The Power of Many Samples in Query Complexity

Andrew Bassilakis
Stanford University, CA, USA
abass20@stanford.edu

Andrew Drucker
University of Chicago, IL, USA
andy.drucker@gmail.com

Mika Göös
Stanford University, CA, USA
goos@stanford.edu

Lunjia Hu
Stanford University, CA, USA
lunjia@stanford.edu

Weiyun Ma
Stanford University, CA, USA
wyma@cs.stanford.edu

Li-Yang Tan
Stanford University, CA, USA
liyang@cs.stanford.edu

Abstract

The randomized query complexity $\mathsf{R}(f)$ of a boolean function $f\colon \{0,1\}^n \to \{0,1\}$ is famously characterized (via Yao's minimax) by the least number of queries needed to distinguish a distribution $\mathcal{D}_0$ over 0-inputs from a distribution $\mathcal{D}_1$ over 1-inputs, maximized over all pairs $(\mathcal{D}_0, \mathcal{D}_1)$. We ask: Does this task become easier if we allow query access to infinitely many samples from either $\mathcal{D}_0$ or $\mathcal{D}_1$? We show the answer is *no*: There exists a hard pair $(\mathcal{D}_0, \mathcal{D}_1)$ such that distinguishing $\mathcal{D}_0^\infty$ from $\mathcal{D}_1^\infty$ requires $\Theta(\mathsf{R}(f))$ many queries. As an application, we show that for any composed function $f \circ g$ we have $\mathsf{R}(f \circ g) \geq \Omega(\mathsf{fbs}(f)\mathsf{R}(g))$ where fbs denotes fractional block sensitivity.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Probabilistic computation

Keywords and phrases Query complexity, Composition theorems

Digital Object Identifier 10.4230/LIPIcs.ICALP.2020.9

Category Track A: Algorithms, Complexity and Games

Funding *Lunjia Hu*: supported by NSF Award IIS-1908774 and a VMware fellowship.
Weiyun Ma: supported by a Stanford Graduate Fellowship.
Li-Yang Tan: supported by NSF grant CCF-1921795 and CAREER Award CCF-1942123.

Acknowledgements We thank Shalev Ben-David for correspondence about their ongoing work [5, 6]. AD thanks Mark Braverman for interesting discussions of related topics.

47th International Colloquium on Automata, Languages, and Programming (ICALP 2020).
Editors: Artur Czumaj, Anuj Dawar, and Emanuela Merelli; Article No. 9; pp. 9:1–9:18
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Randomized query complexity (see [8] for a classic survey) is often studied using Yao's minimax principle [20]. The principle states that for every boolean function $f\colon \{0,1\}^n \to \{0,1\}$

$$\textbf{Yao's minimax:}\qquad \mathsf{R}_\epsilon(f) \;=\; \max\nolimits_{\mathcal{D}} \mathsf{D}_\epsilon(f,\mathcal{D}).$$

- Here $\mathsf{R}_\epsilon(f)$ is the randomized ϵ-error query complexity of f. More precisely, $\mathsf{R}_\epsilon(f)$ equals the least number of queries a randomized algorithm (decision tree) must make to the input bits $x_i \in \{0,1\}$ of an unknown input $x \in \{0,1\}^n$ in order to output $f(x)$ with probability at least $1-\epsilon$ (where the probability is over the coin tosses of the algorithm). We often set $\epsilon = 1/3$ and omit ϵ from notation, as it is well known that this choice only affects constant factors in query complexity.
- $\mathcal{D}$ is a distribution over the inputs $\{0,1\}^n$. We may assume wlog that $\mathcal{D}$ is *balanced*: $\mathcal{D} = \frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$ where $\mathcal{D}_b$ is a distribution over $f^{-1}(b)$.
- $\mathsf{D}_\epsilon(f,\mathcal{D})$ is the *distributional* ϵ-error query complexity of f relative to $\mathcal{D}$. More precisely, $\mathsf{D}_\epsilon(f,\mathcal{D})$ equals the least number of queries a *deterministic* algorithm must make to an input $x \sim \mathcal{D}$ in order to output $f(x)$ with probability at least $1-\epsilon$ (where the probability is over $x \sim \mathcal{D}$).

