THEORY OF COMPUTING, Volume 19 (11), 2023, pp. 1–14
www.theoryofcomputing.org

A Strong XOR Lemma for Randomized Query Complexity

Joshua Brody Jae Tak Kim Peem Lerdputtipongporn
Hariharan Srinivasulu

Received August 3, 2020; Revised December 30, 2023; Published December 31, 2023

Abstract. We give a strong direct sum theorem for computing $\mathrm{XOR}_k \circ g$, the XOR of k instances of the partial Boolean function g. Specifically, we show that for every g and every $k \geq 2$, the randomized query complexity of computing the XOR of k instances of g satisfies $\overline{\mathrm{R}}_\varepsilon(\mathrm{XOR}_k \circ g) = \Theta(k\,\overline{\mathrm{R}}_{\frac{\varepsilon}{k}}(g))$, where $\overline{\mathrm{R}}_\varepsilon(f)$ denotes the expected number of queries made by the most efficient randomized algorithm computing f with ε error. This matches the naive success amplification upper bound and answers a conjecture of Blais and Brody (CCC'19).

As a consequence of our strong direct sum theorem, we give a total function g for which $\mathrm{R}(\mathrm{XOR}_k \circ g) = \Theta(k \log(k) \cdot \mathrm{R}(g))$, where $\mathrm{R}(f)$ is the number of queries made by the most efficient randomized algorithm computing f with $1/3$ error. This answers a question from Ben-David et al. (RANDOM'20).

1 Introduction

We show that XOR admits a strong direct sum theorem for randomized query complexity. Generally, the *direct sum problem* asks how the cost of computing a partial function g scales with the number k of instances of the function that we need to compute simultaneously

ACM Classification: F.1.1, F.1.3

AMS Classification: 68Q09,68Q10,68Q17

Key words and phrases: lower bounds, query complexity, direct sum

 DOI: 10.4086/toc.2023.v019a011

(in parallel) . This is a foundational computational problem that has received considerable attention [9, 2, 13, 14, 10, 6, 8, 7, 3, 4, 5], including recent a recent paper by Blais and Brody [7], which showed that *expected* query complexity obeys a direct sum theorem in a strong sense—computing k copies of a partial function g with overall error ε requires k times the cost of computing g on one input with very low (ε/k) error. This matches the naive success amplification algorithm which runs an $\frac{\varepsilon}{k}$-error algorithm for g once on each of k inputs and applies a union bound to get an overall error guarantee of ε.

What happens if we do not need to compute g on all instances, but only on a *function* $f \circ g$ of those instances? Clearly the same success amplification trick (compute g on each input with low error, then apply f to the answers) works for computing $f \circ g$; however, in principle, computing $f \circ g$ can be easier than computing each instance of g individually. When a function $f \circ g$ requires success amplification for all g, we say that f *admits a strong direct sum theorem*. Our main result shows that XOR admits a strong direct sum theorem.

1.1 Query complexity

A *query algorithm*, also known as a *decision tree*, computing f, is an algorithm $\mathcal{A}$ that takes an input x to f, examines (or *queries*) bits of x, and outputs an answer for $f(x)$. A *leaf* of $\mathcal{A}$ is a bit string $q \in \{0,1\}^*$ representing the answers to the queries made by $\mathcal{A}$ on input x. Let $\mathrm{leaf}(\mathcal{A}, x)$ denote the leaf of $\mathcal{A}$ reached on input x. Naturally, our general goal is to minimize the length of q, i. e., minimize the number of queries needed to compute f.

A randomized algorithm $\mathcal{A}$ *computes* a function $f : \{0,1\}^n \to \{0,1\}$ *with error* $\epsilon \geq 0$ if for every input $x \in \{0,1\}^n$, the algorithm outputs the value $f(x)$ with probability at least $1-\epsilon$. The *query cost* of $\mathcal{A}$ is the maximum number of bits of x that it queries, with the maximum taken over both the choice of input x and the internal randomness of $\mathcal{A}$. The *ϵ-error randomized query complexity* of f (also known as the *randomized decision tree complexity* of f) is the minimum query cost of an algorithm $\mathcal{A}$ that computes f with error at most ϵ. We denote this complexity by $\mathrm{R}_\epsilon(f)$, and we write $\mathrm{R}(f) := \mathrm{R}_{\frac{1}{3}}(f)$ to denote the $\frac{1}{3}$-error randomized query complexity of f.

Another natural measure for the query cost of a randomized algorithm $\mathcal{A}$ is the *expected* number of coordinates of an input x that it queries. Taking the maximum expected number of coordinates queried by $\mathcal{A}$ over all inputs yields the *expected query cost* of $\mathcal{A}$. The minimum expected query cost of an algorithm $\mathcal{A}$ that computes a function f with error at most ϵ is the *ϵ-error expected query complexity* of f, which we denote by $\overline{\mathrm{R}}_\epsilon(f)$. We again write $\overline{\mathrm{R}}(f) := \overline{\mathrm{R}}_{\frac{1}{3}}(f)$. Note that $\overline{\mathrm{R}}_0(f)$ corresponds to the standard notion of *zero-error randomized query complexity* of f.

1.2 Our results

Our main result is a strong direct sum theorem for XOR.

Theorem 1.1. *For every partial function $g : \{0,1\}^n \to \{0,1\}$ and all $\varepsilon > 0$, we have* $\overline{\mathrm{R}}_\varepsilon(\mathrm{XOR}_k \circ g) = \Omega(k \cdot \overline{\mathrm{R}}_{\varepsilon/k}(g))$.

