of all x_i in the initial circuit h. As the algorithm progresses, the circuit will change, and some inputs that were initially high degree may become low degree as the algorithm progresses, but a low

[8] The new circuit h' is only promised to satisfy $h(x) = h'(x)$ on the specific query input x on which this algorithm is run.

degree input will never become high degree. But note that the definition of a high-degree input bit does not change, since it only depends on S_0 and $Q(f)$, which are fixed for the duration of the algorithm.

Finally, we call an input bit x_i is *marked* if $x_i = 0$. We are now ready to describe our algorithm by the following pseudocode (see Algorithm 1).

Algorithm 1 The algorithm of Lemma 11.

$S_0 \leftarrow$ Set of high fan-in AND gates in h
repeat
  $M \leftarrow$ Set of high-degree marked inputs ▷ $M := \left\{i : x_i = 0 \wedge \deg(i) \geq \frac{|S_0|}{(2Q(f))}\right\}$
  Grover Search for an index i in M
  if we find such an i **then**
    Delete all AND gates that use x_i as an input
  end if
until Grover Search fails to find an $i \in M$
Delete all remaining high-degree inputs and all outgoing wires from these inputs

In more detail, we repeatedly use the version of Grover's algorithm in Lemma 10 to find a high-degree marked input, which is an input x_i such that $x_i = 0$ and $\deg(i) \geq \frac{|S_0|}{2Q(f)}$. If we find such an input, we delete all the AND gates that use x_i as an input, and repeat this procedure. Note that when we repeat this procedure, the circuit has changed, and hence the set of high-degree input bits may become smaller. The algorithm halts when Grover's algorithm is unable to find any high-degree marked inputs. At this point, all the high-degree inputs are necessarily unmarked with very high probability, which means they are set to 1. We can now delete all these input bits and their outgoing wires because AND gates are unaffected by input bits set to 1.

Let us now argue that this algorithm is correct. Let S' denote the set of high fan-in AND gates in the new circuit h' obtained at the end of the algorithm, and $w(S')$ be the total fan-in of gates in S'. Note that when the algorithm terminates, there are no high-degree inputs (marked or unmarked). Hence every input bit that has not been deleted has $\deg(i) < \frac{|S_0|}{2Q(f)}$. Since there are at most n input bits, we have

$$w(S') = \sum_{i \in [n]} \deg(i) < \frac{n}{2Q(f)}|S_0|. \tag{7}$$

But we also know that we started with $w(S) \geq n|S_0|/Q(f)$, since each gate in S_0 has fan-in at least $n/Q(f)$. Hence $w(S') \leq w(S)/2$, which proves that the algorithm is correct.

We now analyze the query complexity of this algorithm. Let the loop in the algorithm execute r times. It is easy to see that $r \leq 2Q(f)$ because each time a high-degree marked input is found, we delete all the AND gates that use it as an input, which is at least $|S_0|/(2Q(f))$ gates. Since there were at most S_0 gates to begin with, this procedure can only repeat $2Q(f)$ times.

When we run Grover's algorithm to search for a high-degree marked input bit x_i in the first iteration of the loop, suppose there are k_1 high-degree marked inputs. Then the variant of Grover's algorithm in Lemma 10 finds a marked high-degree input and makes $O(\sqrt{n/k_1}\log n)$

queries with probability $1 - \frac{1}{\text{poly}(n)}$. In the second iteration of the loop, the number of high-degree marked inputs, k_2, has decreased by at least one. It can also decrease by more than 1 since we deleted several AND gates, and some high-degree inputs can become low-degree. In this iteration, our variant of Grover's algorithm (Lemma 10) makes $O(\sqrt{n/k_2}\log n)$ queries, and we know that $k_1 > k_2$. This process repeats and we have $k_1 > k_2 > \cdots > k_r$. Since there was at least one high-degree marked input in the last iteration, $k_r \geq 1$. Combining these facts we have for all $j \in [r]$, $k_j \geq r - j + 1$. Thus the total expected query complexity is

$$O\left(\sum_{j=1}^{r}\sqrt{\frac{n}{k_j}}\log n\right) = O\left(\sum_{j=1}^{r}\sqrt{\frac{n}{r-j+1}}\log n\right) = O\left(\sqrt{n}\sum_{j=1}^{r}\frac{1}{\sqrt{j}}\log n\right) = O\left(\sqrt{nr}\log n\right), \tag{8}$$

which is $O\left(\sqrt{n \cdot Q(f)}\log n\right)$. We now have a quantum query algorithm that satisfies the conditions of the lemma with probability at least $1 - \frac{1}{\text{poly}(n)}$. □

We are now ready to prove Theorem 1.

Proof of Theorem 1. We start by applying the algorithm in Lemma 11 to our circuit as many times as needed to ensure that set S is empty. Since each run of the algorithm reduces $w(S)$ by a factor of 2, and $w(S)$ can start off being as large as $m \cdot n$, where m is the number of AND gates and n is the number of inputs, we need to run the algorithm $\log(mn)$ times. Since the algorithm of Lemma 11 is correct with probability $1 - \frac{1}{\text{poly}(n)}$, we do not need to boost the success probability of the algorithm. The total number of queries needed to ensure S is empty is $O(\sqrt{Q(f) \cdot n}\log(n)\log(mn))$.

Now we are left with a circuit h' with no high fan-in AND gates. That is, all AND gates have fan-in at most $n/Q(f)$. We now evaluate h' using the standard composition theorem for disjoint sets of inputs, which has query complexity

$$O(Q(f) \cdot Q(\mathsf{AND}_{n/Q(f)})) = O(Q(f) \cdot \sqrt{n/Q(f)}) = O\left(\sqrt{Q(f) \cdot n}\right). \tag{9}$$

The total query complexity is $O(\sqrt{Q(f) \cdot n}\log(n)\log(mn)) = O(\sqrt{Q(f) \cdot n}\log^2(mn))$. □

Note that we have not attempted to reduce the logarithmic factors in this upper bound. We believe it is possible to make the quantum upper bound match the upper bound for approximate degree with a more careful analysis and slightly different choice of parameters in the algorithm.

3 Approximating polynomials for composed functions

3.1 Preliminaries

We now define the various measures of Boolean functions and polynomials that we require in this section. Since we only care about polynomials approximating Boolean functions, we focus without loss of generality on multilinear polynomials as any polynomial over the domain $\{0,1\}^n$ can be converted into a multilinear polynomial (since it never helps to raise a Boolean variable to a power greater than 1).

The approximate degree of a Boolean function, commonly denoted $\widetilde{\deg}(f)$, is the minimum degree of a polynomial that entrywise approximates the Boolean function. It is a basic complexity measure and is known to be polynomially related to a host of other complexity measures such as decision tree complexity, certificate complexity, and quantum query complexity [BdW02, BT21]. We also use another complexity measure of polynomials, which is the sum of absolute values of all the coefficients of the polynomial. This is the query analogue of the so-called μ-norm used in communication complexity [LS09, Definition 2.7]. We now formally define these measures.

Definition 12. Let $p : \mathbb{R}^n \to \mathbb{R}$ be a multilinear polynomial

$$p(x_1, \ldots, x_n) = \sum_{s \in \{0,1\}^n} \alpha_s x_1^{s_1} \cdots x_n^{s_n}. \tag{10}$$

We define the following complexity measures of the polynomial p:

$$\deg(p) = \max\Big\{ \sum_{i \in [n]} |s_i| : \alpha_s \neq 0 \Big\} \qquad \text{and} \qquad \mu(p) = \sum_{s \in \{0,1\}^n} |\alpha_s|. \tag{11}$$

For a Boolean function $f : \{0,1\}^n \to \{0,1\}$, we define the following complexity measures:

$$\deg_\varepsilon(f) = \min\{\deg(p) : \forall x \in \{0,1\}^n,\ |f(x) - p(x)| \leq \varepsilon\} \tag{12}$$

$$\mu_\varepsilon(f) = \min\{\mu(p) : \forall x \in \{0,1\}^n,\ |f(x) - p(x)| \leq \varepsilon\} \tag{13}$$

Finally, we define $\widetilde{\deg}(f) = \deg_{1/3}(f)$ and $\widetilde{\mu}(f) = \mu_{1/3}(f)$.

We use the following standard relationship between the two measures in our results.

Lemma 13. *For any multilinear polynomial $p : \mathbb{R}^n \to \mathbb{R}$ such that $|p(x)| = O(1)$ for all $x \in \{0,1\}^n$, we have*

$$\log \mu(p) = O(\deg(p) \log n). \tag{14}$$

Consequently, for any Boolean function $f : \{0,1\}^n \to \{0,1\}$ and $\varepsilon \in [0, 1/3]$, we have

$$\log \mu_\varepsilon(f) = O(\deg_\varepsilon(f) \log n). \tag{15}$$

Proof. First let us switch to the $\{-1,1\}$ representation instead of the $\{0,1\}$ representation we have used so far. Let $y_i = (-1)^{x_i}$, and replace every occurrence of x_i in the polynomial p with $\frac{1}{2}(1 + y_i)$ to obtain a multilinear polynomial $p(y_1, \ldots, y_n) = \sum_{s \in \{0,1\}^n} \beta_s y_1^{s_1} \cdots y_n^{s_n}$. In this representation, a coefficient β_s is simply the expectation over the hypercube of the product of p and a parity function, and hence is at most $O(1)$ in magnitude. Since there are only $\binom{n}{\leq \deg(p)} \leq n^{\deg(p)}$ monomials, the sum of absolute values of all coefficients is $O(n^{\deg(p)})$.

When we switch from this representation back to the $\{0,1\}$ representation, we replace every y_i with $2x_i - 1$. Consider this transformation on a single monomial with coefficient 1. This converts the monomial of degree d into a polynomial over those d variables, such that the sum of coefficients in this polynomial is at most 3^d. Thus the sum of absolute values of all coefficients is $\mu(p) = O(3^{\deg(p)} n^{\deg(p)}) = n^{O(\deg(p))}$, which proves (14).

