

A Strong XOR Lemma for Randomized Query Complexity

Joshua Brody¹, Jae Tak Kim¹, Peem Lerdputtipongporn¹, and Hariharan Srinivasulu¹

¹Swarthmore College

Abstract

We give a strong direct sum theorem for computing XOR $\circ g$. Specifically, we show that for every function g and every $k \geq 2$, the randomized query complexity of computing the XOR of k instances of g satisfies $\overline{R}_\varepsilon(\text{XOR} \circ g) = \Theta(k \overline{R}_{\frac{\varepsilon}{k}}(g))$. This matches the naive success amplification upper bound and answers a conjecture of Blais and Brody [7].

As a consequence of our strong direct sum theorem, we give a total function g for which $R(\text{XOR} \circ g) = \Theta(k \log(k) \cdot R(g))$, answering an open question from Ben-David et al. [5].

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

What happens if we don't 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.

Query Complexity

A *query algorithm* also known as *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 . 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 $f : \{0, 1\}^n \rightarrow \{0, 1\}$ 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 (worst-case) randomized query complexity of f (also known as the randomized decision tree complexity of f) is the minimum query complexity of an algorithm $\mathcal{A}$ that computes f with error at most ϵ . We denote this complexity by $R_\epsilon(f)$, and we write $R(f) := R_{\frac{1}{3}}(f)$ 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 *average query cost* of $\mathcal{A}$. The minimum average query complexity of an algorithm $\mathcal{A}$ that computes a function f with error at most ϵ is the *average ϵ -error query complexity* of f , which we denote by $\bar{R}_\epsilon(f)$. We again write $\bar{R}(f) := \bar{R}_{\frac{1}{3}}(f)$. Note that $\bar{R}_0(f)$ corresponds to the standard notion of *zero-error randomized query complexity* of f .

1.1 Our Results

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

Theorem 1. *For every function $g : \{0, 1\}^n \rightarrow \{0, 1\}$ and all $\epsilon > 0$, we have $\bar{R}_\epsilon(\text{XOR} \circ g) = \Omega(k \cdot \bar{R}_{\epsilon/k}(g))$.*

This answers Conjecture 1 of Blais and Brody [7] in the affirmative.

We prove Theorem 1 by proving an analogous result in distributional query complexity. We also allow our algorithms to *abort* with constant probability. Let $D_{\delta, \epsilon}^\mu(f)$ 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 $R_{\delta, \epsilon}(f)$ denote the minimal query cost of a randomized algorithm that computes f with abort probability at most δ and error probability at most ϵ (here probabilities are taken over the internal randomness of the algorithm).

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

Lemma 1 (Main Technical Lemma, informally stated.). *For every function $g : \{0, 1\}^n \rightarrow \{0, 1\}$, every distribution μ , and every small enough $\delta, \epsilon > 0$, we have*

$$D_{\delta, \epsilon}^{\mu^k}(\text{XOR} \circ g) = \Omega(k D_{\delta', \epsilon'}^\mu(g)) ,$$

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

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

Corollary 1. *There exists a total function $g : \{0, 1\}^n \rightarrow \{0, 1\}$ such that $R_\epsilon(\text{XOR} \circ g) = \Omega(k \log(k) \cdot R_\epsilon(g))$.*

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

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

where the second inequality is by Theorem 1 and the third inequality is from the query complexity guarantee of g . $\square$

This answers Open Question 1 from recent work of Ben-David et al. [5].

1.2 Previous and Related Work

Jain et al. [10] gave direct sum theorems for deterministic and randomized query complexity. While their direct sum result holds for worst-case randomized query complexity, they incur an *increase* in error ($R_\epsilon(f^k) \geq \delta \cdot k \cdot R_{\epsilon+\delta}(f)$) when computing a single copy of f . Shaltiel [14] gave a counterexample function for which direct sum fails to hold for distributional complexity. Drucker [8] gave a strong *direct product* theorem for randomized query complexity.

Our work is most closely related to that of Blais and Brody [7], who give a strong direct sum theorem for $\bar{R}_\epsilon(f^k) = \Omega(k \bar{R}_{\epsilon/k}(f))$, and explicitly conjecture that XOR admits a strong direct product theorem. Both [7] and ours use techniques similar to work of Molinaro et al. [11, 12] who give strong direct sum theorems for communication complexity.