This answers Conjecture 1 of Blais and Brody [7] in the affirmative. We prove Theorem 1.1 by proving an analogous result in distributional query complexity. We also allow our algorithms to *abort* with a given probability. Let μ be a distribution on valid inputs for f. Let $\mathrm{D}^{\mu}_{\delta,\varepsilon}(f)$ denote the minimal query cost of a deterministic query algorithm that aborts with probability at most δ and errs with probability at most ε, where the probability is taken over inputs $X \sim \mu$. Similarly, let $\mathrm{R}_{\delta,\varepsilon}(f)$ denote the minimal query cost of a randomized algorithm that computes f with abort probability at most δ and error probability at most ε for each valid input. (Here the probabilities are taken over the internal randomness of the algorithm.)

Our main technical result is the following strong direct sum result for $\mathrm{XOR}_k \circ g$ for distributional algorithms.

Lemma 1.2 (Main Technical Lemma, informally stated.)**.** *For every partial function $g : \{0,1\}^n \to \{0,1\}$, every distribution μ on the set of valid inputs and every sufficiently small $\delta, \varepsilon > 0$, we have*

$$\mathrm{D}^{\mu^k}_{\delta,\varepsilon}(\mathrm{XOR}_k \circ g) = \Omega(k\, \mathrm{D}^{\mu}_{\delta',\varepsilon'}(g)) ,$$

for $\delta' = \Theta(1)$ and $\varepsilon' = \Theta(\varepsilon/k)$.

In [7], Blais and Brody also gave a total function $g : \{0,1\}^n \to \{0,1\}$ whose ε-error expected query complexity satisfies $\overline{\mathrm{R}}_{\varepsilon}(g) = \Omega(\mathrm{R}(g) \cdot \log \frac{1}{\varepsilon})$. We use our strong XOR Lemma together with this function to show the following.

Corollary 1.3. *There exists a total function $g : \{0,1\}^n \to \{0,1\}$ such that*

$$\mathrm{R}_{\varepsilon}(\mathrm{XOR}_k \circ g) = \Omega(k \log(k) \cdot \mathrm{R}_{\varepsilon}(g)) .$$

Proof. Let $g : \{0,1\}^n \to \{0,1\}$ be a function guaranteed by [7]. Then, we have

$$\mathrm{R}(\mathrm{XOR}_k \circ g) \geq \overline{\mathrm{R}}(\mathrm{XOR}_k \circ g) \geq \Omega(k \cdot \overline{\mathrm{R}}_{1/3k}(g)) \geq \Omega(k \cdot \mathrm{R}(g) \cdot \log(3k)) = \Omega(k \log(k) \cdot \mathrm{R}(g)) ,$$

where the second inequality is by Theorem 1.1 and the third inequality is from the query complexity guarantee of g. □

This answers Open Question 1 from a recent paper by Ben-David et al. [5].

1.3 Previous and related work

Jain et al. [10] gave direct sum theorems for deterministic and randomized query complexity. In particular, Jain et al. show $\mathrm{R}_{\varepsilon}(f^k) \geq \delta \cdot k \cdot \mathrm{R}_{\varepsilon+\delta}(f)$. While their direct sum result holds for randomized query complexity, the lower bound is in terms of the query complexity of computing f with an *increased error* of $\varepsilon + \delta$. This weakens the right-hand side of their inequality. Shaltiel [14] gave a function f such that $\mathrm{D}^{\mu^k}_{0,\varepsilon}(f^k) \ll k\, \mathrm{D}^{\mu}_{0,\varepsilon}(f)$, thus showing that a similar direct sum theorem fails to hold for distributional complexity.

Drucker [8] gave a *direct product theorem* for randomized query complexity, showing that any algorithm computing g^k using $\alpha k\, \mathrm{R}(g)$ queries for a constant $\alpha < 1$ has success probability exponentially small in k. Drucker also gave the following XOR Lemma, showing that any algorithm for $\mathrm{XOR}_k \circ g$ that makes $\ll kR(g)$ queries has success probability exponentially close to 1/2 [8, Theorem 1.3].

Theorem 1.4 (Drucker). *Suppose any randomized T-query algorithm has success probability $\leq 1-\varepsilon'$ in computing the Boolean function g on input $x \sim \mu$ for some input distribution μ. Then, for all $0 < \alpha < 1$, any randomized algorithm making $\alpha\varepsilon' T k$ queries to compute $\mathrm{XOR}_k \circ g$ on input distribution μ^k (k inputs drawn independently from μ) has success probability at most $\frac{1}{2}\left(1 + [1 - 2\varepsilon' + 6\alpha \ln(2/\alpha)\varepsilon']^k\right)$.*

Drucker's XOR Lemma applies to randomized query complexity $\mathrm{R}(\mathrm{XOR}_k \circ g)$, while ours applies to expected randomized query complexity $\overline{\mathrm{R}}(\mathrm{XOR}_k \circ g)$.

Note the ε' factor in the query complexity in Drucker's theorem. When ε' is a constant close to $1/2$, Drucker's lower bound is stronger than ours by a large constant factor. However, when $\varepsilon' = o(1)$, his bound degrades significantly. Couched in our notation, Drucker's XOR Lemma yields $\mathrm{R}_\varepsilon(\mathrm{XOR}_k \circ g) = \Omega(\varepsilon' k \, \mathrm{R}_{\varepsilon'}(g))$, for some $\varepsilon' = O(\varepsilon/k)$. This simplifies to $\mathrm{R}_\varepsilon(\mathrm{XOR}_k \circ g) = \Omega(\varepsilon R_{\varepsilon/k}(g))$, a loss of a factor of k.

As far as we know, it remains open whether this ε' factor is needed in the query complexity lower bound of Drucker's XOR Lemma. However, Shaltiel's counterexample [14] shows that the ε' factor is required for distributional query complexity. This rules out the most direct approach for proving a tighter XOR Lemma for $\mathrm{R}(\mathrm{XOR}_k \circ g)$.