Now consider any Boolean function $f : \{0,1\}^n \to \{0,1\}$, and a multilinear polynomial p that minimizes $\deg_\varepsilon(f)$. We can apply (14) to this polynomial to obtain $\log \mu(p) = O(\deg(p) \log n)$. Since $\deg(p) = \deg_\varepsilon(f)$ by assumption, and $\mu_\varepsilon(f) \leq \mu(p)$, since $\mu_\varepsilon(f)$ minimizes over all ε-approximating polynomials, we get $\log \mu_\varepsilon(f) = O(\deg_\varepsilon(f) \log n)$. □

This shows that $\log \mu(p)$ is at most $\deg(p)$ (up to log factors). However, $\log \mu(p)$ may be much smaller than $\deg(p)$, as evidenced by the polynomial $p(x) = x_1 \cdots x_n$. Similarly, $\log \widetilde{\mu}(f)$ may be much smaller than $\widetilde{\deg}(f)$, as evidenced by the AND function on n bits, which has $\widetilde{\deg}(\mathsf{AND}_n) = \Theta(\sqrt{n})$ [NS94], but $\widetilde{\mu}(\mathsf{AND}_n) \leq 1$.

3.2 Polynomial upper bound

In this section we prove Theorem 2, which follows from the following more general composition theorem.

Theorem 14. *Let $h : \{0,1\}^n \to \{0,1\}$ be computed by a depth-2 circuit where the top gate is a function $f : \{0,1\}^m \to \{0,1\}$ and the bottom level gates are* AND *gates on a subset of the input bits and their negations (as depicted in Figure 1). Then*

$$\deg_\varepsilon(h) = O\left(\sqrt{n \log \mu_\varepsilon(f)} + \sqrt{n \log(1/\varepsilon)}\right) = O\left(\sqrt{n \deg_\varepsilon(f) \log m} + \sqrt{n \log(1/\varepsilon)}\right). \quad (16)$$

Proof. Let us first fix some notation. We will use $x \in \{0,1\}^n$ to refer to the input of the full circuit $h : \{0,1\}^n \to \{0,1\}$. Let the inputs to the top $f : \{0,1\}^m \to \{0,1\}$ gate be called $y_1, \ldots, y_m$.

Let $p : \{0,1\}^m \to \{0,1\}$ be a polynomial that minimizes $\mu_\varepsilon(f)$. Thus we have for all $y \in \{0,1\}^m$, $|p(y) - f(y)| \leq \varepsilon$. More explicitly, $p(y_1, \ldots, y_m) = \sum_{s \in \{0,1\}^m} \alpha_s y_1^{s_1} \cdots y_n^{s_n}$, where $\mu_\varepsilon(f) = \sum_{s \in \{0,1\}^m} |\alpha_s|$, and each y_i is the AND of some subset of bits in x. Since the product of ANDs of variables is just an AND of all the variables involved in the product, for each $s \in \{0,1\}^m$, there is a subset $T_s \subseteq [n]$ such that $y_1^{s_1} \cdots y_n^{s_n} = \bigwedge_{i \in T_s} x_i$.

Using this we can replace all the y variables in the polynomial p, to obtain

$$q(x) = \sum_{s \in \{0,1\}^m} \alpha_s \bigwedge_{i \in T_s} x_i. \quad (17)$$

Since p was an ε approximation to f, q is an ε approximation to h. Now we can replace every occurrence of $\bigwedge_{i \in T_s} x_i$ with a low error approximating polynomial for the AND of the bits in T_s. We know that the approximate degree of the AND function to error δ is $O(\sqrt{n \log(1/\delta)})$ [BCdWZ99]. If we approximate each AND to error $\delta = \varepsilon/\mu_\varepsilon(f)$, then by the triangle inequality the total error incurred by this approximation is at most $\sum_{s \in \{0,1\}^m} |\alpha_s| \varepsilon/\mu_\varepsilon(f) = \varepsilon$. Choosing $\delta = \varepsilon/\mu_\varepsilon(f)$, each AND is approximated by a polynomial of degree $O(\sqrt{n \log(1/\delta)}) = O\left(\sqrt{n \log \mu_\varepsilon(f)} + \sqrt{n \log(1/\varepsilon)}\right)$. Hence the resulting polynomial $q(x)$ has this degree and approximates the function h to error 2ε. By standard error reduction techniques [BNRdW07], we can make this error smaller than ε at a constant factor increase in the degree. This establishes the first equality in (16), and the second equality follows from Lemma 13. □

4 Applications to linear-size AC^0 circuits

4.1 Preliminaries

A Boolean circuit is defined via a directed acyclic graph. Vertices of fan-in 0 represent input bits, vertices of fan-out 0 represent outputs, and all other vertices represent one of the following

logical operations: a NOT operation (of fan-in 1), or an unbounded fan-in AND or OR operation. The size of the circuit is the total number of AND and OR gates. The depth of the circuit is the length of the longest path from an input bit to an output bit.

For any constant integer $d > 0$, AC^0_d refers to the class of all such circuits of polynomial size and depth d. AC^0 refers to $\cup_{d=1}^{\infty}\mathsf{AC}^0_d$. Similarly, LC^0_d refers to the class of all such circuits of size $O(n)$ and depth d, while LC^0 refers to $\cup_{d=1}^{\infty}\mathsf{LC}^0_d$. We will associate any circuit C with the function it computes, so for example $\widetilde{\deg}(C)$ denotes the approximate degree of the function computed by C.

It will be convenient to assume that any AC^0_d circuit is layered, in the sense that it consists of d levels of gates which alternate between being comprised of all AND gates or all OR gates, and all negations appear at the input level of the circuit. Any AC^0_d circuit of size s can be converted into a layered circuit of size $O(d \cdot s)$, and hence making this assumption does not change any of our upper bounds.

4.2 Quantum query complexity

Applying our composition theorem for quantum algorithms (Theorem 1) inductively, we obtain a sublinear upper bound on the quantum query complexity of LC^0_d circuits.

Theorem 3. *For all constants $d \geq 0$ and all functions $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d, we have $Q(h) = \widetilde{O}(n^{1-2^{-d}})$.*

Proof. We prove this for depth-d LC^0 circuits by induction on d. The base case is $d = 1$, where the function is either AND or OR on n variables, both of which have quantum query complexity $O(\sqrt{n})$ [Gro96].

Now consider a function h, which is a layered depth-d AC^0 circuit of size $O(n)$. It can be written as a depth-2 circuit (as in Theorem 1) where the top function is a LC^0 circuit f of depth $d-1$ on at most $O(n)$ inputs, and the bottom layer has only AND gates. (If the bottom layer has OR gates we can consider the negation of the function without loss of generality, since the quantum query complexity of a function and its negation is the same.)

By the induction hypothesis we know that the quantum query complexity of any depth-$(d-1)$, size-$O(n)$ AC^0 circuit with $O(n)$ inputs is $\widetilde{O}(n^{1-2^{-(d-1)}})$. Invoking Theorem 1, we have that the quantum query complexity of the depth-d function h is $\widetilde{O}(n^{1-2^{-d}})$. □

4.3 Approximate degree upper bound

We can now prove Theorem 5, restated below for convenience:

Theorem 5. *For all constant $d \geq 0$, and any $\varepsilon > 0$, and all functions $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d, we have*

$$\deg_\varepsilon(h) = \widetilde{O}\Big(n^{1-2^{-d}} \log^{2^{-d}}(1/\varepsilon)\Big). \tag{5}$$

This follows from a more general result:

Theorem 15. *For any function $h : \{0,1\}^n \to \{0,1\}$ computed by an AC^0 circuit of size $s \geq 1$ and depth $d \geq 1$, we have*

$$\deg_\varepsilon(h) = \begin{cases} O\left(\sqrt{n \log(1/\varepsilon)}\right) & \text{if } \varepsilon \leq 2^{-s} \Leftrightarrow \log(1/\varepsilon) \geq s \\ \widetilde{O}\left(\sqrt{n} s^{1/2-2^{-d}} (\log(1/\varepsilon))^{2^{-d}}\right) & \text{if } \varepsilon > 2^{-s} \Leftrightarrow \log(1/\varepsilon) < s \end{cases}. \tag{18}$$

In particular, for any $h \in \mathsf{LC}^0_d$, we have $\widetilde{\deg}(h) = \widetilde{O}(n^{1-2^{-d}})$.

Proof. We prove this for depth-d AC^0 circuits by induction on d. The base case is $d = 1$, where the function is either AND or OR on n variables, both of which have ε-approximate degree $O(\sqrt{n \log(1/\varepsilon)})$ [BCdWZ99].

Now consider a function h, which is a general depth-d AC^0 circuit of size s. It can be written as a depth-2 circuit (as in Theorem 2) where the top function is a size-s AC^0 circuit f of depth $d-1$ on at most s inputs, and the bottom layer has only AND gates. If the bottom layer has OR gates we can consider the negation of the function without loss of generality, since the ε-approximate degree of a function and its negation is the same.

In the first case, if $\varepsilon \leq 2^{-s}$, then for any function $f : \{0,1\}^s \to \{0,1\}$ there is a polynomial of degree s and sum of coefficients at most 2^s that exactly equals f on all Boolean inputs. Hence we can apply Theorem 2 to get that $\deg_\varepsilon(h) = O(\sqrt{ns} + \sqrt{n \log(1/\varepsilon)}) = O(\sqrt{n \log(1/\varepsilon)})$.