1.1 Correlated samples problem

One way to think about the distributional complexity of f relative to $\mathcal{D} = \frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$ is as the following task: A deterministic algorithm is given query access to a sample from either $\mathcal{D}_0$ or $\mathcal{D}_1$ and it needs to decide which is the case. In this work, we ask: Does this task become easier if we allow query access to an unlimited number of independent samples from either $\mathcal{D}_0$ or $\mathcal{D}_1$? In short,

Is it easier to distinguish $\mathcal{D}_0^\infty$ from $\mathcal{D}_1^\infty$ than it is to distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$?

More formally, we define the *correlated samples problem* for f relative to $\mathcal{D} = \frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$ by

$$\mathsf{Corr}_\epsilon(f,\mathcal{D}) \;:=\; \min_{k\geq 1} \mathsf{D}_\epsilon(f^k, \tfrac{1}{2}\mathcal{D}_0^k + \tfrac{1}{2}\mathcal{D}_1^k).$$

Here $f^k\colon (\{0,1\}^n)^k \to \{0,1\}^k$ is the function that evaluates k copies of f on disjoint inputs. We also use the notation $\mathcal{D}^k := \mathcal{D} \times \cdots \times \mathcal{D}$ (k times) for the k-fold product distribution. In particular, under $\frac{1}{2}\mathcal{D}_0^k + \frac{1}{2}\mathcal{D}_1^k$, the function f^k outputs either 0^k or 1^k; the correlated samples problem is to decide which is the case. We note that the expression to be minimized on the right side is a non-increasing function of k (access to more samples is only going to help). We may also assume wlog that $k \leq n$ (when an algorithm queries a sample for the first time, we may assume it is the first unqueried sample so far).

Shaltiel examples. It is not hard to give examples of input distributions where access to multiple correlated samples *does* help. Such examples were already discussed by Shaltiel [18] in the context of direct product theorems. For instance, consider the n-bit $\textsc{Xor}_n$ function. It is well known that $\mathsf{R}_\epsilon(\textsc{Xor}_n) = n$ for all $\epsilon > 0$. Define a balanced input distribution (here $\mathcal{U}$ is a uniform random bit in $\{0,1\}$)

$$\mathcal{D} \;:=\; \begin{cases} 0\,\mathcal{U}^{n-1} & \text{with probability } 99\%, \\ 1\,\mathcal{U}\,0^{n-2} & \text{with probability } 1\%. \end{cases}$$

This distribution is hard 99% of the time: if the first bit is 0, an algorithm has to compute $\textsc{Xor}_{n-1}$ relative to $\mathcal{U}^{n-1}$, which requires $n-1$ queries. For the remaining 1%, the distribution is easy: if the first bit is 1, the output can be deduced from the second bit. Here multiple correlated samples help a lot (for $\epsilon = 1/3$):

$$\begin{aligned} \mathsf{D}(\textsc{Xor}_n, \mathcal{D}) &= \Omega(n), \\ \mathsf{Corr}(\textsc{Xor}_n, \mathcal{D}) &= O(1). \end{aligned}$$

Indeed, given a single sample from $\mathcal{D}$, an algorithm is likely to have to solve the hard case of the distribution. By contrast, given multiple correlated samples, we can query the first bit for a large constant number of samples. This will give us a high chance to encounter at least one easy sample.

Error reduction. An important fact (which fails in the single-sample setting!) is that we can amplify the success probability of any algorithm for correlated samples. This is achieved by a variant of the usual trick: repeatedly run the algorithm on fresh samples to gain more confidence about the output.[1]

▶ **Fact 1.1.** $\mathsf{Corr}_\epsilon(f, \mathcal{D}) \leq O(\log(1/\epsilon)/\delta^2) \cdot \mathsf{Corr}_{1/2-\delta}(f, \mathcal{D})$ *for every* $(f, \mathcal{D})$.