Our strong direct sum for XOR is an example of a *composition theorem*—lower bound on the query complexity of functions of the form $f \circ g$. Several very recent works studied composition theorems in query complexity. Bassilakis et al. [1] show that $R(f \circ g) = \Omega(\text{fbs}(f)R(g))$, where $\text{fbs}(f)$ is the *fractional block sensitivity* of f . Ben-David and Blais [3, 4] give a tight lower bound on $R(f \circ g)$ as a product of $R(g)$ and a new measure they define called $\text{noisyR}(f)$, which measures the complexity of computing f on noisy inputs. They also characterize $\text{noisyR}(f)$ 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 $\text{XOR} \circ g$ 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 $R(\text{XOR} \circ g) = \Omega(k \log(k)R(g))$.

1.3 Our Technique.

Our technique most closely follows the strong direct sum theorem of Blais and Brody. We start with a query algorithm that computes $\text{XOR} \circ g$ and use it to build a query algorithm for computing g with low error. To do this, we'll take an input for g and *embed* it into an input for $\text{XOR} \circ g$. Given $x \in \{0, 1\}^n$, $i \in [k]$, and $y \in \{0, 1\}^{n \times k}$, let $y^{(i \leftarrow x)} := (y^{(1)}, \dots, y^{(i-1)}, x, y^{(i+1)}, \dots, y^{(k)})$ denote 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 of Drucker [8].

Lemma 2 ([8], Lemma 3.2). *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 $\text{XOR} \circ g(y)$ 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 $\text{adv}(X) := 2r(X) - 1$. Note that when $\text{adv}(X) = \delta$, then $r(X) = \frac{1}{2}(1 + \delta)$.

Fact 1. *Let $X_1, \dots, X_k$ bit independent random bits, and let a_i be the advantage of X_i . Then,*

$$\text{adv}(X_1 \oplus \dots \oplus X_k) = \prod_{i=1}^k \text{adv}(X_i) .$$

For completeness, we provide a proof of Fact 1 in Appendix A.

Given an algorithm for $\text{XOR} \circ g$ that has error ε , it follows that for typical leaves the advantage of computing $\text{XOR} \circ g$ is $\gtrsim 1 - 2\varepsilon$. Fact 1 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 reaching this leaf of the query algorithm, we could compute $g(y^{(i)})$ with very high probability. We'd 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 $\text{XOR} \circ g$ doesn't 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 $\text{XOR} \circ g$ 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 $\text{XOR} \circ g$ aborts or when it queries too many bits of x . We defer full details of the proof to Section 2.

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

²We use μ^k to denote the distribution on k -tuples where each coordinate is independently distributed $\sim \mu$.

1.4 Preliminaries and Notation

Suppose that f is a Boolean function on domain $\{0, 1\}^n$ and that μ is a distribution on $\{0, 1\}^n$. 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 . We write $\mathcal{A}(x) = \perp$ to denote that $\mathcal{A}$ aborts on input x .

Our main theorem is a direct sum result for $\text{XOR} \circ g$ for average case randomized query complexity; however, Lemma 1 uses distributional query complexity. The following results from Blais and Brody [7] connect the query complexities in the randomized, average-case randomized, and distributional query models.

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

$$\delta \cdot R_{\delta, \epsilon}(f) \leq \bar{R}_\epsilon(f) \leq \frac{1}{1-\delta} \cdot R_{\delta, (1-\delta)\epsilon}(f).$$

Fact 3 ([7], Lemma 15). *For any $\alpha, \beta > 0$ such that $\alpha + \beta \leq 1$, we have*

$$\max_{\mu} D_{\delta/\alpha, \varepsilon/\beta}^{\mu}(f) \leq R_{\delta, \varepsilon}(f) \leq \max_{\mu} D_{\alpha\delta, \beta\varepsilon}^{\mu}(f).$$

We'll also use the following convenient facts about probability and expectation. For completeness we provide proofs in Appendix A.

Fact 4. *Let S, T be random variables. Let $\mathcal{E} = \mathcal{E}(S, T)$ and $\mathcal{A}$ be events, and for any s , let μ_s be the distribution on T conditioned on $S = s$. Then,*

$$\Pr_{S, T}[\mathcal{E} | \mathcal{A}] = \mathbb{E}_S \left[\Pr_{T \sim \mu_S}[\mathcal{E}(S, T) | \mathcal{A}] \right].$$

Fact 5 (Markov Inequality for Bounded Variables). *Let X be a real-valued random variable with $0 \leq X \leq 1$. Suppose that $E[X] \geq 1 - \varepsilon$. Then, for any $T > 1$ it holds that*

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

2 Strong XOR Lemma

In this section, we prove our main result.