In the second case, if $\varepsilon > 2^{-s}$, by the induction hypothesis we know that the ε-approximate degree of any depth-$(d-1)$, size-$O(s)$ AC^0 circuit with s inputs is $\widetilde{O}(s^{1-2^{-(d-1)}}(\log(1/\varepsilon))^{2^{-(d-1)}})$. Invoking Theorem 2, we have that the approximate degree of the depth-d function is

$$\widetilde{O}\left(\sqrt{ns^{1-2^{-(d-1)}}(\log(1/\varepsilon))^{2^{-(d-1)}}} + \sqrt{n \log(1/\varepsilon)}\right) = \widetilde{O}\left(\sqrt{n} s^{1/2-2^{-d}} (\log(1/\varepsilon))^{2^{-d}}\right). \quad (19)$$

□

4.4 Approximate degree lower bound

In this section we prove our lower bound on the approximate degree of LC^0_d, restated below for convenience.

Theorem 6. *For all constants $d \geq 0$, there exists a function $h : \{0,1\}^n \to \{0,1\}$ in LC^0_d with $\widetilde{\deg}(h) \geq n^{1-2^{-\Omega(\sqrt{d})}}$.*

Before proving the theorem, we will need to introduce several lemmas. The first lemma follows from the techniques of [ABO84] (see [Kop13] for an exposition).

Lemma 16. *There exists a Boolean circuit C with n inputs, of depth 3, and size $\widetilde{O}(n^2)$ satisfying the following two properties:*

- *$C(x) = 0$ for all x of Hamming weight at most $n/3$.*
- *$C(x) = 1$ for all x of Hamming weight at least $2n/3$.*

We refer to the function computed by the circuit C of Lemma 16 as GAPMAJ, short for a gapped majority function (such a function is sometimes also called an *approximate majority* function).

The following lemma of [BCH+17] says that if f has large ε-approximate degree for $\varepsilon = 1/3$, then block-composing f with GAPMAJ on $O(\log n)$ bits yields a function with just as high ε'-approximate degree, with ε' very close to $1/2$.

Lemma 17 ([BCH+17])**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be any function. Then for $\varepsilon = 1/2 - 1/n^2$, $\deg_\varepsilon(\mathsf{GAPMAJ}_{10 \log n} \circ f) \geq \widetilde{\deg}(f)$.*

The following lemma says that if f has large ε-approximate degree for ε very close to $1/2$, then block-composing any function g with f results in a function of substantially larger approximate degree than g itself.

Lemma 18 ([She13a])**.** *Let* $g\colon \{0,1\}^m \to \{0,1\}$ *and* $f\colon \{0,1\}^n \to \{0,1\}$ *be any functions. Then* $\widetilde{\deg}(g \circ f) \geq \widetilde{\deg}(g) \cdot \deg_{1/2-1/m^2}(f)$.

Combining Lemmas 17 and 18, we conclude:

Corollary 19. *Let* $g\colon \{0,1\}^m \to \{0,1\}$ *and* $f\colon \{0,1\}^n \to \{0,1\}$ *be any functions. Then* $\widetilde{\deg}(g \circ \mathsf{GAPMAJ}_{10\log n} \circ f) \geq \widetilde{\deg}(g) \cdot \widetilde{\deg}(f)$.

We are now ready to prove Theorem 6, which is restated at the beginning of this section.

Proof of Theorem 6. Let $\ell \geq 1$ be any constant integer to be specified later (ultimately, we will set $\ell = \Theta(\sqrt{d})$, where d is as in the statement of the theorem). [BKT18] exhibit a circuit family $C^*\colon \{0,1\}^n \to \{0,1\}$ of depth at most 3ℓ, size at most n^2, and approximate degree satisfying $\widetilde{\deg}(C^*) \geq D$ for some $D \geq \widetilde{\Omega}(n^{1-2^{-\ell}})$. We need to transform this quadratic-size circuit into a circuit C of *linear* size, without substantially reducing its approximate degree, or substantially increasing its depth (in particular, the depth of C should be at most d).

To accomplish this, we apply the following iterative transformation. At each iteration i, we produce a new circuit $C^i\colon \{0,1\}^n \to \{0,1\}$ of linear size, such that $\widetilde{\deg}(C^i)$ gets closer and closer to $\widetilde{\deg}(C)$ as i grows. Our final circuit will be $C := C^\ell$.

C^1 is defined to simply be OR_n, which is clearly in LC^0_1.

The transformation from C^{i-1} into C^i works as follows. C^i feeds $\sqrt{n}$ copies of $C^{i-1}_{\sqrt{n}/(10\log n)}$ into the circuit $C^*_{\sqrt{n}} \circ \mathsf{GAPMAJ}_{10\log n}$. Here, C^{i-1}_k denotes the function C^{i-1} constructed in the previous iteration, and defined on k inputs; similarly, $C^*_k\colon \{0,1\}^k \to \{0,1\}^n$ refers to the function C^* constructed by [BKT18], defined on k inputs. That is:

$$C^i = C^*_{\sqrt{n}} \circ \mathsf{GAPMAJ}_{10\log n} \circ C^{i-1}_{\sqrt{n}/(10\log n)}. \tag{20}$$

Observe that C^i is a function on $\sqrt{n} \cdot 10\log n \cdot (\sqrt{n}/(10\log n)) = n$ bits. We now establish the following two lemmas about C^i.

Lemma 20. *C^i is computed by a circuit of depth at most $(3\ell+3)\cdot i$, and size at most $2 \cdot i \cdot n$.*

Proof. Clearly this is true for $i = 1$, since C^1 is computed by a circuit of size and depth 1. Assume by induction that it is true for $i-1$. Recalling that $\mathsf{GAPMAJ}_{10\log n}$ is computed by a circuit of size $O(\log^2 n)$ and depth 3, and $C^*_{\sqrt{n}}$ is computed by a circuit of size n and depth 3ℓ, it is immediate from Equation (20) that C^i is computed by a circuit satisfying the following properties:

- The depth is at most $3\ell + 3 + (3\ell+3)(i-1) = (3\ell+3)i$.
- The size is at most $n + O(\sqrt{n}\cdot\log^2 n) + (\sqrt{n}\cdot 10\log n)\cdot(2\cdot(i-1)\cdot\sqrt{n}/(10\log n))$. For large enough n, this is at most $2n + 2\cdot(i-1)\cdot n = 2\cdot i\cdot n$.

□

Lemma 21. *For $i > 1$, $\widetilde{\deg}(C^i) \geq \Omega\left(\widetilde{\deg}(C^*_{\sqrt{n}}) \cdot \widetilde{\deg}(C^{i-1}_{\sqrt{n}/(10\log n)})\right)$.*

Proof. Immediate from Corollary 19. □

Since $\widetilde{\deg}(C^1) = \Omega(\sqrt{n})$, repeated application of Lemma 21 implies that $\widetilde{\deg}(C^2) = \Omega(\sqrt{D} \cdot n^{1/4})$, $\widetilde{\deg}(C^3) = \Omega\left(\sqrt{D} \cdot (\sqrt{D} \cdot n^{1/4})^{1/2}\right) = \Omega(D^{3/4} \cdot n^{1/8})$, and in general, $\widetilde{\deg}(C^i) = \Omega\left(D^{1-2^{-i}} \cdot n^{2^{-i}}\right)$.

Setting $i = \ell$, we obtain a circuit $C^\ell \colon \{0,1\}^n \to \{0,1\}$ with the following properties:

- By Lemma 20, C^ℓ has size at most $2\ell n$ and depth at most $d := 2\ell^2$.
- There is a constant c_0 such that C^ℓ has approximate degree at least $\Omega\left(c_0^\ell \cdot D^{1-2^{-\ell+1}} \cdot n^{2^{-\ell}}\right) \geq \Omega(c_0^\ell \cdot n^{1-2^{-\ell+1/2}})$.

Hence, for any constant value of $d = 2\ell^2$, we have constructed a circuit of depth d, size $O(n)$, and approximate degree at least $\Omega(n^{1-2^{-\Omega(\sqrt{d})}})$, as required by the theorem. □

4.5 Sublinear-size circuits of arbitrary depth

Theorem 1 and Theorem 2 also allow us to prove sublinear quantum query complexity and approximate degree upper bounds for arbitrary circuits of sublinear size.

Theorem 22. *Let $h : \{0,1\}^n \to \{0,1\}$ be computed by a layered circuit of size $s \leq n$. Then h has quantum query complexity $Q(h) = \tilde{O}(\sqrt{ns})$ and approximate degree $\widetilde{\deg}(h) = O(\sqrt{ns})$.*

Proof. Without loss of generality, a function h computed by a layered circuit of size $s \leq n$ can be written as a depth-2 circuit with a function $f : \{0,1\}^s \to \{0,1\}$ as the top gate and AND gates at the bottom. (The case where the bottom level consists of OR gates can be handled by negating the function.) The quantum query upper bound then follows immediately from Theorem 1, as $Q(f) \leq s$. Moreover, for any function f, we have $\log \mu_0(f) = O(s)$, since the trivial polynomial obtained by adding all conjunctions over yes-inputs of f satisfies this. Hence from Theorem 2 we have $\widetilde{\deg}(h) = O(\sqrt{ns})$. □

5 Applications to agnostic PAC learning

Our new upper bounds on the approximate degree of LC^0 circuits yield new subexponential time learning algorithms in the agnostic model. In this section, we provide background for, and the proof of, our main learning result restated below.