The aforementioned Shaltiel example $(\textsc{Xor}_n, \mathcal{D})$ can alternatively be computed as follows: By querying the first two bits of a single sample $x \sim \mathcal{D}$ one can predict $\textsc{Xor}_n(x)$ to within error 49.5%. Now apply Fact 1.1 to reduce the error below 1/3 at the cost of a constant-factor blowup in query cost.

1.2 Main result

We study whether Shaltiel examples can be avoided if we restrict our attention to the hardest possible input distribution. Namely, we define a distribution-free complexity measure by

$$\mathsf{Corr}_\epsilon(f) \;:=\; \max_{\mathcal{D}} \mathsf{Corr}_\epsilon(f, \mathcal{D}).$$

Our main result is that multiple correlated samples do not help for the hardest distribution.

▶ **Theorem 1.2.** $\mathsf{Corr}(f) = \Theta(\mathsf{R}(f))$ *for any (partial) boolean function* f.

The main challenge in proving Theorem 1.2 is precisely the existence of Shaltiel examples: How to construct hard distributions that do not contain any hidden easy parts? We resolve it by building decision trees that can exploit the easy parts not only in its own input distribution, but in various other distributions as well.

1.3 Application 1: Selection problem

Next we describe a consequence of our main result to a natural query task that we dub the *selection problem*. A similar problem, called *choose*, was studied by [4] in communication complexity.

[1] In more detail: An algorithm T with error $1/2 - \delta$ has $|p_0 - p_1| \geq 2\delta$ where $p_i := \Pr[T(x_i) = 1]$ for $x_i \sim \mathcal{D}_i^k$. Reducing error below $\epsilon > 0$ boils down to distinguishing two random coins with heads-probabilities p_0 and p_1. Given multiple samples from one of the coins, Chernoff bounds state that $O(\log(1/\epsilon)/\delta^2)$ samples are enough to tell which coin the samples came from.

Fix an n-bit function f together with an input distribution $\mathcal{D}$. In the *k-selection problem* for $(f, \mathcal{D})$ the input is a random kn-bit string $x = (x^1, \dots, x^k) \sim \mathcal{D}^k$, and the goal is to output $(i, f(x^i))$ for some $i \in [k]$. That is, the algorithm gets access to k independent samples from $\mathcal{D}$ and it selects one of them to solve. We define

$$\begin{aligned} k\text{-}\mathsf{Sel}_\epsilon(f,\mathcal{D}) &\coloneqq \epsilon\text{-error query complexity of } k\text{-selection for } (f,\mathcal{D}),\\ \mathsf{Sel}_\epsilon(f,\mathcal{D}) &\coloneqq \min\nolimits_{k\geq 1}\ k\text{-}\mathsf{Sel}_\epsilon(f,\mathcal{D}),\\ \mathsf{Sel}_\epsilon(f) &\coloneqq \max\nolimits_{\mathcal{D}}\ \mathsf{Sel}_\epsilon(f,\mathcal{D}). \end{aligned}$$

The selection problem is interesting because it, too, is subject to Shaltiel examples: for $(\textsc{Xor}_n, \mathcal{D})$ as described in Subsection 1.1, we have $\mathsf{Sel}(\textsc{Xor}_n, \mathcal{D}) = O(1)$ using the same idea of searching for an easy sample.

The following relates selection to correlated samples; see Section 5 for the proof.

▶ **Theorem 1.3.** *The correlated samples problem is easier than selection:*

1. $\mathsf{Corr}(f, \mathcal{D}) \leq O(\mathsf{Sel}(f, \mathcal{D}))$ *for every* $(f, \mathcal{D})$.
2. *There exists an n-bit $(f, \mathcal{D})$ such that* $\mathsf{Sel}(f, \mathcal{D}) = \Omega(n)$ *but* $\mathsf{Corr}(f, \mathcal{D}) = O(1)$.
3. *Selection does not admit efficient error reduction (as in Fact 1.1).*

Combining the first item of Theorem 1.3 with our main result (Theorem 1.2) we conclude that multiple samples do not help in the selection problem for the hardest distribution.