Lemma 3 (Formal Restatement of Lemma 1). *For every function $g : \{0, 1\}^n \rightarrow \{0, 1\}$, every distribution μ on $\{0, 1\}^n$, every $0 \leq \delta \leq \frac{1}{5}$, and every $0 < \varepsilon \leq \frac{1}{800}$, we have*

$$D_{\delta, \varepsilon}^{\mu^k}(\text{XOR} \circ g) = \Omega \left(k \cdot D_{\delta', \varepsilon'}^{\mu}(g) \right),$$

for $\delta' = 0.34 + 4\delta$ and $\varepsilon' = \frac{320000\varepsilon}{k}$.

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

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

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

The purpose of $\mathcal{A}$ is to compute XOR og; however, we'll 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

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

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'll need to complete the proof.

Definition 1 (Good leaves, good coordinates).

- Call a leaf ℓ good if $r_\ell \geq 1 - 200\varepsilon$.
- Call a leaf ℓ good for i if $a_{i,\ell} \geq 1 - 80000\varepsilon/k$.
- Call coordinate i good if $\Pr_{x \sim \mu^k} [\text{leaf}(\mathcal{A}, x) \text{ is good for } i | \mathcal{A}(x) \text{ doesn't abort}] \geq 1 - \frac{3}{50}$.

When a leaf is good for i , then $\mathcal{A}$, conditioned on reaching this leaf, computes $g(x^{(i)})$ with very high probability. When a coordinate i is good, then with high probability $\mathcal{A}$ reaches a leaf that is good for i . To make our embedding work, we need to fix a good coordinate i^* such that $\mathcal{A}$ makes only $O(q/k)$ queries on this coordinate. The following claim shows that most coordinates are good.

Claim 1. i is good for at least $\frac{2}{3}k$ indices $i \in [k]$.

We defer the proof of Claim 1 to the following subsection. Next, for each $i \in [k]$, let $q_i(x)$ denote the number of queries that $\mathcal{A}$ makes to $x^{(i)}$ on input x . The query cost of $\mathcal{A}$ guarantees that for each input x , $\sum_{1 \leq i \leq k} q_i(x) \leq q$. Therefore, $\sum_{i \in [k]} \mathbb{E}_{x \sim \mu^k} [q_i(x)] \leq q$, and so at least $\frac{2}{3}k$ indices $i \in [k]$ satisfy

$$\mathbb{E}_{x \sim \mu^k} [q_i(x)] \leq \frac{3q}{k} . \quad (1)$$

Thus, there exists i^* which satisfies both Claim 1 and inequality (1). Fix such an i^* . For inputs $y \in \{0,1\}^{n \times k}$ and $x \in \{0,1\}^n$, let $y^{(i^* \leftarrow x)} := (y^{(1)}, \dots, y^{(i^*-1)}, x, y^{(i^*+1)}, \dots, y^{(k)})$ denote the input obtained from y by replacing $y^{(i^*)}$ with x . Note that if $y \sim \mu^k$ and $x \sim \mu$, then $y^{(i \leftarrow x)} \sim \mu^k$ for all $i \in [k]$. With this notation and using Fact 4, the conditions from inequality (1) and Claim 1 satisfied by i^* can be rewritten as

$$\mathbb{E}_{y \sim \mu^k} \left[\mathbb{E}_{x \sim \mu} [q_{i^*}(y^{(i^* \leftarrow x)})] \right] \leq \frac{3q}{k} ,$$

and

$$\mathbb{E}_{y \sim \mu^k} \left[\Pr_{x \sim \mu} [\text{leaf}(\mathcal{A}, y^{(i^* \leftarrow x)}) \text{ is bad for } i^* | \mathcal{A}(y^{(i^* \leftarrow x)}) \text{ doesn't abort}] \right] \leq \frac{3}{50} .$$