Theorem 7. *The concept class of n-bit functions computed by LC^0 circuits of depth d can be learned in the distribution-free agnostic PAC model in time $2^{\widetilde{O}(n^{1-2^{-d}})}$. More generally, size-$s$ AC^0_d circuits can be learned in time $2^{\widetilde{O}(\sqrt{n}s^{1/2-2^{-d}})}$.*

PAC and agnostic learning models. In the classic Probably Approximately Correct (PAC) learning model of Valiant [Val84], we have access to an unknown function $f : \{0,1\}^n \to \{0,1\}$ from a known class of functions $\mathcal{C}$, called the concept class, through samples $(x, f(x))$, where x is drawn from an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$. The goal is to learn a hypothesis $h : \{0,1\}^n \to \{0,1\}$, such that with probability $1 - \delta$ (over the choice of samples), $h(x)$ has (Boolean) loss at most ε with respect to $\mathcal{D}$. Here, the Boolean loss $\mathsf{err}_{\mathcal{D}}(h, f)$ of h is defined to be $\Pr_{x \sim \mathcal{D}}[h(x) \neq f(x)] \leq \varepsilon$.

Since the learning algorithm does not know $\mathcal{D}$ and is required to work for all $\mathcal{D}$, this model is also called the distribution-independent (or distribution-free) PAC model. Unfortunately, in the distribution-free setting, very few concept classes are known to be PAC learnable in polynomial time or even subexponential time (i.e., time $2^{n^{1-\delta}}$ for some constant $\delta > 0$).

Kearns, Schapire, and Sellie [KSS94] then proposed the more general (and challenging) agnostic PAC learning model, which removes the assumption that examples are determined by a function at all, let alone a function in the concept class $\mathcal{C}$. The learner now knows nothing about how examples are labeled, but is only required to learn a hypothesis h that is at most ε worse than the best possible classifier from the class $\mathcal{C}$.

We now describe the agnostic PAC model more formally. Let $\mathcal{D}$ be any distribution on $\{0,1\}^n \times \{0,1\}$, and let $\mathcal{C}$ be a concept class, i.e., a set of Boolean functions on $\{0,1\}^n$. Define the error of $h\colon \{0,1\}^n \to \{0,1\}$ to be $\mathsf{err}_{\mathcal{D}}(h) := \Pr_{(x,b)\sim\mathcal{D}}[h(x) \neq y]$, and define $\mathsf{opt} := \min_{c\in\mathcal{C}} \mathsf{err}_{\mathcal{D}}(c)$. We say that $\mathcal{C}$ is agnostically learnable in time $T(n,\varepsilon,\delta)$ if there exists an algorithm which takes as input n and δ and has access to an example oracle $\mathsf{EX}(\mathcal{D})$, and satisfies the following properties. It runs in time at most $T(n,\varepsilon,\delta)$, and with probability at least $1-\delta$, it outputs a hypothesis h satisfying $\mathsf{err}_{\mathcal{D}}(h) \leq \mathsf{opt} + \varepsilon$. We say that the learning algorithm runs in *subexponential time* if there is some constant $\eta > 0$ such that for any constants ε and δ, the running time $T(n,\varepsilon,\delta) \leq 2^{n^{1-\eta}}$ for sufficiently large n.

The agnostic model is able to capture a range of realistic scenarios that do not fit within the standard PAC model. In many situations it is unreasonable to know exactly that f belongs to some class $\mathcal{C}$, since f may be computed by a process outside of our control. For example, the labels of f may be (adversarially) corrupted by noise, resulting in a function that is no longer in $\mathcal{C}$. Alternatively, f may be "well-modeled," but not *perfectly* modeled, by some concept in $\mathcal{C}$. In fact, the agnostic learning model even allows the input sample to not be described by a function f at all, in the sense that the distribution over the sample may have both $(x,0)$ and $(x,1)$ in its support. This is also realistic when the model being used does not capture all of the variables on which the true function depends.

5.1 Related work

Since the agnostic PAC model generalizes the standard PAC model, it is (considerably) harder to learn a concept class in this model. Consequently, even fewer concept classes are known to be agnostically learnable, even in subexponential time. For example, as mentioned in Section 1.4, the best known algorithm for agnostically learning the simple concept class of disjunctions, which are size-1, depth-1 Boolean circuits, runs in time[9] $2^{\widetilde{O}(\sqrt{n})}$ [KKMS08]. In contrast, they can be learned in polynomial time in the PAC model [Val84]. Meanwhile, several hardness results are known for agnostically learning disjunctions, including NP-hardness for proper learning [KSS94], and that even improper learning is as hard as PAC learning DNF [LBW95].

While it is an important and interesting problem to agnostically learn more expressive classes of circuits in subexponential time, relatively few results are known. The best known general result is that all de Morgan formulas (formulas over the gate set of AND, OR, and NOT gates) of size s can be learned in time $2^{\widetilde{O}(\sqrt{s})}$ [KKMS08, Rei11]. In particular, linear-size formulas (i.e., $s = \Theta(n)$) can be learned in time $2^{\widetilde{O}(\sqrt{n})}$, which is the same as the best known upper bound for disjunctions.

[9] For simplicity, we suppress runtime dependence on ε and δ.

Even in the relatively easier PAC model, only a small number of circuit classes are known to be learnable in subexponential time. For the well-studied class of polynomial-size DNFs, or depth-2 AC^0 circuits, we have an algorithm running in time $2^{\widetilde{O}(n^{1/3})}$ [KS04], and we know that new techniques will be needed to improve this bound [RS10]. Little is known about larger subclasses of AC^0, other than a recent paper that studied depth-3 AC^0 circuits with top fan-in t, giving a PAC learning algorithm of runtime $2^{\widetilde{O}(t\sqrt{n})}$ [DRG17], which is only subexponential when $t \ll \sqrt{n}$.

Given the current state of affairs, a subexponential-time algorithm to learn all of AC^0 in the standard PAC model would represent significant progress. Indeed, for $d > 2$, the fastest known PAC learning algorithm for depth-d AC^0 circuits runs in time $2^{n-\Omega(n/\log^{d-1} n)}$ [ST17], which is quite close to the trivial runtime of 2^n.

We view our new results for learning LC^0 and sublinear-size AC^0 circuits as intermediate steps toward this goal. We clarify that our results are incomparable to the known results about agnostically learning de Morgan formulas. A simple counting argument [Nis11] shows that there are linear-size DNFs that are not computable by formulas of size $o(n^2/\log n)$, so one cannot learn even depth-2 LC^0 in subexponential time via the learning algorithm for de Morgan formulas. On the other hand, there are linear-size de Morgan formulas (of superconstant depth) that are not in LC^0, or even AC^0.

Motivated by the lack of positive results in the distribution-free PAC learning model, [ST17] study algorithms for learning various circuit classes, with the goal of "only" achieving a *non-trivial savings* over trivial 2^n-time algorithms. By achieving non-trivial savings, [ST17] mean a runtime of $2^{n-o(n)}$; prior work had already connected non-trivial learning algorithms to circuit lower bounds [KKO13, OS17]. The subexponential runtimes we achieve in our work are significantly faster than the $2^{n-o(n)}$-time algorithms of [ST17]; in addition, our algorithms work in the challenging agnostic setting, rather than just the PAC setting. On the other hand, the algorithms of [ST17] apply to more general circuit classes than LC^0.

As mentioned previously, [KS04] gave a $2^{\widetilde{O}(n^{1/3})}$-time algorithm for PAC learning polynomial size DNF formulas; their algorithm is based on a $\widetilde{O}(n^{1/3})$ upper bound on the *threshold degree* of such formulas. In unpublished work, [Tal18] has observed that the argument in [KS04, Theorem 4] can be generalized to show that for constant $d \geq 2$, any depth-d LC^0 circuit has threshold degree at most $\widetilde{O}\big(n^{1-1/(3\cdot 2^{d-3})}\big)$. This in turn yields a PAC learning algorithm for LC^0 running in time $\exp\big(\widetilde{O}\big(n^{1-1/(3\cdot 2^{d-3})}\big)\big)$. Note that this is in the standard PAC model, not the agnostic PAC model. As mentioned in Section 1, prior to our work, no subexponential time algorithm was known for agnostically learning even LC^0_3 in subexponential time.

5.2 Linear regression and the proof of Theorem 7

Our learning algorithm applies the well-known *linear regression* framework for agnostic learning that was introduced by [KKMS08]. The algorithm of [KKMS08] works whenever there is a "small" set of "features" $\mathcal{F}$ (where each feature is a function mapping $\{0,1\}^n$ to $\mathbb{R}$) such that each concept in the concept class $\mathcal{C}$ can be approximated to error ε in the ℓ_∞ norm via a linear combination of the features in $\mathcal{F}$. Roughly speaking, given a sufficiently large sample S from an (unknown) distribution over $\{0,1\}^n \times \{0,1\}$, the algorithm finds a linear combination h of the features of $\mathcal{F}$ that minimizes the empirical ℓ_1 loss, i.e., h minimizes $\sum_{(x_i,b_i)\in S} |h(x_i) - b_i|$ among all linear combinations of features from $\mathcal{F}$. An (approximately) optimal h can be found

in time $\mathrm{poly}(\mathcal{F})$ by solving a linear program of size $\mathrm{poly}(|\mathcal{F}|, |S|)$.

Lemma 23 ([KKMS08])**.** *Let $\mathcal{F}$ be a set of functions mapping $\{0,1\}^n$ to $\mathbb{R}$, and assume that each $\phi_i \in \mathcal{F}$ is efficiently computable, in the sense that for any $x \in \{0,1\}^n$, $\phi_i(x)$ can be computed in time $\mathrm{poly}(n)$. Suppose that for every $c \in \mathcal{C}$, there exist coefficients $\alpha_i \in \mathbb{R}$ such that for all $x \in \{0,1\}^n$, $|c(x) - \sum_{\phi_i \in \mathcal{F}} \alpha_i \cdot \phi_i(x)| \leq \varepsilon$. Then there is an algorithm that takes as input a sample S of size $|S| = \mathrm{poly}(n, |\mathcal{F}|, 1/\varepsilon, \log(1/\delta))$ from an unknown distribution $\mathcal{D}$, and in time $\mathrm{poly}(|S|)$ outputs a hypothesis h such that, with probability at least $1-\delta$ over S, $\Pr_{(x,b)\sim\mathcal{D}}[h(x) \neq b] \leq \varepsilon$.*

A feature set $\mathcal{F}$ that is commonly used in applications of Lemma 23 is the set of all monomials whose degree is at most some bound d. Indeed, an immediate corollary of Lemma 23 is the following.