Our paper is most closely related to that of Blais and Brody [7], who give a strong direct sum theorem for the expected query complexity of computing k copies of f in parallel, for any partial function f, and explicitly conjecture that XOR admits a strong direct sum theorem. Both [7] and our paper use techniques similar to work of Molinaro et al. [11, 12] who give strong direct sum theorems for communication complexity.

Our strong direct sum theorem for XOR is an example of a *composition theorem*—a lower bound on the query complexity of functions of the form $f \circ g$. Several recent articles study composition theorems in query complexity. Bassilakis et al. [1] show that $\mathrm{R}(f \circ g) = \Omega(\mathrm{fbs}(f)\,\mathrm{R}(g))$, where $\mathrm{fbs}(f)$ is the *fractional block sensitivity* of f. Ben-David and Blais [3, 4] give a tight lower bound on $\mathrm{R}(f \circ g)$ as a product of $\mathrm{R}(g)$ and a new measure they define called $\mathrm{noisyR}(f)$, which measures the complexity of computing f on noisy inputs. They also characterize $\mathrm{noisyR}(f)$ in terms of the gap-majority function. Ben-David et al [5] explicitly consider strong direct sum theorems for composed functions in randomized query complexity, asking whether the naive success amplification algorithm is necessary to compute $f \circ g$. They give a partial strong direct sum theorem, showing that there exists a partial function g such that computing $\mathrm{XOR}_k \circ g$ requires success amplification, even in a model where the abort probability may be arbitrarily close to 1.[1] Ben-David et al. explicitly ask whether there exists a total function g such that $\mathrm{R}(\mathrm{XOR}_k \circ g) = \Omega(k \log(k)\,\mathrm{R}(g))$.

1.4 Our technique

Our technique most closely follows the strong direct sum theorem of Blais and Brody. We start with a query algorithm that computes $\mathrm{XOR}_k \circ g$ and use it to build a query algorithm for computing g with low error. To do this, we will take an input for g and *embed* it into an input for $\mathrm{XOR}_k \circ g$. Given $x \in \{0,1\}^n$, $i \in [k]$, and $y \in \{0,1\}^{n \times k}$, let $y^{(i \leftarrow x)} := (y^{(1)}, \ldots, y^{(i-1)}, x, y^{(i+1)}, \ldots y^{(k)})$ denote

[1]In this query complexity model, called PostBPP, the query algorithm is allowed to abort with any probability strictly less than 1. When it does not abort, it must output f with probability at least $1 - \varepsilon$.

the input obtained from y by replacing the i-th coordinate $y^{(i)}$ with x. Note that if $x \sim \mu$ and $y \sim \mu^k$,[2] then $y^{(i \leftarrow x)} \sim \mu^k$ for all $i \in [k]$.

We require the following observation [8, Lemma 3.2].

Lemma 1.5 (Drucker). *Let $y \sim \mu^k$ be an input for a query algorithm $\mathcal{A}$, and consider any execution of queries by $\mathcal{A}$. The distribution of coordinates of y, conditioned on the queries made by $\mathcal{A}$, remains a product distribution.*

In particular, the answers to $g(y^{(i)})$ remain independent bits conditioned on any set of queries made by the query algorithm. Our first observation is that in order to compute $\mathrm{XOR}_k \circ g(y)$ with high probability, we must be able to compute $g(y^{(i)})$ with very high probability for many i's. The intuition behind this observation is captured by the following simple fact about the XOR of independent random bits.

Define the *bias* of a random bit $X \in \{0,1\}$ as $r(X) := \max_{b \in \{0,1\}} \Pr[X = b]$. Define the *advantage* of X as $\mathrm{adv}(X) := 2r(X) - 1$. Note that when $\mathrm{adv}(X) = \delta$, then $r(X) = \frac{1}{2}(1+\delta)$.

Fact 1.6. *Let $X_1, \dots, X_k$ be independent random bits, and let a_i be the advantage of X_i. Then,*

$$\mathrm{adv}(X_1 \oplus \cdots \oplus X_k) = \prod_{i=1}^{k} \mathrm{adv}(X_i) \, .$$

Proof. For each i, let $b_i := \operatorname{argmax}_{b \in \{0,1\}} \Pr[X_i = b]$ and $\delta_i := \mathrm{adv}(X_i)$. Then $\Pr[X_i = b_i] = \frac{1}{2}(1+\delta_i)$. We prove Fact 1.6 by induction on k. When $k = 1$, there is nothing to prove. For $k = 2$, note that

$$\begin{aligned}
\Pr[X_1 \oplus X_2 = b_1 \oplus b_2] &= \frac{1}{2}(1+\delta_1)\frac{1}{2}(1+\delta_2) + \frac{1}{2}(1-\delta_1)\frac{1}{2}(1-\delta_2) \\
&= \frac{1}{4}(1+\delta_1+\delta_2+\delta_1\delta_2) + \frac{1}{4}(1-\delta_1-\delta_2+\delta_1\delta_2) \\
&= \frac{1}{2}(1+\delta_1\delta_2) \, .
\end{aligned}$$

Hence $X_1 \oplus X_2$ has advantage $\delta_1\delta_2$ and the claim holds for $k = 2$. For an induction hypothesis, suppose that the claim holds for $X_1 \oplus \cdots \oplus X_{k-1}$. Then, setting $Y := X_1 \oplus \cdots \oplus X_{k-1}$, by the induction hypothesis, we have $\mathrm{adv}(Y) = \prod_{i=1}^{k-1} \mathrm{adv}(X_i)$. Moreover, $X_1 \oplus \cdots \oplus X_k = Y \oplus X_k$, and

$$\mathrm{adv}(X_1 \oplus \cdots \oplus X_k) = \mathrm{adv}(Y \oplus X_k) = \mathrm{adv}(Y)\,\mathrm{adv}(X_k) = \prod_{i=1}^{k} \mathrm{adv}(X_i) \, . \qquad \square$$