Since $\mathcal{A}$ has at most δ abort probability, we have

$$\mathbb{E}_{y \sim \mu^k} \left[\Pr_{x \sim \mu} [\mathcal{A}(y^{(i^* \leftarrow x)}) = \perp] \right] \leq \delta .$$

Finally, for any leaf ℓ for which i^* is good, we have $a_{i^*,\ell} \geq 1 - 80000\varepsilon/k$. Hence

$$\mathbb{E}_{y \sim \mu^k} \left[\Pr_{x \sim \mu} [\mathcal{A}(y^{(i^* \leftarrow x)})_{i^*} \neq g(x) | \text{leaf}(\mathcal{A}, y^{(i^* \leftarrow x)}) \text{ is good for } i^*] \right] \leq \frac{80000\varepsilon}{k} .$$

Therefore by Markov's Inequality, there exists $z \in \{0, 1\}^{n \times k}$ such that

$$\mathbb{E}_{x \sim \mu} \left[q_{i^*}(z^{(i^* \leftarrow x)}) \right] \leq \frac{12q}{k}, \quad (2)$$

$$\Pr_{x \sim \mu} \left[\text{leaf}(\mathcal{A}, z^{(i^* \leftarrow x)}) \text{ is bad for } i^* \mid \mathcal{A}(z^{(i^* \leftarrow x)}) \neq \perp \right] \leq \frac{6}{25}, \quad (3)$$

$$\Pr_{x \sim \mu} \left[\mathcal{A}(z^{(i^* \leftarrow x)}) = \perp \right] \leq 4\delta, \text{ and} \quad (4)$$

$$\Pr_{x \sim \mu} \left[\mathcal{A}(z^{(i^* \leftarrow x)})_{i^*} \neq g(x) \mid \text{leaf}(\mathcal{A}, z^{(i^* \leftarrow x)}) \text{ is good for } i^* \right] \leq \frac{320000\varepsilon}{k}. \quad (5)$$

Fix this z . Now that i^* and z are fixed, we are ready to describe our algorithm.

Algorithm 1 $\mathcal{A}'_{z, i^*}(x)$

- 1: $y \leftarrow z^{(i^* \leftarrow x)}$
 - 2: Emulate algorithm $\mathcal{A}$ on input y .
 - 3: Abort if $\mathcal{A}$ aborts, if $\mathcal{A}$ queries more than $\frac{120q}{k}$ bits of x , or if $\mathcal{A}$ reaches a bad leaf.
 - 4: 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. It remains to show that $\mathcal{A}'$ is a $\left[\frac{120q}{k}, 0.34 + 4\delta, \frac{320000\varepsilon}{k}, \mu \right]$ -distributional query algorithm for f .

First, note that $\mathcal{A}'$ makes at most $120q/k$ queries, since it aborts instead of making more queries. Next, consider the abort probability of $\mathcal{A}'$. Our algorithm aborts if $\mathcal{A}$ aborts, if $\mathcal{A}$ probes more than $\frac{120q}{k}$ bits, or if $\mathcal{A}$ reaches a bad leaf. By inequality (4), $\mathcal{A}$ aborts with probability at most 4δ . By inequality (2) and Markov's Inequality, the probability that $\mathcal{A}$ probes $120q/k$ bits is at most $1/10$. By inequality (3), we have $\Pr_{x \sim \mu}[\mathcal{A} \text{ reaches a bad leaf}] \leq 6/25$. Hence, $\mathcal{A}'$ aborts with probability at most $4\delta + \frac{1}{10} + \frac{6}{25} = 0.34 + 4\delta$. Finally, note that if $\mathcal{A}'$ doesn't abort, then $\mathcal{A}$ reaches a leaf which is good for i^* . By inequality (5), $\mathcal{A}'$ errs with probability at most $320000\varepsilon/k$ in this case.

We have constructed an algorithm $\mathcal{A}'$ for g that makes at most $120q/k$ queries, and when the input $x \sim \mu$, $\mathcal{A}'$ aborts with probability at most δ' and errs with probability at most ε' . Hence, $D_{\delta', \varepsilon'}^\mu(g) \leq 120q/k$. Rearranging terms and recalling that $q = D_{\delta, \varepsilon}^{\mu^k}(\text{XOR} \circ g)$, we get

$$D_{\delta, \varepsilon}^{\mu^k}(\text{XOR} \circ g) \geq \frac{k}{120} D_{\delta', \varepsilon'}^\mu(g),$$

completing the proof. $\square$

2.1 Proof of Claim 1.

Proof of Claim 1. Let I be uniform on $[k]$. We want to show that $\Pr[I \text{ is good}] \geq 2/3$.