▶ **Corollary 1.4.** $\mathsf{Sel}(f) = \Theta(\mathsf{R}(f))$ *for any (partial) boolean function f.*

1.4 Application 2: Randomized composition

We give another application of our main result to the *randomized composition conjecture* studied in [7, 3, 9, 6]. In fact, this application is what originally motivated our research project!

For an n-bit function f and an m-bit function g we define their composition

$$f \circ g \colon (\{0,1\}^m)^n \to \{0,1\} \qquad \text{such that} \qquad (f \circ g)(x^1, \dots, x^n) \coloneqq f(g(x^1), \dots, g(x^n)).$$

A *composition theorem* aims to understand the query complexity of $f \circ g$ in terms of f and g. Such theorems are known for deterministic query complexity, $\mathsf{D}(f \circ g) = \mathsf{D}(f)\mathsf{D}(g)$ [17, 19, 14], and quantum query complexity, $\mathsf{Q}(f \circ g) = \Theta(\mathsf{Q}(f)\mathsf{Q}(g))$ [12, 16]. The conjecture in the randomized case is:

▶ **Conjecture 1.5.** $\mathsf{R}(f \circ g) \geq \Omega(\mathsf{R}(f)\mathsf{R}(g))$ *for all boolean functions f and g.*

Gavinsky et al. [9] have shown that the conjecture fails if f is allowed to be a *relation*. They also show $\mathsf{R}(f \circ g) \geq \Omega(\mathsf{R}(f)\mathsf{R}(g)^{1/2})$ for any relation f and partial function g. In a very recent work (concurrent to ours) Ben-David and Blais [5, 6] have found a counterexample to the randomized conjecture for partial f and g, albeit with a tiny query complexity compared to input length; see also Subsection 1.5. The conjecture is still open for total functions.

Fractional block sensitivity. We show a new composition theorem in terms of *fractional block sensitivity* $\mathsf{fbs}(f)$, introduced by [19, 10]; see also [13, 2]. This measure is at most randomized query complexity, $\mathsf{fbs}(f) \le O(\mathsf{R}(f))$, and it is equivalent to *randomized certificate complexity* [1].

Let us define $\mathsf{fbs}(f)$ for an n-bit f. We say that a block $B \subseteq [n]$ is *sensitive* on input x iff $f(x) \neq f(x^B)$ where x^B is x but with bits in B flipped. Fix an input x and introduce a real weight $w_B \in [0,1]$ for each sensitive block B of x. Define $\mathsf{fbs}(f,x)$ as the optimum value of the following linear program

$$\begin{array}{rll} \max & \sum_B w_B & \\ \textit{subject to} & \sum_{B \ni i} w_B \le 1, & \forall i \in [n], \\ & w_B \ge 0, & \forall B. \end{array}$$

Finally, define $\mathsf{fbs}(f) := \max_x \mathsf{fbs}(f,x)$. For comparison, the more usual *block sensitivity* $\mathsf{bs}(f)$ [15] is defined the same way except with the integral constraint $w_B \in \{0,1\}$. In particular $\mathsf{bs}(f) \le \mathsf{fbs}(f)$, and moreover a polynomial gap (power 1.5) between the two is known for a total function [10].

We make progress towards the composition conjecture; see Section 6 for the proof.

▶ **Theorem 1.6.** $\mathsf{R}(f \circ g) \ge \Omega(\mathsf{fbs}(f)\mathsf{R}(g))$ *for any (partial) boolean functions f and g.*

The previous best comparable composition theorem was $\mathsf{R}(f \circ g) \ge \Omega(\mathsf{bs}(f)\mathsf{R}(g))$, a proof of which is virtually the same as for the result that $\mathsf{R}(\textsc{And}_n \circ g) \ge \Omega(n\mathsf{R}(g))$; see [11, §5.1]. In fact, we were originally motivated to consider the correlated samples problem when trying to strengthen this composition result from block sensitivity to fractional block sensitivity.