Corollary 24. *Suppose that for every $c \in \mathcal{C}$, the ε-approximate degree of c is at most d. Then for every $\delta > 0$, there is an algorithm running in time $\mathrm{poly}(n^d, 1/\varepsilon, \log(1/\delta))$ that agnostically learns $\mathcal{C}$ to error ε with respect to any (unknown) distribution $\mathcal{D}$ over $\{0,1\}^n \times \{0,1\}$.*

The best known algorithms for agnostically learning disjunctions and de Morgan formulas of linear size [KKMS08, Rei11] combine Corollary 24 with known approximate degree upper bounds for disjunctions and de Morgan formulas of bounded size. We use the same strategy: our results for agnostic learning (Theorem 7) follow from combining Corollary 24 with our new approximate degree upper bounds. Specifically, Theorem 5 shows that the ε-approximate degree of any LC^0_d circuit is at most $\tilde{O}(n^{1-2^{-d}} \log^{2^{-d}}(1/\varepsilon))$, yielding our new result for agnostically learning LC^0 circuits. Theorem 15 shows that AC^0 circuits of size s have ε-approximate degree $\tilde{O}(\sqrt{n} s^{1/2-2^{-d}} (\log(1/\varepsilon))^{2^{-d}})$, giving our new result for learning sublinear-size AC^0.

Furthermore, since our upper bound on the approximate degree of LC^0 circuits is nearly tight, new techniques will be needed to significantly surpass our results. In particular, new techniques will be needed to agnostically learn *all* of LC^0 in subexponential time. Theorem 6 implies that if $\mathcal{F}$ is the set of all monomials of at most a given degree d, then one cannot use Corollary 24 to learn LC^0_d in time less than $2^{n^{1-2^{-\Omega(\sqrt{d})}}}$. However, standard techniques [She11a] automatically generalize the lower bound of Theorem 6 from the feature set of low-degree monomials to *arbitrary feature sets*. Specifically, we obtain the following theorem.

Theorem 25. *Let $\mathcal{C} = \mathsf{LC}^0_d$, and let $\mathcal{F}^*$ denote the minimum size set of features such that each $c \in \mathcal{C}$ can be approximated point-wise to error $1/3$ by a linear combination of the features in $\mathcal{F}$. Then $|\mathcal{F}^*| \geq 2^{n^{1-2^{-\Omega(\sqrt{d})}}}$.*

For completeness, we provide the proof of Theorem 25 below.

Proof. For a matrix $F \in \{0,1\}^{N \times N}$, the ε-approximate rank of F, denoted $\mathrm{rank}_\varepsilon(F)$, is the least rank of a matrix $A \in \mathbb{R}^{N \times N}$ such that $|A_{ij} - F_{ij}| \leq \varepsilon$ for all $(i,j) \in [N] \times [N]$. Sherstov's pattern matrix method [She11a] allows one to translate in a black-box manner an approximate degree lower bound for a function f into an approximate rank lower bound for a related matrix F, called the pattern matrix of f.

Specifically, invoking Theorem 6, let f be the function in LC^0_{d-1} satisfying $\widetilde{\deg}(f) \geq D$ for some $D = n^{1-2^{-\Omega(\sqrt{d})}}$. Viewing F as a $2^{4n} \times 2^{4n}$ matrix in the natural way, the pattern matrix

method [She11a, Theorem 8.1] implies that the function $F\colon \{0,1\}^{4n} \times \{0,1\}^{4n} \to \{0,1\}$ given by $F(x,y) = f\left(\ldots, \vee_{j=1}^{4} (x_{i,j} \wedge y_{i,j}) \ldots\right)$ satisfies

$$\mathrm{rank}_{1/3}(F) \geq 2^{\Omega(D)}, \tag{21}$$

where the expression $\mathrm{rank}_{1/3}(F)$ views F as a $2^{4n} \times 2^{4n}$ matrix.

Let $\mathcal{F}^*$ be a feature set satisfying the hypothesis of Theorem 25, i.e., for every function $c\colon \{0,1\}^{4n} \to \{0,1\}$ in LC^0_d, there exist constants $\alpha_1, \ldots, \alpha_{|\mathcal{F}|}$ such that

$$|c(x) - \sum_{\phi_j \in \mathcal{F}} \alpha_j \phi_j(x)| \leq 1/3 \tag{22}$$

for all $x \in \{0,1\}^{4n}$. We claim that this implies that

$$\mathrm{rank}_{1/3}(F) \leq |\mathcal{F}^*|. \tag{23}$$

Theorem 25 then follows by combining Equation (23) with Equation (21).

Let us view each row i of F as a function F_i mapping $\{0,1\}^{4n} \to \{0,1\}$. Then clearly, if f is in LC^0_{d-1}, each row F_i is in LC^0_d. Hence, there exist constants $\alpha_{i,1}, \ldots, \alpha_{i,|\mathcal{F}|}$ such that

$$|F_i(x) - \sum_{\phi_j \in \mathcal{F}} \alpha_{i,j} \cdot \phi_j(x)| \leq 1/3 \text{ for all } x \in \{0,1\}^{4n}. \tag{24}$$

Let M denote the $2^{4n} \times |\mathcal{F}|$ matrix whose (i,j)'th entry is $\alpha_{i,j}$. And let R denote that $|\mathcal{F}| \times 2^{4n}$ matrix whose (j,x)'th entry is $\phi_j(x)$, where we associate x with an input in $\{0,1\}^{4n}$. Then Equation (24) implies that $|M \cdot R - F_{ij}| \leq 1/3$ for all $(i,j) \in [2^{4n}] \times [2^{4n}]$. Since $M \cdot R$ is a matrix of rank at most $|\mathcal{F}|$, Equation (23) follows. □

6 Circuit Lower Bounds (Proof of Theorem 8)

In this section, we view Boolean functions as mapping domain $\{-1,1\}^n$ to $\{-1,1\}$. Recall that

$$\mathsf{IP}(x,y) = \oplus_{i=1}^{n}(x_i \wedge y_i)$$

denotes the Boolean inner product on $2n$ bits. As a warmup, we start by establishing a worst-case version of Theorem 8.

Proposition 26. *The Inner Product function cannot be computed by any depth-$(d+1)$ $\mathsf{AC}^0 \circ \oplus$ circuit of size $\widetilde{\Omega}(n^{1/(1-2^{-d})})$.*

Proof. Theorem 5 shows that any depth-d AC^0 circuit of size $s \geq n$ on n inputs has approximate degree at most $D = \tilde{O}(s^{1-2^{-d}})$. Clearly, the approximating polynomial has at most $\binom{s}{\leq D} \leq s^D$ many monomials.

From this, one can conclude that any depth-$(d+1)$ $\mathsf{AC}^0 \circ \oplus$ circuit $\mathcal{C}$ on n inputs of size $s \geq n$ can be approximated by a polynomial p over $\{-1,1\}^n$ with at most $\binom{s}{D}$ many monomials. To see why, let us write $\mathcal{C}(x,y) = \mathcal{C}'(h_1(x,y), \ldots, h_N(x,y))$, where $N \leq s$, $\mathcal{C}'$ is an AC^0 circuit of depth d and size at most s, and each h_i is a parity function. Since $\mathcal{C}'$ is an AC^0 circuit of depth d and size at most s on $N \leq s$ inputs, it has approximate degree at most D. Accordingly,

let q be a polynomial of degree at most D that point-wise approximates $\mathcal{C}'$ to error at most $1/3$. Now obtain p by replacing the i'th input to q with the corresponding parity gate, namely h_i, of $\mathcal{C}$. This yields a polynomial p that point-wise approximates $\mathcal{C}$ to error at most $1/3$, i.e., $|p(x,y) - \mathcal{C}(x,y)| \leq 1/3$ for all $(x,y) \in \{-1,1\}^n \times \{-1,1\}^n$. Since q is defined over domain $\{-1,1\}^N$, replacing any number of inputs to q with parity functions preserves the number of monomials of q.

On the other hand, it is known that that any polynomial p over $\{-1,1\}^n \times \{-1,1\}^n$ that point-wise approximates the Inner Product function to any error strictly less than 1 requires $2^{\Omega(n)}$ many monomials [BS92].

Combining the above two facts means that s^D must be at least $2^{\Omega(n)}$, which means that s must be at least $\tilde{\Omega}(n^{1/(1-2^{-d})})$. □

We now prove Theorem 8, restated here for convenience.

Theorem 8. *For any constant integer $d \geq 4$, any depth-$(d+1)$ $\mathsf{AC}^0 \circ \oplus$ circuit computing the Inner Product function on n bits on greater than a $1/2 + n^{-\log n}$ fraction of inputs has size $\tilde{\Omega}(n^{1/(1-2^{-d})}) = n^{1+2^{-d}+\Omega(1)}$.*

Proof Outline. The proof follows a similar outline to Proposition 26, but builds on an observation of Tal [Tal16, Lemma 4.2]. Roughly, Lemma 4.2 of [Tal16] shows that bipartite de Morgan formulas of size s cannot compute the Inner Product function on more than a $1/2 + n^{-\log n}$ fraction of inputs unless they have size at least roughly n^2. The only property of de Morgan formulas of size $\ll n^2$ that Tal uses is that they have sublinear approximate degree.

Similarly, Theorem 5 shows that an AC^0 circuit of size s and depth d on n inputs, for which $n \leq s \ll n^{1/(1-2^{-d})}$, has sublinear approximate degree.