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

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

Thus, $\mathbb{E}[r_L] \geq 1 - 2\varepsilon$. Recalling that ℓ is good if $r_\ell \geq 1 - 200\varepsilon$ and using Fact 5, L is good with probability at least 0.99. Note also that when ℓ is good, then $a_\ell \geq 1 - 400\varepsilon$. Let $\beta_\ell := \Pr_I[\ell \text{ is bad for } I]$. 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

$$1 - 400\varepsilon \leq a_\ell = \prod_{i=1}^k a_{i,\ell} \leq \left(1 - \frac{80000\varepsilon}{k}\right)^{k\beta_\ell} \leq e^{-80000\varepsilon \cdot \beta_\ell} \leq 1 - 40000\varepsilon\beta_\ell.$$

Rearranging terms, we see that $\beta_\ell \leq 0.01$. We've just shown that a random leaf ℓ is good with high probability, and when ℓ is good, it is good for many i . We need to show that there are many i such that most leaves are good for i . Towards that end, let $\delta_{i,\ell} := 1$ if ℓ is good for i ; otherwise, set $\delta_{i,\ell} := 0$.

$$\begin{aligned} \mathbb{E}_I \left[\Pr_{x \sim \mu^k} [\text{leaf}(\mathcal{A}, x) \text{ good for } I | \mathcal{A} \text{ doesn't abort}] \right] &= \mathbb{E}_I \left[\sum_{\ell} \Pr[L = \ell] \cdot \delta_{I,\ell} \right] \\ &= \sum_{\ell} \Pr[L = \ell] \mathbb{E}_I[\delta_{I,\ell}] \\ &= \sum_{\ell} \Pr[L = \ell] \Pr_I[\ell \text{ good for } I] \\ &\geq \sum_{\text{good } \ell} \Pr[L = \ell] \cdot (1 - \beta_\ell) \\ &= \Pr_L[L \text{ is good}] \cdot (1 - \beta_\ell) \\ &\geq 0.99(1 - \beta_\ell) \\ &> 0.98. \end{aligned}$$

Thus, $\mathbb{E}_I [\Pr_{x \sim \mu^k} [\text{leaf}(\mathcal{A}, x) \text{ good for } I | \mathcal{A} \text{ doesn't abort}]] \geq 1 - \frac{1}{50}$. Recalling that i is good if $\Pr[\text{leaf}(\mathcal{A}, x) \text{ good for } i | \mathcal{A}(x) \text{ doesn't abort}] \geq 1 - \frac{3}{50}$ and using Fact 5, it follows that $\Pr_I[I \text{ is good}] \geq 2/3$. This completes the proof. $\square$

2.2 Proof of Theorem 1

Proof of Theorem 1. Define $\varepsilon' := 640000\varepsilon$. Let μ be the input distribution for g achieving $\max_{\mu} D_{\frac{1}{2}, \frac{\varepsilon'}{k}}^{\mu}(g)$, and let μ^k be the k -fold product distribution of μ . By the first inequality of Fact 2 and the first inequality of Fact 3, we have

$$\bar{R}_{\varepsilon}(\text{XOR} \circ g) \geq \frac{1}{50} R_{\frac{1}{50}, \varepsilon}(\text{XOR} \circ g) \geq \frac{1}{50} D_{\frac{1}{25}, 2\varepsilon}^{\mu^k}(\text{XOR} \circ g).$$

Additionally, by Lemma 1 and the second inequalities of Facts 2 and 3, we have

$$D_{\frac{1}{25}, 2\varepsilon}^{\mu^k}(\text{XOR} \circ g) \geq \frac{k}{120} D_{\frac{1}{2}, \frac{\varepsilon'}{k}}^{\mu}(g) \geq \frac{k}{120} R_{\frac{2}{3}, \frac{4\varepsilon'}{k}}(g) \geq \frac{k}{360} \bar{R}_{\frac{12\varepsilon'}{k}}(g).$$

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

$$\bar{R}_{\varepsilon}(\text{XOR} \circ g) = \Omega\left(D_{\frac{1}{25}, 2\varepsilon}^{\mu^k}(\text{XOR} \circ g)\right) = \Omega\left(k \bar{R}_{\frac{12\varepsilon'}{k}}(g)\right) = \Omega\left(\bar{R}_{\frac{\varepsilon}{k}}(g)\right),$$

hence $\bar{R}_{\varepsilon}(\text{XOR} \circ g) = \Omega\left(k \bar{R}_{\frac{\varepsilon}{k}}(g)\right)$ completing the proof. $\square$

Acknowledgments

The authors thank Runze Wang for several helpful discussions.