1.5 Independent work by Ben-David and Blais

In an independent and concurrent work, Ben-David and Blais [5, 6] have also studied the randomized composition conjecture and ways of circumventing Shaltiel examples via improved minimax theorems. They develop a powerful framework for constructing hard *Shaltiel-free* distributions, which is general enough to apply not only to query complexity but also, for instance, to communication complexity. In particular, their framework is able to give an alternative proof of our main result (Theorem 1.2) as well as our fbs-based composition theorem (Theorem 1.6). Their proof techniques involve information theory and analysis; by contrast, our techniques are more elementary and directly tailored to the correlated samples problem (which does not explicitly appear in their work).

1.6 Roadmap

We will prove our main theorem (Theorem 1.2) in Section 3 and Section 4. Before that, we introduce our basic notions regarding decision trees in Section 2. In Section 3, we characterize decision trees as *likelihood boosters*, emphasizing that a good query algorithm must make significant progress in terms of boosting the likelihood of one of the outputs (0 or 1) to much higher than the other, and vice versa. This characterization frees us from considering inputs from both $\mathcal{D}_0$ and $\mathcal{D}_1$ simultaneously: if an algorithm is certain about the output on $\mathcal{D}_1$, then it must also make few errors on $\mathcal{D}_0$. We thus reduce the proof of Theorem 1.2 to bootstrapping decision trees that can make *overall* progress across multiple samples to a decision tree that makes uniform progress. In Section 4, we build such a bootstrapping algorithm and show that it makes satisfactory progress with a careful analysis. Proofs for our two applications are in Section 5 and Section 6.

2 Preliminaries

Let $f : \Sigma^n \to \{0, 1, *\}$ be a partial function for some alphabet Σ (typically $\Sigma = \{0, 1\}$). Let $\mathcal{D}_0, \mathcal{D}_1$ be distributions supported on $f^{-1}(0)$, $f^{-1}(1)$ respectively. For each $x \in \Sigma^n$, let $\mathcal{D}_0(x)$ (resp. $\mathcal{D}_1(x)$) denote the probability mass on x in distribution $\mathcal{D}_0$ (resp. $\mathcal{D}_1$). For a subset $S \subseteq \Sigma^n$, we define $\mathcal{D}_b(S) = \sum_{x \in S} \mathcal{D}_b(x)$ for $b = 0, 1$. If $\mathcal{D}_b(S) > 0$, we define the conditional distribution $\mathcal{D}_b|_S$ by $\mathcal{D}_b|_S(x) = \frac{\mathcal{D}_b(x)}{\mathcal{D}_b(S)}$ when $x \in S$, and $\mathcal{D}_b|_S(x) = 0$ when $x \notin S$. We define the *likelihood-ratio* of S as

$$\mathsf{LR}(S) := \frac{\mathcal{D}_1(S)}{\mathcal{D}_0(S)}.$$

Let T be a deterministic decision tree that takes as input a sample $x \in \Sigma^n$ drawn from either $\mathcal{D}_0$ or $\mathcal{D}_1$. For every vertex v in T, we use $\mathsf{Input}(v) \subseteq \Sigma^n$ to denote the set of strings that can reach v, or equivalently, the set of strings that agree with all the queries made so far. Typically, every non-leaf vertex in T corresponds to a query to a certain position in the sample, but we will allow non-leaf vertices v in T that do not make any query, each of them having only a single child v' with $\mathsf{Input}(v') = \mathsf{Input}(v)$. We abuse notation slightly and use v as a shorthand for $\mathsf{Input}(v)$, so we have $\mathcal{D}_0(v) = \sum_{x \in v} \mathcal{D}_0(x)$, $\mathcal{D}_1(v) = \sum_{x \in v} \mathcal{D}_1(x)$ and

$$\mathsf{LR}(v) = \frac{\mathcal{D}_1(v)}{\mathcal{D}_0(v)}.$$

Note that the likelihood-ratio $\mathsf{LR}(v)$ is non-negative, but could be zero or infinite. We can eliminate the undefined case ($\mathcal{D}_0(v) = \mathcal{D}_1(v) = 0$) by trimming the unreachable parts of the decision tree.