Any parity function is an example of a bipartite function of size $O(1)$, meaning that the parity function applied to some subset of an input $(x,y) \in \{-1,1\}^n \times \{-1,1\}^n$ is computable by a constant-sized circuit with leaves computing a function of only x or y. Hence, Tal's argument applies with cosmetic changes not only to sub-quadratic size bipartite de Morgan formulas, but also to $\mathsf{AC}^0 \circ \oplus$ circuits of size $s \ll n^{1/(1-2^{-d})}$.

We remark that the entire argument (and hence the lower bound of Theorem 8 itself) applies not only to $\mathsf{AC}^0 \circ \oplus$ circuits, but more generally to depth-d AC^0 circuits augmented with a layer of low-communication gates above the inputs; we omit this extension for brevity.

Proof of Theorem 8, closely following the proof of Lemma 4.2 of [Tal16]. Let $\mathcal{C}\colon \{-1,1\}^{2n} \to \{-1,1\}$ be an $\mathsf{AC}^0 \circ \oplus$ circuit of depth $(d+1)$ and size $s \geq n$, and let

$$q = \Pr_{x,y \in \{-1,1\}^n}[\mathcal{C}(x,y) = \mathsf{IP}(x,y)].$$

Suppose that $q \geq 1/2 + \varepsilon$. Our goal is to show that s must be large, even for negligible values of ε.

Let $N \leq s$ denote the number of parity gates in $\mathcal{C}$, with the ith parity gate denoted by $h_i(x)\colon \{-1,1\}^n \to \{-1,1\}$. Then we may write $\mathcal{C}(x,y) = \mathcal{C}'(h_1(x,y), \ldots, h_N(x,y))$, where $\mathcal{C}'$ is an AC^0 circuit on at most s inputs, of depth d and size at most s. By Theorem 5, there exists a polynomial p of degree at most $D \leq \tilde{O}\left(s^{1-2^{-d}} \log^{2^{-d}}(1/\varepsilon)\right)$ such that, for all $w \in \{-1,1\}^N$, $|p(w) - \mathcal{C}'(w)| \leq \varepsilon$.

Next, we show that under the uniform distribution, the function $\mathsf{IP}(x,y)$ correlates well with $p(h_1(x),\ldots,h_N(x))$. We decompose the expectation $\mathbf{E}_{x,y\in\{-1,1\}^n}[p(x,y)\cdot\mathsf{IP}(x,y)]$ according to whether or not $\mathsf{IP}(x,y)=\mathcal{C}(x,y)$:

$$\begin{aligned}
&\mathbf{E}_{x,y\in\{-1,1\}^n}[p(h_1(x,y),\ldots,h_N(x,y))\cdot\mathsf{IP}(x,y)] = \\
&\mathbf{E}_{x,y\in\{-1,1\}^n}[p(h_1(x,y),\ldots,h_N(x,y))\cdot\mathsf{IP}(x,y)|\mathsf{IP}(x,y)=\mathcal{C}(x,y)]\cdot\Pr[\mathsf{IP}(x,y)=\mathcal{C}(x,y)]+ \\
&\mathbf{E}_{x,y\in\{-1,1\}^n}[p(h_1(x,y),\ldots,h_N(x,y))\cdot\mathsf{IP}(x,y)|\mathsf{IP}(x,y)\neq\mathcal{C}(x,y)]\cdot\Pr[\mathsf{IP}(x,y)\neq\mathcal{C}(x,y)] \\
&\qquad\geq (1-\varepsilon)\cdot q+(-1-\varepsilon)\cdot(1-q) \\
&\qquad= 2q-1-\varepsilon\geq 2\cdot(1/2+\varepsilon)-1-\varepsilon=\varepsilon.
\end{aligned} \tag{25}$$

Next, we write $p(z)$ as a multi-linear polynomial: $p(z)=\sum_{S\subseteq[N],|S|\leq D}\hat{p}(S)\cdot\prod_{i\in S}z_i$. Since $\hat{p}(S)=\mathbf{E}_{z\in\{-1,1\}^N}[p(z)\cdot\prod_{i\in S}z_i]$, we have that $|\hat{p}(S)|\leq 1+\varepsilon$ for every S. Note that there are at most $\binom{N}{\leq D}$ monomials in p. Invoking Inequality (25), we have:

$$\begin{aligned}
\varepsilon &\leq \mathbf{E}_{x,y\in\{-1,1\}^n}\left[p(h_1(x,y),\ldots,h_N(x,y))\cdot\mathsf{IP}(x,y)\right] \\
&= \mathbf{E}_{x,y\in\{-1,1\}^n}\left[\sum_{S\subseteq[N],|S|\leq D}\hat{p}(S)\prod_{i\in S}h_i(x,y)\cdot\mathsf{IP}(x,y)\right] \\
&= \sum_{S\subseteq[N],|S|\leq D}\hat{p}(S)\cdot\mathbf{E}_{x,y\in\{-1,1\}^n}\left[\prod_{i\in S}h_i(x,y)\cdot\mathsf{IP}(x,y)\right] \\
&\leq \sum_{S\subseteq[N],|S|\leq D}(1+\varepsilon)\left|\mathbf{E}_{x,y\in\{-1,1\}^n}\left[\prod_{i\in S}h_i(x,y)\cdot\mathsf{IP}(x,y)\right]\right|.
\end{aligned}$$

Hence there must exist a set $S\subseteq[N]$ with size at most D such that

$$\left|\mathbf{E}_{x,y\in\{-1,1\}^n}\left[\prod_{i\in S}h_i(x,y)\cdot\mathsf{IP}(x,y)\right]\right|\geq\frac{\varepsilon}{\binom{N}{\leq D}(1+\varepsilon)}\geq(\varepsilon/2)\cdot s^{-D}\geq\exp\left(\tilde{O}(-s^{1-2^{-d}}\log^{2^{-d}}(1/\varepsilon))\right).$$

It is well-known that IP is $2^{-\Omega(n)}$ correlated with any parity function h_i (indeed, IP on $2n$ bits is a *bent* function, meaning that all its Fourier coefficients have magnitude 2^{-n}, and hence its correlation with any parity is at most 2^{-n}). We conclude that

$$s^{1-2^{-d}}\log^{2^{-d}}(1/\varepsilon)\geq\tilde{\Omega}(n).$$

The theorem is an immediate consequence of this inequality. □

Acknowledgements

We thank Nikhil Mande, Ronald de Wolf, and Shuchen Zhu for comments on earlier drafts of this paper. R.K. thanks Luke Schaeffer for comments on the proof of Theorem 1.

References

[ABG+14] Adi Akavia, Andrej Bogdanov, Siyao Guo, Akshay Kamath, and Alon Rosen. Candidate weak pseudorandom functions in $AC^0 \circ \mathsf{MOD}_2$. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, ITCS '14, pages 251–260, 2014. doi:10.1145/2554797.2554821. [p. 8]

[ABO84] Miklos Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing*, STOC '84, pages 471–474, 1984. doi:10.1145/800057.808715. [p. 19]

[ACR+10] Andris Ambainis, Andrew M. Childs, Ben W. Reichardt, Robert Špalek, and Shengyu Zhang. Any AND-OR formula of size N can be evaluated in time $N^{1/2+o(1)}$ on a quantum computer. *SIAM Journal on Computing*, 39(6):2513–2530, 2010. doi:10.1137/080712167. [p. 5]

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933. [pp. 3, 11]

[BBC+01] Robert Beals, Harry Buhrman, Richard Cleve, Michele Mosca, and Ronald de Wolf. Quantum lower bounds by polynomials. *Journal of the ACM*, 48(4):778–797, 2001. doi:10.1145/502090.502097. [p. 4]

[BBHT98] Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. *Fortschritte der Physik*, 46(4-5):493–505, 1998. doi:10.1002/(SICI)1521-3978(199806)46:4/5<493::AID-PROP493>3.0.CO;2-P. [p. 11]

[BCdWZ99] Harry Buhrman, Richard Cleve, Ronald de Wolf, and Christof Zalka. Bounds for small-error and zero-error quantum algorithms. In *40th Annual Symposium on Foundations of Computer Science*, pages 358–368, 1999. doi:10.1109/sffcs.1999.814607. [pp. 17, 19]

[BCH+17] Adam Bouland, Lijie Chen, Dhiraj Holden, Justin Thaler, and Prashant Nalini Vasudevan. On the power of statistical zero knowledge. In *58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 708–719, 2017. doi:10.1109/focs.2017.71. [p. 19]

[BdW02] Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: a survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X. [pp. 11, 16]

[BKT18] Mark Bun, Robin Kothari, and Justin Thaler. The polynomial method strikes back: Tight quantum query bounds via dual polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 297–310, 2018. doi:10.1145/3188745.3188784. [pp. 5, 6, 10, 20]

[BKT19] Mark Bun, Robin Kothari, and Justin Thaler. Quantum algorithms and approximating polynomials for composed functions with shared inputs. In *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 662–678, 2019. doi:10.1137/1.9781611975482.42. [pp. 1, 8, 10]

[BM12] Paul Beame and Widad Machmouchi. The quantum query complexity of AC^0. *Quantum Information & Computation*, 12(7-8):670–676, 2012. [pp. 5, 6, 9]

[BNRdW07] Harry Buhrman, Ilan Newman, Hein Röhrig, and Ronald de Wolf. Robust polynomials and quantum algorithms. *Theory of Computing Systems*, 40(4):379–395, 2007. doi:10.1007/s00224-006-1313-z. [p. 17]

[BS92] Jehoshua Bruck and Roman Smolensky. Polynomial threshold functions, AC^0 functions, and spectral norms. *SIAM Journal on Computing*, 21(1):33–42, 1992. doi:10.1137/0221003. [p. 26]