References

- [1] Andrew Bassilakis, Andrew Drucker, Mika Gs, Lunjia Hu, Weiyun Ma, and Li-Yang Tan. The power of many samples in query complexity. In *Proceedings 47th Annual International Colloquium on Automata, Languages, and Programming*, 2020.
- [2] Yosi Ben-Asher and Ilan Newman. Decision trees with AND, OR queries. In *Proceedings 10th Annual Structure in Complexity Theory Conference*, pages 74–81, 1995.
- [3] Shalev Ben-David and Eric Blais. A new minimax theorem for randomized algorithms, 2020.
- [4] Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions, 2020.
- [5] Shalev Ben-David, Mika Göös, Robin Kothari, and Thomas Watson. When is amplification necessary for composition in randomized query complexity? *CoRR*, abs/2006.10957, 2020.
- [6] Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(1):1–27, 2018.
- [7] Eric Blais and Joshua Brody. Optimal separation and strong direct sum for randomized query complexity. (*Originally appeared in CCC 2019*) *CoRR*, abs/1908.01020, 2019.
- [8] Andrew Drucker. Improved direct product theorems for randomized query complexity. *Computational Complexity*, 21(2):197–244, 2012.
- [9] Russell Impagliazzo, Ran Raz, and Avi Wigderson. A direct product theorem. In *Proceedings 9th Annual Structure in Complexity Theory Conference*, pages 88–96, 1994.
- [10] Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Inf. Process. Lett.*, 110(20):893–897, 2010.
- [11] Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1738–1756. SIAM, 2013.
- [12] Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev. Amplification of one-way information complexity via codes and noise sensitivity. In *International Colloquium on Automata, Languages, and Programming*, pages 960–972. Springer, 2015.
- [13] Noam Nisan, Steven Rudich, and Michael E. Saks. Products and help bits in decision trees. *SIAM Journal on Computing*, 28(3):1035–1050, 1999.
- [14] Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1-2):1–22, 2003.

A Proofs of Technical Lemmas

Proof of Fact 1. For each i , let $b_i := \operatorname{argmax}_{b \in \{0,1\}} \Pr[X_i = b]$ and $\delta_i := \operatorname{adv}(X_i)$. Then $\Pr[X_i = b_i] = \frac{1}{2}(1 + \delta_i)$. We prove Fact 1 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 $X_1 \oplus \dots \oplus X_{k-1}$. Then, setting $Y := X_1 \oplus \dots \oplus X_{k-1}$, by the induction hypothesis, we have $\text{adv}(Y) = \prod_{i=1}^{k-1} \text{adv}(X_i)$. Moreover, $X_1 \oplus \dots \oplus X_k = Y \oplus X_k$, and

$$\text{adv}(X_1 \oplus \dots \oplus X_k) = \text{adv}(Y \oplus X_k) = \text{adv}(Y) \text{adv}(X_k) = \prod_{i=1}^k \text{adv}(X_i) .$$

□

Proof of Fact 4. We condition $\Pr_{S,T}[\mathcal{E}(S,T)|\mathcal{A}]$ on S .

$$\begin{aligned} \Pr_{S,T}[\mathcal{E}|\mathcal{A}] &= \sum_s \Pr[S = s|\mathcal{A}] \Pr_T[\mathcal{E}(S,T)|\mathcal{A}, S = s] \\ &= \sum_s \Pr[S = s|\mathcal{A}] \Pr_{T \sim \mu_s}[\mathcal{E}(S,T)|\mathcal{A}] \\ &= \mathbb{E}_S \left[\Pr_{T \sim \mu_S}[\mathcal{E}(S,T)|\mathcal{A}] \right] . \end{aligned}$$

□

Proof of Fact 5. Let $Y := 1 - X$. Then, $E[Y] \leq \varepsilon$. By Markov's Inequality we have

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

□