Given an algorithm for $\mathrm{XOR}_k \circ g$ that has error ε, it follows that for typical leaves the advantage of computing $\mathrm{XOR}_k \circ g$ is $\gtrsim 1 - 2\varepsilon$. Fact 1.6 shows that for such leaves, the advantage of computing $g(y^{(i)})$ for most coordinates i is $\gtrsim (1-2\varepsilon)^{1/k} = 1 - \Theta(\varepsilon/k)$. Thus, conditioned on

[2] We use μ^k to denote the distribution obtained on k-tuples of $\{0,1\}^n$ obtained by sampling each coordinate independently according to μ.

reaching this leaf of the query algorithm, we could compute $g(y^{(i)})$ with very high probability. We would like to fix a coordinate i^* such that for most leaves, our advantage in computing g on coordinate i^* is $1 - O(\varepsilon/k)$. There are other complications, namely that (i) our construction needs to handle aborts gracefully and (ii) our construction must ensure that the algorithm for $\mathrm{XOR}_k \circ g$ does not query the i^*-th coordinate too many times. Our construction identifies a coordinate i^* and a string $z \in \{0,1\}^{n\times k}$, and on input $x \in \{0,1\}^n$ it emulates a query algorithm for $\mathrm{XOR}_k \circ g$ on input $z^{(i^*\leftarrow x)}$ and outputs our best guess for $g(x)$ (which is now g evaluated on coordinate i^* of $z^{(i^*\leftarrow x)}$), aborting when needed e. g., when the algorithm for $\mathrm{XOR}_k \circ g$ aborts or when it queries too many bits of x. We defer full details of the proof to Section 2.

1.5 Preliminaries and notation

A partial Boolean function on the domain $\{0,1\}^n$ is a function $f : S \to \{0,1\}$ for some subset $S \subseteq \{0,1\}^n$. Call S the set of valid inputs for f. Let f be a partial Boolean function on $\{0,1\}^n$ and μ a distribution whose support is a subset of the valid inputs. We use $[n]$ to denote the set $\{1,\ldots,n\}$ and $X \in_R S$ to denote an element X sampled uniformly from a set S. Let μ^k denote the distribution obtained on k-tuples of $\{0,1\}^n$ obtained by sampling each coordinate independently according to μ.

An algorithm $\mathcal{A}$ is a $[q,\delta,\varepsilon,\mu]$-distributional query algorithm for f if $\mathcal{A}$ is a deterministic algorithm with query cost q that computes f with error probability at most ε and abort probability at most δ when the input x is drawn from μ.[3]

Our main theorem is a direct sum result for $\mathrm{XOR}_k \circ g$ for expected randomized query complexity; however, Lemma 1.2 uses distributional query complexity with aborts. To translate between the two, we need two results from Blais and Brody [7] that connect the query complexities in the randomized, expected randomized, and distributional query models.

Fact 1.7 ([7], Proposition 14)**.** *For every partial function $f : \{0,1\}^n \to \{0,1\}$, every $0 \le \epsilon < \frac{1}{2}$ and every $0 < \delta < 1$,*

$$\delta \cdot \mathrm{R}_{\delta,\varepsilon}(f) \le \overline{\mathrm{R}}_{\epsilon}(f) \le \tfrac{1}{1-\delta} \cdot \mathrm{R}_{\delta,(1-\delta)\epsilon}(f).$$

Fact 1.7 shows that when $\delta = 1 - \Omega(1)$, to achieve a lower bound for $\overline{\mathrm{R}}_{\varepsilon}(f)$, it suffices to lower bound $\mathrm{R}_{\delta,\varepsilon}(f)$. Next, we need the following generalization of Yao's minimax lemma, which connects randomized and distributional query complexity in the presence of aborts.

Fact 1.8 ([7], Lemma 15)**.** *For any $\alpha,\beta > 0$ such that $\alpha+\beta \le 1$, we have*

$$\max_{\mu} \mathrm{D}^{\mu}_{\delta/\alpha,\varepsilon/\beta}(f) \le \mathrm{R}_{\delta,\varepsilon}(f) \le \max_{\mu} \mathrm{D}^{\mu}_{\alpha\delta,\beta\varepsilon}(f).$$

For simplicity, it might be helpful to consider the simplest case where $\alpha = \beta = \frac{1}{2}$. In this case, we recover $\max_\mu \mathrm{D}^{\mu}_{2\delta,2\varepsilon}(f) \le \mathrm{R}_{\delta,\varepsilon}(f) \le \max_\mu \mathrm{D}^{\mu}_{\delta/2,\varepsilon/2}(f)$. Fact 1.8 shows that to prove a lower

[3]Note: in the literature, the error probability is sometimes defined as being conditioned on not aborting (e. g.,[5]). We define the error probabilty without conditioning to match article [7] most closely related to our work.

bound on $\mathrm{R}_{\delta,\epsilon}(f)$, it suffices to prove a lower bound on distributional complexity (albeit with a constant factor increase in abort and error probabilities).

We will also use the following convenient facts about expected value.

Fact 1.9 (Law of Conditional Expectations)**.** *Let X and Y be random variables. Then, we have*

$$\mathrm{E}[X] = \mathrm{E}[\mathrm{E}[X|Y]] .$$

Fact 1.10 (Markov Inequality for Bounded Variables)**.** *Let X be a real-valued random variable with $0 \le X \le 1$. Suppose that $\mathrm{E}[X] \ge 1-\varepsilon$. Then, for any $T > 1$ it holds that*

$$\Pr[X < 1 - T\varepsilon] < \frac{1}{T} .$$

Proof. Let $Y := 1 - X$. Then, $\mathrm{E}[Y] \le \varepsilon$. By Markov's Inequality we have

$$\Pr[X < 1 - T\varepsilon] = \Pr[Y > T\varepsilon] \le \frac{1}{T} . \qquad \square$$

2 Strong XOR lemma

In this section, we prove our main result.