[BT17] Mark Bun and Justin Thaler. A nearly optimal lower bound on the approximate degree of AC^0. In *IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS 2017)*, pages 1–12, 2017. doi:10.1109/FOCS.2017.10. [p. 6]

[BT21] Mark Bun and Justin Thaler. Guest column: Approximate degree in classical and quantum computing. *SIGACT News*, 51(4):48–72, January 2021. doi:10.1145/3444815.3444825. [p. 16]

[CCJYM09] Andrew M. Childs, Richard Cleve, Stephen P. Jordan, and David Yonge-Mallo. Discrete-query quantum algorithm for NAND trees. *Theory of Computing*, 5:119–123, 2009. doi:10.4086/toc.2009.v005a005. [p. 5]

[CGJ+16] Mahdi Cheraghchi, Elena Grigorescu, Brendan Juba, Karl Wimmer, and Ning Xie. $AC^0 \circ MOD_2$ lower bounds for the Boolean inner product. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 55. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.35. [p. 8]

[CIP09] Chris Calabro, Russell Impagliazzo, and Ramamohan Paturi. The complexity of satisfiability of small depth circuits. In *International Workshop on Parameterized and Exact Computation*, pages 75–85, 2009. doi:10.1007/978-3-642-11269-0_6. [p. 6]

[CKK12] Andrew M. Childs, Shelby Kimmel, and Robin Kothari. The quantum query complexity of read-many formulas. In *20th Annual European Symposium on Algorithms (ESA 2012)*, pages 337–348, 2012. doi:10.1007/978-3-642-33090-2_30. [pp. 4, 6, 7, 9]

[CR96] Shiva Chaudhuri and Jaikumar Radhakrishnan. Deterministic restrictions in circuit complexity. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, STOC '96, pages 30–36, 1996. doi:10.1145/237814.237824. [p. 10]

[CS16] Gil Cohen and Igor Shinkar. The complexity of DNF of parities. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 47–58, 2016. doi:10.1145/2840728.2840734. [p. 8]

[DRG17] Ning Ding, Yanli Ren, and Dawu Gu. PAC learning depth-3 AC^0 circuits of bounded top fanin. In *Proceedings of the 28th International Conference on Algorithmic Learning Theory*, volume 76 of *Proceedings of Machine Learning Research*, pages 667–680, 2017. URL: http://proceedings.mlr.press/v76/ding17a.html. [p. 23]

[ER21] Michael Ezra and Ron D. Rothblum. Small circuits imply efficient Arthur-Merlin protocols. Technical Report TR21-127, Electronic Colloquium on Computational Complexity (ECCC), 2021. URL: https://eccc.weizmann.ac.il/report/2021/127/. [p. 8]

[FGG08] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum algorithm for the Hamiltonian NAND tree. *Theory of Computing*, 4(1):169–190, 2008. doi:10.4086/toc.2008.v004a008. [p. 5]

[FHH+14] Yuval Filmus, Hamed Hatami, Steven Heilman, Elchanan Mossel, Ryan O'Donnell, Sushant Sachdeva, Andrew Wan, and Karl Wimmer. Real analysis in computer science: A collection of open problems, 2014. URL: https://simons.berkeley.edu/sites/default/files/openprobsmerged.pdf. [p. 10]

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 212–219, 1996. doi:10.1145/237814.237866. [pp. 3, 10, 11, 18]

[HLŠ07] Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Symposium on Theory of Computing (STOC 2007)*, pages 526–535, 2007. doi:10.1145/1250790.1250867. [p. 2]

[KKMS08] Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM Journal on Computing*, 37(6):1777–1805, 2008. doi:10.1137/060649057. [pp. 5, 7, 22, 23, 24]

[KKO13] Adam Klivans, Pravesh Kothari, and Igor C. Oliveira. Constructing hard functions using learning algorithms. In *IEEE Conference on Computational Complexity (CCC 2013)*, pages 86–97, 2013. doi:10.1109/CCC.2013.18. [p. 23]

[KLPT06] Michal Koucký, Clemens Lautemann, Sebastian Poloczek, and Denis Therien. Circuit lower bounds via Ehrenfeucht-Fraisse games. In *21st Annual IEEE Conference on Computational Complexity (CCC 2006)*, pages 190–201, 07 2006. doi:10.1109/CCC.2006.12. [p. 6]

[Kop13] Swastik Kopparty. AC^0 lower bounds and pseudorandomness. Lecture notes of "Topics in Complexity Theory and Pseudorandomness (Spring 2013)" at Rutgers University. http://sites.math.rutgers.edu/~sk1233/courses/topics-S13/lec4.pdf (Retrieved July 12, 2018), 2013. [p. 19]

[Kou09] Michal Koucký. Circuit complexity of regular languages. *Theory of Computing Systems*, 45(4):865–879, 2009. doi:10.1007/s00224-009-9180-z. [p. 6]

[KS04] Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{O}(n^{1/3})}$. *Journal of Computer and System Sciences*, 68(2):303–318, 2004. doi:10.1016/j.jcss.2003.07.007. [p. 23]

[KSS94] Michael J. Kearns, Robert E. Schapire, and Linda M. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994. doi:10.1007/bf00993468. [pp. 7, 22]

[LBW95] Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. On efficient agnostic learning of linear combinations of basis functions. In *Proceedings of the eighth*

annual conference on Computational learning theory, pages 369–376, 1995. doi:10.1145/225298.225343. [pp. 7, 22]

[Lee12] Troy Lee. Slides for the paper "improved quantum query algorithms for triangle finding and associativity testing" by T. Lee, F. Magniez, M. Santha. Available at http://research.cs.rutgers.edu/~troyjlee/troy_triangle.pdf (Retrieved July 11, 2018), 2012. [p. 9]

[LS09] Troy Lee and Adi Shraibman. Lower bounds in communication complexity. *Foundations and Trends in Theoretical Computer Science*, 3(4):263–399, 2009. doi:10.1561/0400000040. [p. 16]

[Nis11] Noam Nisan. Shortest formula for an n-term monotone CNF. Theoretical Computer Science Stack Exchange, 2011. https://cstheory.stackexchange.com/q/7087 (version: 2011-06-23). [p. 23]

[NS94] Noam Nisan and Mario Szegedy. On the degree of Boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994. doi:10.1007/BF01263419. [p. 17]

[OS17] Igor C. Carboni Oliveira and Rahul Santhanam. Conspiracies Between Learning Algorithms, Circuit Lower Bounds, and Pseudorandomness. In *32nd Computational Complexity Conference (CCC 2017)*, volume 79 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 18:1–18:49, 2017. doi:10.4230/LIPIcs.CCC.2017.18. [p. 23]

[Raz87] 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. [p. 8]

[Rei11] Ben Reichardt. Reflections for quantum query algorithms. In *SODA '11: Proceedings of the 22nd ACM-SIAM Symposium on Discrete Algorithms*, pages 560–569, 2011. doi:10.1137/1.9781611973082.44. [pp. 2, 5, 7, 22, 24]

[RS10] Alexander A. Razborov and Alexander A. Sherstov. The sign-rank of AC^0. *SIAM Journal on Computing*, 39(5):1833–1855, 2010. doi:10.1137/080744037. [p. 23]

[RW91] Prabhakar Ragde and Avi Wigderson. Linear-size constant-depth polylog-threshold circuits. *Information Processing Letters*, 39(3):143–146, 1991. doi:10.1016/0020-0190(91)90110-4. [p. 6]

[She11a] Alexander A. Sherstov. The pattern matrix method. *SIAM Journal on Computing*, 40(6):1969–2000, 2011. doi:10.1137/080733644. [pp. 7, 24, 25]

[She11b] Alexander A. Sherstov. Strong direct product theorems for quantum communication and query complexity. In *Proceedings of the 43rd annual ACM symposium on Theory of computing (STOC 2011)*, pages 41–50, 2011. doi:10.1145/1993636.1993643. [p. 5]

[She13a] Alexander A. Sherstov. The intersection of two halfspaces has high threshold degree. *SIAM Journal on Computing*, 42(6):2329–2374, 2013. doi:10.1137/100785260. [p. 20]

[She13b] Alexander A. Sherstov. Making polynomials robust to noise. *Theory of Computing*, 9:593–615, 2013. doi:10.4086/toc.2013.v009a018. [p. 5]

[She15] Alexander A. Sherstov. The power of asymmetry in constant-depth circuits. In *IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 431–450, 2015. doi:10.1109/FOCS.2015.34. [pp. 5, 6, 9]

[She18] Alexander A. Sherstov. Algorithmic polynomials. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, 2018. doi:10.1145/3188745.3188958. [p. 5]

[SS12] Rahul Santhanam and Srikanth Srinivasan. On the limits of sparsification. In *International Colloquium on Automata, Languages, and Programming*, pages 774–785. Springer, 2012. doi:10.1007/978-3-642-31594-7_65. [p. 6]

[ST17] Rocco A. Servedio and Li-Yang Tan. What Circuit Classes Can Be Learned with Non-Trivial Savings? In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 30:1–30:21, 2017. doi:10.4230/LIPIcs.ITCS.2017.30. [p. 23]

[SV12] Rocco A Servedio and Emanuele Viola. On a special case of rigidity. Technical Report TR12-144, Electronic Colloquium on Computational Complexity (ECCC), 2012. URL: https://eccc.weizmann.ac.il/report/2012/144/. [p. 8]

[Tal16] Avishay Tal. The bipartite formula complexity of inner-product is quadratic. Technical Report TR16-181, Electronic Colloquium on Computational Complexity (ECCC), 2016. URL: https://eccc.weizmann.ac.il/report/2016/181/. [p. 26]

[Tal17] Avishay Tal. Formula lower bounds via the quantum method. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2017)*, pages 1256–1268, 2017. doi:10.1145/3055399.3055472. [p. 5]

[Tal18] Avishay Tal. Personal communication, 2018. [p. 23]

[Val84] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, Nov 1984. doi:10.1145/1968.1972. [pp. 21, 22]

A Depth-vs-Size Upper Bounds for Approximate Majorities

In this section, we prove a result alluded to in Section 1.6, namely an upper bound on the size of small-depth circuits computing approximate majorities.