Now if the decision tree T takes as input k samples from Σ^n, it is not hard to see that $\mathsf{Input}(v)$ can be written as a Cartesian product $\mathsf{Input}(v) = \mathsf{Input}_1(v) \times \cdots \times \mathsf{Input}_k(v)$, where $\mathsf{Input}_j(v) \subseteq \Sigma^n$ is the set of strings that agree with all the queries made to the j-th sample so far. Again, we abuse notation slightly and use v_j as a shorthand for $\mathsf{Input}_j(v)$, so we will often write $v = v_1 \times \cdots \times v_k$. We define the *overall likelihood ratio* of v as the product

$$\mathsf{OLR}(v) := \mathsf{LR}(v_1) \cdots \mathsf{LR}(v_k) = \frac{\mathcal{D}_1(v_1)}{\mathcal{D}_0(v_1)} \cdots \frac{\mathcal{D}_1(v_k)}{\mathcal{D}_0(v_k)}.$$

It is often more convenient to consider the logarithm of likelihood ratios. We will use natural logarithm throughout the paper, i.e. $\log(\cdot) = \ln(\cdot)$.

3 Query Algorithms as Likelihood Boosters

Our overarching goal (Theorem 1.2) is to construct an efficient deterministic query algorithm that distinguishes $\mathcal{D}_0$ from $\mathcal{D}_1$, assuming the existence of one that distinguishes $\mathcal{D}_0^k$ from $\mathcal{D}_1^k$. As the starting point, we introduce the notion of *likelihood boosters* as a way of measuring the progress made by a query algorithm T in distinguishing $\mathcal{D}_0$ from $\mathcal{D}_1$. The key idea is that, as more queries are being made, the algorithm narrows down the possibilities of the unknown input, driving the likelihood of one of the output (0 or 1) much higher than the other. In fact, we show that T can distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ well if and only if a sample drawn from $\mathcal{D}_1$ has a high probability of arriving at a leaf of T where most of the remaining possibilities produce output 1. (Lemma 3.4 and Lemma 3.5).

In the multiple-sample setting, we use the notions of *overall likelihood boosters* and *uniform likelihood boosters*, which have different levels of guarantees, to measure the progress of a query algorithm on simultaneously classifying each of the samples in the input. We show that an efficient query algorithm that distinguishes $\mathcal{D}_0^k$ from $\mathcal{D}_1^k$ is an efficient overall likelihood booster (Corollary 3.6). Moreover, we show that an efficient uniform likelihood booster on multiple samples induces an efficient likelihood booster on a single sample (Lemma 3.7), which in turn implies an efficient query algorithm that distinguishes $\mathcal{D}_0$ from $\mathcal{D}_1$ (Lemma 3.4). These results will enable us to reduce proving Theorem 1.2 to relating overall likelihood boosters to uniform likelihood boosters, which is the focus of Section 4 (see Theorem 4.1).

We now formally define the three types of likelihood boosters mentioned above:

▶ **Definition 3.1.** *We say a deterministic decision tree T is a (δ, M)-*likelihood booster *for $\mathcal{D}_0, \mathcal{D}_1$ if, with probability at least $1-\delta$, an input sample drawn from $\mathcal{D}_1$ reaches a leaf ℓ of T with likelihood ratio $\mathsf{LR}(\ell) \geq M$.*

▶ **Definition 3.2.** *We say a deterministic decision tree T is a (δ, M)-*overall likelihood booster *for $\mathcal{D}_0^k, \mathcal{D}_1^k$ if, with probability at least $1-\delta$, an input drawn from $\mathcal{D}_1^k$ consisting of k samples reaches a leaf ℓ of T with* overall *likelihood ratio $\mathsf{OLR}(\ell) \geq M$.*

▶ **Definition 3.3.** *We say a deterministic decision tree T is a (δ, ε, M)-*uniform likelihood booster *for $\mathcal{D}_0^k$ and $\mathcal{D}_1^k$ if, with probability at least $1-\delta$, an input x drawn from $\mathcal{D}_1^k$ consisting of k samples reaches a leaf $\ell = \ell_1 \times \cdots \times \ell_k$ of T with the property that at least $(1-\varepsilon)k$ different samples $j \in \{1, \cdots, k\}$ satisfy $\mathsf{LR}(\ell_j) \geq M$.*

Note that the above definitions do not depend on the actual output of the decision tree T. We now show in the following two lemmas that likelihood boosters are in some sense equivalent to query algorithms that distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$.