Lemma 2.1 (Formal Restatement of Lemma 1.2)**.** *For every partial function $g : \{0,1\}^n \to \{0,1\}$, every distribution μ on $\{0,1\}^n$, every $0 \le \delta \le \frac{1}{5}$, and every $0 < \varepsilon \le \frac{1}{200}$, we have*

$$\mathrm{D}^{\mu^k}_{\delta,\varepsilon}(\mathrm{XOR}_k \circ g) \ge \frac{k}{25}\, \mathrm{D}^{\mu}_{\delta',\varepsilon'}(g) ,$$

$\delta' = 0.36 + 3\delta$ and $\varepsilon' = \frac{15000\varepsilon}{k}$.

Proof. Let $q := \mathrm{D}^{\mu^k}_{\delta,\varepsilon}(\mathrm{XOR}_k \circ g)$, and suppose that $\mathcal{A}$ is a $[q,\delta,\varepsilon,\mu^k]$-distributional query algorithm for $\mathrm{XOR}_k \circ g$. Our goal is to construct an $[O(q/k),\delta',\varepsilon',\mu]$-distributional query algorithm for g. Towards that end, for each leaf ℓ of $\mathcal{A}$ define

$$\begin{aligned}
b_\ell &:= \operatorname*{argmax}_{b\in\{0,1\}} \Pr_{x\sim\mu^k}[\mathrm{XOR}_k \circ g(x) = b \,|\, \mathrm{leaf}(\mathcal{A},x) = \ell] \\
r_\ell &:= \Pr_{x\sim\mu^k}[\mathrm{XOR}_k \circ g(x) = b_\ell \,|\, \mathrm{leaf}(\mathcal{A},x) = \ell] \\
a_\ell &:= 2r_\ell - 1 .
\end{aligned}$$

Call a_ℓ the *advantage* of $\mathcal{A}$ on leaf ℓ.

The purpose of $\mathcal{A}$ is to compute $\mathrm{XOR}_k \circ g$; however, we will show that $\mathcal{A}$ must additionally be able to compute g reasonably well on many coordinates of x. For any $i \in [k]$ and any leaf ℓ, define

$$\begin{aligned}
b_{i,\ell} &:= \operatorname*{argmax}_{b\in\{0,1\}} \Pr_{x\sim\mu^k}[b = g(x^{(i)}) | \operatorname{leaf}(\mathcal{A}, x) = \ell] \\
r_{i,\ell} &:= \Pr_{x\sim\mu^k}[b_{i,\ell} = g(x^{(i)}) | \operatorname{leaf}(\mathcal{A}, x) = \ell] \\
a_{i,\ell} &:= 2r_{i,\ell} - 1 \,.
\end{aligned}$$

If $\mathcal{A}$ reaches leaf ℓ on input y, then write $\mathcal{A}(y)_i := b_{i,\ell}$. $\mathcal{A}(y)_i$ represents $\mathcal{A}$'s best guess for $g(y^{(i)})$.

Next, we define some structural characteristics of leaves that we will need to complete the proof.

Definition 2.2 (Good leaves, good coordinates)**.**

- Call a leaf ℓ *good* if $r_\ell \geq 1 - 50\varepsilon$. Otherwise, call ℓ *bad*.
- Call a leaf ℓ *good for* i if $a_{i,\ell} \geq 1 - 5000\varepsilon/k$. Otherwise, call a leaf ℓ *bad for* i.

When a leaf is good for i, then $\mathcal{A}$, conditioned on reaching this leaf, computes $g(x^{(i)})$ with very high probability. Before presenting the main reduction, we give a few simple claims to help our proof. Our first claim shows that we reach a good leaf with high probability.

Claim 2.3. $\Pr_{x\sim\mu^k}[\operatorname{leaf}(\mathcal{A}, x)$ *is bad* $|\mathcal{A}(x)$ *doesn't abort*$] \leq \frac{1}{25}$.

Proof. Conditioned on $\mathcal{A}$ not aborting, it outputs the correct value of $\mathrm{XOR}_k \circ g$ with probability at least $1 - \frac{\varepsilon}{1-\delta} \geq 1 - 2\varepsilon$. We analyze this error probability by conditioning on which leaf is reached. Let ν be the distribution on $\operatorname{leaf}(\mathcal{A}, x)$ when $x \sim \mu^k$, conditioned on $\mathcal{A}$ not aborting. Let $L \sim \nu$. Then, we have:

$$\begin{aligned}
1 - 2\varepsilon &\leq \Pr_{x\sim\mu^k}[\mathcal{A}(x) = \mathrm{XOR}_k \circ g(x) | \mathcal{A} \text{ doesn't abort}] \\
&= \sum_{\text{leaf } \ell} \Pr_{L\sim\nu}[L = \ell] \cdot \Pr[\mathcal{A}(x) = \mathrm{XOR}_k \circ g(x) | L = \ell] \\
&= \sum_{\ell} \Pr[L = \ell] \cdot r_\ell \\
&= \mathop{\mathrm{E}}_{L}[r_L] \,.
\end{aligned}$$

Thus, $\mathrm{E}[r_L] \geq 1 - 2\varepsilon$. Recalling that ℓ is good if $r_\ell \geq 1 - 50\varepsilon$ and using Fact 1.10, L is bad with probability at most $\frac{1}{25}$. ◻

Next, we claim that each good leaf is good for many i.