For $0 < p < q < 1$, we use $\mathsf{AMAJ}_{n,p,q}$ to refer to any total function f on n bits satisfying the following two properties:

- $f(x) = 0$ for all x of Hamming weight at most pn.
- $f(x) = 1$ for all x of Hamming weight at least qn.

We also say that such an f computes $\mathsf{AMAJ}_{n,p,q}$.

Theorem 27. *For any constant $\delta > 0$, there are positive constants $0 < p < 1/2 < q < 1$ such that the following holds. There is a total function f computing $\mathsf{AMAJ}_{n,p,q}$ such that f is itself computable by a depth-$(2d+3)$ circuit of size $O_{d,\delta}(n^{1+2^{-d}+\delta})$. Here, the $O_{d,\delta}$ notation hides a leading factor that depends only on d and $1/\delta$.*

Proof. A careful application of well-known arguments yields the following claim, whose proof we defer to the end of this section.

Claim 28. *For any constant $\delta > 0$, there exist constants $0 < p_0 < 1/2 < q_0 < 1$ such that the following holds. There is a monotone depth-3 AC^0 circuit C_0 of size $O(m^{2+\delta})$ computing $\mathsf{AMAJ}_{m,p_0,q_0}$. The top and bottom layers of gates of C_0 are AND gates.*

To prove Theorem 27, we need to use C_0 to construct deeper but smaller circuits that also compute approximate majorities. More generally, we establish the following iterative transformation that takes any circuit C_i computing an approximate majority function and turns it into a smaller and only slightly deeper circuit C_{i+1} that also computes an approximate majority function.[10]

Lemma 29. *Let $d > 0$ and $\delta > 0$ be fixed constants, and let p_0 and q_0 be the associated constants from Claim 28. Suppose that there exists a family of monotone circuits $\mathcal{C}_i$ with the following properties.*

- *There exist constants p_i and q_i satisfying $0 < p_i \leq p_0 < 1/2 < q_0 \leq q_i < 1$, such that for all input sizes m, $\mathcal{C}_i$ contains a circuit computing $\mathsf{AMAJ}_{m,p_i,q_i}$.*
- *Each circuit in $\mathcal{C}_i$ has depth $2i + 3$, and the top and bottom layers consist of AND gates.*
- *There is a constant $k_i > 0$ such that the circuit in $\mathcal{C}_i$ defined over inputs of size m has size at most $O_{d,p_i,q_i}(m^{k_i+\delta})$, where the O_{d,p_i,q_i} notation hides factors depending only on d, p_i, and q_i.*

Then there exists a family of monotone circuits $\mathcal{C}_{i+1}$ satisfying the above three properties, with $p_{i+1} = (1 - 1/(10d))p_i$, $q_{i+1} = 1 - (1 - q_i)(1 - 1/(10d))$, and $k_{i+1} \leq (1 + k_i)/2$.

Proof. Let C_i be the assumed circuit from family $\mathcal{C}_i$ on m inputs computing $\mathsf{AMAJ}_{m,p_i,q_i}$. Let

$$n = m^2$$

and

$$M = 700d^2(1/p_i^2 + 1/q_i^2)m.$$

Consider generating a circuit C_{i+1} on n inputs via the following random process. C_{i+1} will have the form

$$\mathsf{AMAJ}_{M,p_i,q_i}(\mathsf{AMAJ}_{m,p_i,q_i}, \ldots, \mathsf{AMAJ}_{m,p_i,q_i}), \tag{26}$$

Here, p_i and q_i are fixed constants as per the statement of the lemma, and each of the bottom AMAJ circuits are evaluated on a randomly chosen (size-m) subset of the n inputs of C_{i+1}. Since $p_i \leq p_0$ and $q_0 \leq q_i$, we may use a circuit C_0 from family $\mathcal{C}_0$ (as per Claim 28) to compute the outer function $\mathsf{AMAJ}_{M,p_i,q_i}$. We use the circuit C_i from family $\mathcal{C}_i$ to compute each copy of the inner function $\mathsf{AMAJ}_{m,p_i,q_i}$.

In summary, we generate the circuit C_{i+1} to be the composition $C_0 \circ C_i$, but where each copy of C_i is evaluated over a randomly chosen (size-m) subset of the n inputs of C_{i+1} (i.e., C_{i+1} is a shared-input composition of C_0 and C_i).

[10] Not coincidentally, this iterative transformation to reduce circuit size at the expense of depth is reminiscent of the transformation used to prove the approximate degree lower bound for linear-size circuits given in Theorem 6.

We claim that with strictly positive probability, this circuit C_{i+1} computes $\mathsf{AMAJ}_{n,p_{i+1},q_{i+1}}$. To see this, first fix an input x with Hamming weight at most $p_{i+1} \cdot n$, so that the expected number of 1-inputs to any bottom $\mathsf{AMAJ}_{m,p_i,q_i}$ circuit is at most $\mu := p_{i+1} \cdot m$. Note that $p_i \cdot m > (1 + 1/(10d))\mu$. If any $\mathsf{AMAJ}_{m,p_i,q_i}$ circuit "makes an error" on x (i.e., evaluates to 1 on x), then at least $p_i \cdot m > (1 + 1/(10d)) \cdot \mu$ of the randomly chosen inputs to the gate are 1. By a Chernoff bound, for each of the bottom $\mathsf{AMAJ}_{m,p_i,q_i}$ gates, this happens on input x with probability at most $\exp(-\mu/(3(10d)^2)) \leq \exp(-\mu/(300d^2)) \leq \exp(-p_i m/(600d^2))$.

The probability that more than $(700d^2/p_i)m \leq p_i \cdot M$ of these circuits makes an error is at most $2^M \cdot (\exp(-p_i m/(600d^2)))^{(700d^2/p_i)m} \ll \exp(-m^2)$. Thus, with probability at least $1 - \exp(-m^2)$, the circuit C_{i+1} outputs 0 on input x.

An analogous argument holds for inputs x with Hamming weight at least $q_{i+1} \cdot n$, so by a union bound over all at most the 2^n inputs to C_{i+1} with Hamming weight at most $p_{i+1} \cdot n$ or at least $q_{i+1} \cdot n$, with strictly positive probability C_{i+1} computes $\mathsf{AMAJ}_{n,p_{i+1},q_{i+1}}$.

The circuit C_{i+1} has m^2 inputs and has size at most

$$O(M^{2+\delta}) + O(M \cdot m^{k_i+\delta}) = O_{d,p_i,q_i}(n^{1+\delta/2} + m^{1+k_i+\delta}) = O_{d,p_i,q_i}(n^{k_{i+1}+\delta/2}),$$

where recall that $k_{i+1} = (1 + k_i)/2$.

Equation (26) implies that the top and bottom layers of C_{i+1} consist of AND gates, with C_{i+1} inheriting this property directly from C_i and C_0. Moreover, by collapsing the bottom layer of C_0 with the top layer of each copy of C_i (which is possible because C_0 is monotone), we find that the depth of C_{i+1} is at most most $3 + (2i + 3) - 1 = 2(i + 1) + 3$. This completes the proof of the lemma. □

Let p_0, q_0 be as in Claim 28, and let $p = p_0/e$ and $q = 1 - (1 - q_0)/e$. Theorem 27 follows by iteratively applying Lemma 29 d times (starting with $i = 0$; the assumptions of the lemma are satisfied for this value of i by Claim 28) to conclude that $\mathsf{AMAJ}_{n,p,q}$ is computable by a circuit of depth $2d + 3$ and size $O_d(n^{1+2^{-d}+\delta})$. □

Proof of Claim 28. The main idea of the (probabilistic) construction is to have an AND-OR-AND circuit C, where the top AND gate has fan-in $t_1 := m$, the middle layer (of OR gates) all have fan-in $t_2 := m^{1+\delta}$, and the bottom layer of AND gates all have fan-in $t_3 = \log_2(m)$. Each bottom AND gate is connected to t_3 randomly chosen inputs.

Let p be any constant less than $1/2^{1+\delta}$, and $q = 1/2^\delta$. These choices ensure that $p^{\log_2(m)} < 1/(2m^{1+\delta})$ and $q^{\log_2(m)} > 1/m^\delta$. We now show that with positive probability, C computes $\mathsf{AMAJ}_{m,p,q}$.

Consider any m-bit input x with Hamming weight at most $p \cdot m$. Then for any fixed AND gate at the bottom layer of C, the probability the AND gate evaluates to 1 is at most $p^{t_3} < 1/(2m^{1+\delta})$. By a union bound, this implies that for any fixed OR gate at the middle layer of C, the probability the OR gate outputs 1 on x is at most $t_2 \cdot 1/(2m^{1+\delta}) \leq 1/2$. This implies that the probability the top AND gate outputs 1 on x is at most $1/2^{t_1} = 2^{-m}$.

Now consider any m-bit input x with Hamming weight at least $q \cdot m$. Then for any fixed AND gate at the bottom layer of C, the probability the AND gate evaluates to 1 is at least $q^{t_3} > 1/m^\delta$. This implies that for any fixed OR gate at the middle layer of C, the probability the OR gate outputs 1 on x is least $1 - (1 - 1/m^\delta)^{t_2} \geq 1 - e^{-m} \geq 1 - 1/(m2^m)$. This implies that the probability the top AND gate outputs 1 on x is at least $1 - 2^{-m}$.

By a union bound over all the at most 2^m inputs x to C, we conclude that with positive probability C computes $\mathsf{AMAJ}_{m,p,q}$. □