▶ **Lemma 3.4.** *Suppose T is a (δ, M)-likelihood booster for $\mathcal{D}_0, \mathcal{D}_1$. Consider the deterministic decision tree T' that makes exactly the same queries as T and accepts if and only if a leaf ℓ with $\mathsf{LR}(\ell) \geq M$ is reached. Then T' distinguishes $\mathcal{D}_0$ from $\mathcal{D}_1$ with the following guarantees:*

1. *(Completeness) T' accepts $x \sim \mathcal{D}_1$ with probability at* least *$1-\delta$.*
2. *(Soundness) T' accepts $x \sim \mathcal{D}_0$ with probability at* most *$1/M$.*

Proof. Completeness follows directly from the definition of likelihood booster. To prove soundness, consider the set U of leaves ℓ with $\mathsf{LR}(\ell) \geq M$. For all $\ell \in U$, we have $\mathcal{D}_0(\ell) \leq \frac{1}{M}\mathcal{D}_1(\ell)$. Therefore, $\sum_{\ell \in U} \mathcal{D}_0(\ell) \leq \frac{1}{M}\sum_{\ell \in U} \mathcal{D}_1(\ell) \leq \frac{1}{M}$. This means that a sample from $\mathcal{D}_0$ reaches leaves in U with probability at most $\frac{1}{M}$, which is exactly the desired soundness. ◀

▶ **Lemma 3.5.** *Suppose a deterministic decision tree T can distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$ with the following guarantees: T accepts $x \sim \mathcal{D}_0$ with probability at most δ_0, and accepts $x \sim \mathcal{D}_1$ with probability at least $1-\delta_1$. Then T is a $(M\delta_0 + \delta_1, M)$-likelihood booster for any $M > 0$.*

Proof. Let U denote the set of leaves ℓ with $\mathsf{LR}(\ell) < M$. We can partition U as $U = U_0 \cup U_1$, where U_1 corresponds to the leaves at which T accepts. Since T accepts with probability at most δ_0 on $\mathcal{D}_0$, we have $\sum_{\ell \in U_1} \mathcal{D}_0(\ell) \leq \delta_0$. Similarly, we have $\sum_{\ell \in U_0} \mathcal{D}_1(\ell) \leq \delta_1$. Therefore,

$$\sum_{\ell \in U} \mathcal{D}_1(\ell) = \sum_{\ell \in U_0} \mathcal{D}_1(\ell) + \sum_{\ell \in U_1} \mathcal{D}_1(\ell) \leq \sum_{\ell \in U_0} \mathcal{D}_1(\ell) + M \sum_{\ell \in U_1} \mathcal{D}_0(\ell) = \delta_1 + M\delta_0.$$

In other words, a sample from $\mathcal{D}_1$ has probability at most $M\delta_0 + \delta_1$ of reaching a leaf in U, which means that T is a $(M\delta_0 + \delta_1, M)$-likelihood booster. ◀

In the multiple-sample setting, if we view the pair $\mathcal{D}_0^k$ and $\mathcal{D}_1^k$ as $\mathcal{D}_0'$ and $\mathcal{D}_1'$ in the single-sample setting with input length multiplied by k, the definition of overall likelihood ratio coincides with the definition of likelihood ratio in the single-sample setting. Therefore, we have the following corollary of Lemma 3.5, which essentially shows that an efficient query algorithm for the correlated samples problem is an efficient overall likelihood booster:

▶ **Corollary 3.6.** *Suppose a deterministic decision tree T can distinguish $\mathcal{D}_0^k$ from $\mathcal{D}_1^k$ in that T accepts $x \sim \mathcal{D}_0^k$ with probability at most δ_0, and T accepts $x \sim \mathcal{D}_1^k$ with probability at least $1 - \delta_1$. Then T is a $(M\delta_0 + \delta_1, M)$-overall likelihood booster for any $M > 0$.*

To conclude this section, we show that an efficient uniform likelihood booster in the multiple-sample setting implies an efficient likelihood booster in the single-sample setting.