Claim 2.4. *Let ℓ be any good leaf, and let I be uniform on $[k]$. Then, we have:*

$$\Pr_I[\ell \textit{ is bad for } I] \leq \frac{1}{25} \,.$$

Proof. Fix a good leaf ℓ, and let $\beta_\ell := \Pr_I[\ell \text{ is bad for } I]$. Recall that if ℓ is good, then $r_\ell \geq 1 - 50\varepsilon$. Therefore, $a_\ell \geq 1 - 100\varepsilon$. Using $1 + x \leq e^x$ and $e^{-2x} \leq 1 - x$ (which holds for all $0 \leq x \leq 1/2$), we have for any good leaf ℓ

$$1 - 100\varepsilon \leq a_\ell = \prod_{i=1}^{k} a_{i,\ell} \leq \left(1 - \frac{5000\varepsilon}{k}\right)^{k\beta_\ell} \leq e^{-5000\varepsilon\cdot\beta_\ell} \leq 1 - 2500\varepsilon\beta_\ell \ .$$

Rearranging terms, we see that $\beta_\ell \leq \frac{1}{25}$. □

Next, we describe a randomized algorithm $\mathcal{A}'$ for g whose expected query cost, abort probability, and error probability match the guarantees we want to provide when the input $x \sim \mu$. We will complete the proof of Lemma 2.1 by fixing the randomness used in $\mathcal{A}'$. Our algorithm works by independently $z \sim \mu^k$ and i uniformly from $[k]$, embedding x in the i-th coordinate of z, and emulating $\mathcal{A}$ on the resulting string.

Algorithm 1 $\mathcal{A}'(x)$

1: Independently sample I uniformly from $[k]$ and $z \sim \mu^k$.
2: $y \leftarrow z^{(I \leftarrow x)}$
3: Emulate algorithm $\mathcal{A}$ on input y.
4: Abort
  (i) if $\mathcal{A}$ aborts,
  (ii) if $\mathcal{A}$ reaches a bad leaf, or
  (iii) if $\mathcal{A}$ reaches a leaf that is bad for I.
  (iv) if $\mathcal{A}$ queries more than $\frac{25q}{k}$ bits of x,
5: Otherwise, output $\mathcal{A}(y)$.

Note that the emulation is possible since whenever $\mathcal{A}$ queries the j-th bit of $y^{(I)}$, we can query x_j, and we can emulate $\mathcal{A}$ querying a bit of $y^{(i)}$ for $i \neq I$ directly since z is fixed. We claim that (i) $\mathcal{A}'$ makes at most $\frac{25q}{k}$ queries, (ii) $\mathcal{A}'$ aborts with probability at most $\delta + 0.12$, and (iii) $\mathcal{A}'$ errs with probability at most $\frac{5000\varepsilon}{k}$.

First, note that $\mathcal{A}'$ makes at most $\frac{25q}{k}$ queries, since it aborts instead of making more queries.

Second, consider the abort probability of $\mathcal{A}'$. Our algorithm aborts if $\mathcal{A}$ aborts, if we reach a bad leaf, if the leaf we reach is bad for I, of if $\mathcal{A}$ makes more than $\frac{25q}{k}$ bits of $y^{(I)}$. Let $\mathcal{E}_1$ be the event that $\mathcal{A}$ aborts on input y. Similarly, let $\mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4$ be the events that $\mathcal{A}$ reaches a bad leaf, $\mathcal{A}$ reaches a leaf that is bad for i, and $\mathcal{A}$ queries more than $\frac{25q}{k}$ bits of x respectively. Since $x \sim \mu$, $z \sim \mu^k$, and I is uniform on $[k]$, it follows that $y \sim \mu^k$. By the abort guarantees of $\mathcal{A}$, we have $\Pr[\mathcal{E}_1] \leq \delta$. By Claim 2.3 we have $\Pr[\mathcal{E}_2|\mathcal{E}_1] \leq 1/25$, and by Claim 2.4 we have $\Pr[\mathcal{E}_3|\mathcal{E}_1, \mathcal{E}_2] \leq 1/25$. Thus, we have $\Pr[\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3] \leq \delta + \frac{2}{25}$.

Next, for each $i \in [k]$, let $q_i(y)$ denote the number of queries that $\mathcal{A}$ makes to $y^{(i)}$ on input y. The query cost of $\mathcal{A}$ guarantees that for each input y, $\sum_{1\leq i\leq k} q_i(y) \leq q$. Therefore, for any y, at most $\frac{k}{25}$ indices $i \in [k]$ satisfy $q_i(y) \geq \frac{25q}{k}$. Hence, for $I \in_R [k]$, $x \sim \mu$, and $z \sim \mu^k$, and recalling that $y = z^{(I \leftarrow x)}$, we have: $\Pr[\mathcal{E}_4] \leq \frac{1}{25}$. By a union bound, we have $\Pr_{I,z,x}[\mathcal{A}' \text{ aborts on input } y] = \Pr[\mathcal{E}_1 \vee \mathcal{E}_2 \vee \mathcal{E}_3 \vee \mathcal{E}_4] \leq \delta + \frac{3}{25} = \delta + 0.12$.

Third, we analyze the error probability of $\mathcal{A}'$. This algorithm errs only when it reaches a leaf that is good for I. By Claim 2.4, we are correct with probability at least $r_{I,\ell} = \frac{1+a_{I,\ell}}{2} \geq 1 - \frac{5000\varepsilon}{k}$. Thus, we have $\Pr[\mathcal{A}' \text{ errs}] \leq \frac{5000\varepsilon}{k}$.

Letting X be the indicator variable for the event that $\mathcal{A}'$ aborts and $Y = (I, z)$, Fact 1.9 gives

$$\Pr[\mathcal{A}' \text{ aborts }] = \mathrm{E}[\mathcal{A}' \text{ aborts }] = \mathop{\mathrm{E}}_{I,z}[\mathrm{E}[\mathcal{A}' \text{ aborts } | I, z]] = \mathop{\mathrm{E}}_{I,z}[\Pr[\mathcal{A}' \text{ aborts }]] .$$

Thus algortihm $\mathcal{A}'$ is a randomized algorithm that, when given an input $x \sim \mu$, makes at most $\frac{25q}{k}$ queries and has the following guarantees:

$$\mathop{\mathrm{E}}_{I,z}[\Pr_x[\mathcal{A}' \text{ aborts}]] = \Pr_{I,x,z}[\mathcal{A}' \text{ aborts}] \leq \delta + 0.12, \text{ and}$$

$$\mathop{\mathrm{E}}_{I,z}[\Pr_x[\mathcal{A}'(y)_{(I)} \neq g(x)]] = \Pr_{I,x,z}[\mathcal{A}'(y)_{(I)} \neq g(x)] \leq \frac{5000\varepsilon}{k} .$$

By Markov's inequality and a union bound, there must be a setting of (i^*, z^*) such that $\Pr_x[\mathcal{A}' \text{ aborts }] \leq 3\delta + 0.36$ and $\Pr_x[\mathcal{A}'(y)_{(i^*)} \neq g(x)] \leq \frac{15000\varepsilon}{k}$. Let $\mathcal{A}''$ be a *deterministic* algorithm that takes an input $x \sim \mu$ and emulates algorithm $\mathcal{A}'$ with i^* and z^* in place of the randomly sampled I, z. This algorithm queries at most $\frac{25q}{k}$, aborts with probability at most $3\delta + 0.36$, and errs with probability at most $\frac{15000\varepsilon}{k}$. Thus, it is a $[O(q/k), 3\delta + 0.36, \frac{15000\varepsilon}{k}, \mu]$-distributional algorithm for g, as required. □

2.1 Proof of Theorem 1.1

Proof of Theorem 1.1. Define $\varepsilon' := 30000\varepsilon$. Let μ be the input distribution for g achieving $\max_\mu \mathrm{D}^{\mu}_{\frac{1}{2},\frac{\varepsilon'}{k}}(g)$, and let μ^k be the k-fold product distribution of μ. By the first inequality of Fact 1.7 and the first inequality of Fact 1.8, we have

$$\overline{\mathrm{R}}_{\varepsilon}(\mathrm{XOR}_k \circ g) \geq \frac{1}{50}\,\mathrm{R}_{\frac{1}{50},\varepsilon}(\mathrm{XOR}_k \circ g) \geq \frac{1}{50}\,\mathrm{D}^{\mu^k}_{\frac{1}{25},2\varepsilon}(\mathrm{XOR}_k \circ g) .$$

Additionally, by Lemma 2.1 and the second inequalities of Fact 1.7 and Fact 1.8, we have

$$\mathrm{D}^{\mu^k}_{\frac{1}{25},2\varepsilon}(\mathrm{XOR}_k \circ g) \geq \frac{k}{120}\,\mathrm{D}^{\mu}_{\frac{1}{2},\frac{\varepsilon'}{k}}(g) \geq \frac{k}{120}\,\mathrm{R}_{\frac{2}{3},\frac{4\varepsilon'}{k}}(g) \geq \frac{k}{360}\,\overline{\mathrm{R}}_{\frac{12\varepsilon'}{k}}(g) .$$

Thus, we have $\overline{\mathrm{R}}_{\varepsilon}(\mathrm{XOR}_k \circ g) = \Omega\left(\mathrm{D}^{\mu^k}_{\frac{1}{25},2\varepsilon}(\mathrm{XOR}_k \circ g)\right)$ and $\mathrm{D}^{\mu^k}_{\frac{1}{25},2\varepsilon}(\mathrm{XOR}_k \circ g) = \Omega\left(k\,\overline{\mathrm{R}}_{\frac{12\varepsilon'}{k}}(g)\right)$. By standard success amplification $\overline{\mathrm{R}}_{\frac{12\varepsilon'}{k}}(g) = \Theta(\overline{\mathrm{R}}_{\frac{\varepsilon}{k}}(g))$. Putting these together yields

$$\overline{\mathrm{R}}_{\varepsilon}(\mathrm{XOR}_k \circ g) = \Omega\left(\mathrm{D}^{\mu^k}_{\frac{1}{25},2\varepsilon}(\mathrm{XOR}_k \circ g)\right) = \Omega\left(k\,\overline{\mathrm{R}}_{\frac{12\varepsilon'}{k}}(g)\right) = \Omega\left(\overline{\mathrm{R}}_{\frac{\varepsilon}{k}}(g)\right) ,$$

hence $\overline{\mathrm{R}}_\varepsilon(\mathrm{XOR}_k \circ g) = \Omega\left(k\,\overline{\mathrm{R}}_{\frac{\varepsilon}{k}}(g)\right)$ completing the proof. □

References

[1] ANDREW BASSILAKIS, ANDREW DRUCKER, MIKA GÖÖS, LUNJIA HU, WEIYUN MA, AND LI-YANG TAN: The power of many samples in query complexity. In *Proc. 47th Internat. Colloq. on Automata, Languages, and Programming (ICALP'20)*, pp. 9:1–18. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPIcs.ICALP.2020.9, arXiv:2002.10654, ECCC:TR20-027] 4

[2] YOSI BEN-ASHER AND ILAN NEWMAN: Decision trees with boolean threshold queries. *J. Comput. System Sci.*, 51(3):495–502, 1995. Preliminary version in CCC'95. [doi:10.1006/jcss.1995.1085] 2

[3] SHALEV BEN-DAVID AND ERIC BLAIS: A new minimax theorem for randomized algorithms. In *Proc. 61st FOCS*, pp. 403–411. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00045] 2, 4

[4] SHALEV BEN-DAVID AND ERIC BLAIS: A tight composition theorem for the randomized query complexity of partial functions. In *Proc. 61st FOCS*, pp. 240–246. IEEE Comp. Soc., 2020. [doi:10.1109/FOCS46700.2020.00031, arXiv:2002.10809] 2, 4

[5] SHALEV BEN-DAVID, MIKA GÖÖS, ROBIN KOTHARI, AND THOMAS WATSON: When is amplification necessary for composition in randomized query complexity? In *Proc. 24th Internat. Conf. on Randomization and Computation (RANDOM'20)*, pp. 28:1–16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2020. [doi:10.4230/LIPICS.APPROX/RANDOM.2020.28, arXiv:2006.10957] 2, 3, 4, 6

[6] SHALEV BEN-DAVID AND ROBIN KOTHARI: Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(5):1–27, 2018. Preliminary version in ICALP'16. [doi:10.4086/toc.2018.v014a005, arXiv:1605.09071, ECCC:TR16-087] 2

[7] ERIC BLAIS AND JOSHUA BRODY: Optimal separation and strong direct sum for randomized query complexity. In *Proc. 34th Comput. Complexity Conf. (CCC'19)*, pp. 29:1–17. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. [doi:10.4230/LIPIcs.CCC.2019.29, arXiv:1908.01020] 2, 3, 4, 6

[8] ANDREW DRUCKER: Improved direct product theorems for randomized query complexity. *Comput. Complexity*, 21(2):197–244, 2012. Preliminary version in CCC'11. [doi:10.1007/s00037-012-0043-7, arXiv:1005.0644, ECCC:TR10-080] 2, 3, 5

[9] RUSSELL IMPAGLIAZZO, RAN RAZ, AND AVI WIGDERSON: A direct product theorem. In *Proc. 9th IEEE Conf. Structure in Complexity Theory (SCT'94)*, pp. 88–96. IEEE Comp. Soc., 1994. [doi:10.1109/SCT.1994.315814] 2

[10] RAHUL JAIN, HARTMUT KLAUCK, AND MIKLOS SANTHA: Optimal direct sum results for deterministic and randomized decision tree complexity. *Inform. Process. Lett.*, 110(20):893–897, 2010. [doi:10.1016/j.ipl.2010.07.020] 2, 3

[11] MARCO MOLINARO, DAVID P. WOODRUFF, AND GRIGORY YAROSLAVTSEV: Beating the direct sum theorem in communication complexity with implications for sketching. In *Proc. 24th Ann. ACM–SIAM Symp. on Discrete Algorithms (SODA'13)*, pp. 1738–1756. SIAM, 2013. [doi:10.1137/1.9781611973105.125] 4

[12] MARCO MOLINARO, DAVID P. WOODRUFF, AND GRIGORY YAROSLAVTSEV: Amplification of one-way information complexity via codes and noise sensitivity. In *Proc. 42nd Internat. Colloq. on Automata, Languages, and Programming (ICALP'15)*, pp. 960–972. Springer, 2015. [doi:10.1007/978-3-662-47672-7_78, ECCC:TR15-031] 4

[13] NOAM NISAN, STEVEN RUDICH, AND MICHAEL E. SAKS: Products and help bits in decision trees. *SIAM J. Comput.*, 28(3):1035–1050, 1998. Preliminary version in FOCS'94. [doi:10.1137/S0097539795282444] 2

[14] RONEN SHALTIEL: Towards proving strong direct product theorems. *Comput. Complexity*, 12(1):1–22, 2003. Preliminary version in CCC'01. [doi:10.1007/s00037-003-0175-x, ECCC:TR01-009] 2, 3, 4

AUTHORS

Joshua Brody
Associate Professor
Department of Computer Science
Swarthmore College
Swarthmore, PA, USA
brody@cs.swarthmore.edu
http://cs.swarthmore.edu/~brody

Jae Tak Kim
Google
Mountain View, CA, USA
jkim17@swarthmore.edu
https://linkedin.com/in/jae-tak-kim

Peem Lerdputtipongporn
Ph. D. student
Statistics Department
Carnegie Mellon University
Pittsburgh, PA, USA
plerdput@andrew.cmu.edu
https://www.cmu.edu/dietrich/statistics-datascience/people/phd/peem-lerdputtipongporn.html

Hariharan Srinivasulu
Palantir Technologies
New York, NY, USA
srinivasulu.hari@gmail.com
https://www.linkedin.com/in/hari-srinivasulu/

ABOUT THE AUTHORS

JOSHUA BRODY, graduated from Dartmouth College in 2010; his advisor was Amit Chakrabarti. The subject of his thesis was communication complexity. Since graduating, his research interests have broadened to streaming algorithms, property testing, and data structure lower bounds. He was introduced to computer science by his father, who taught him to count on his fingers in binary when he was four years old. He spends most of his spare time with his children, who are good at math but don't seem to share his interest in counting in binary.

JAE TAK KIM graduated Swarthmore College in 2022, with a B. A. in computer science. During his undergraduate studies, he worked on research areas in query complexity theory and static programm analysis. In his free time, he enjoys climbing, reading, and listening to music. Jae Tak Kim currently works at Google, where he is a software engineer.

PEEM LERDPUTTIPONGPORN is a Ph. D. candidate in Statistics and Public Policy at Carnegie Mellon University. His advisors are David Choi and Nynke Niezink. His parents named him "Peem," a short Thai word that means "commanding respect," to offset the fact that he was born underweight. He received a B. A. in Mathematics and Computer Science from Swarthmore College in 2021. At Swarthmore, he he worked with Professor Joshua Brody on Query Complexity and Professor Steve Wang on Statistical Paleontology. His current interests are algorithmic fairness and statistical machine learning.

HARIHARAN SRINIVASULU is a 2022 graduate of Swarthmore College with a B. A. in Physics and Computer Science. He was named after a Hindu deity, although he suspects his name was inspired by one of his dad's favorite musicians. He was born and brought up in Chennai, India, and moved to the US for his undergraduate studies. He was mentored by Mike Brown and Joshua Brody at Swarthmore, and has done research in the areas of Computational Plasma Physics and Query Complexity. He hopes to enter a career in research in the future and is interested in areas such as quantum computing and distributed systems. Hariharan currently works for Palantir Technologies as a software engineer.