▶ **Lemma 3.7.** *For any (δ, ε, M)-*uniform likelihood booster *T for $\mathcal{D}_0^k$ and $\mathcal{D}_1^k$ and any $C > 0$, there is a $(\delta + \varepsilon + \frac{1}{C}, M)$-likelihood booster T' for $\mathcal{D}_0$ and $\mathcal{D}_1$ with $\mathrm{depth}(T') \le C \cdot \frac{\mathrm{depth}(T)}{k}$.*

Proof. Define $Q = C \cdot \frac{\mathrm{depth}(T)}{k}$. We first build a randomized query algorithm $\mathcal{A}'$ for $\mathcal{D}_0$ and $\mathcal{D}_1$, and later derandomize it as T'. On input $x_{\mathcal{A}'}$, $\mathcal{A}'$ generates k random samples $(x_1, \ldots, x_k) \sim \mathcal{D}_1^k$, selects a uniformly random index j, replaces x_j with $\mathcal{A}'$'s own input $x_{\mathcal{A}'}$, and finally simulates T on the modified k samples $(x_1, \ldots, x_{\mathcal{A}'}, \ldots, x_k)$. If T attempts to make the $(\lfloor Q \rfloor + 1)$-th query to the j-th (modified) sample, $\mathcal{A}'$ halts.

It is easy to see that the maximum number of queries made by $\mathcal{A}'$ is at most Q. Moreover, by Markov's inequality, if the input $x_{\mathcal{A}'}$ to $\mathcal{A}'$ is drawn from $\mathcal{D}_1$, the probability that $\mathcal{A}'$ halts early because of T making more than Q queries to the j-th sample is at most $\frac{1}{C}$, since the average number of queries T makes to the j-th sample for a uniformly random j is at most $\frac{\mathrm{depth}(T)}{k}$.

We now show that with probability at least $1-(\delta+\varepsilon+\frac{1}{C})$, $\mathcal{A}'$ reaches a leaf $\ell = \ell_1 \times \cdots \times \ell_k$ of T with $\mathsf{LR}(\ell_j) \ge M$ when its own input $x_{\mathcal{A}'}$ is drawn from $\mathcal{D}_1$. By a union bound, we only need to show that this holds with probability at least $1 - (\delta + \varepsilon)$ for the extended version of $\mathcal{A}'$ that doesn't halt early. If we switch the order of randomness so that j is chosen after a leaf of T is reached, this follows easily from the definition of uniform likelihood boosters (Definition 3.3).

Finally, we derandomize $\mathcal{A}'$. Note that the randomness in $\mathcal{A}'$ only comes from the randomness in j and in all the generated samples x_i except the j-th sample. We can simply fix them so that the probability of reaching a leaf ℓ of T with $\mathsf{LR}(\ell_j) \ge M$ is maximized, assuming that the j-th sample is from $\mathcal{D}_1$. Since j and all generated samples other than the j-th sample have been fixed, the decision tree T now "shrinks" to a decision tree T' with only the first $\lfloor Q \rfloor$ queries to the j-th sample remaining, and every leaf ℓ of T that is reachable when we run $\mathcal{A}'$ now becomes a leaf ℓ' of T'. Shrinking the tree doesn't affect the computation history regarding the j-th sample, so we have $\ell' = \mathsf{Input}(\ell') = \mathsf{Input}_j(\ell) = \ell_j$ and $\mathsf{LR}(\ell') = \mathsf{LR}(\ell_j)$. This proves that T' is a $(\delta + \varepsilon + \frac{1}{C}, M)$-likelihood booster. ◀

4 Bootstrapping Overall Booster to Uniform Booster

The results from the previous section (Section 3) reduce proving our main result (Theorem 1.2) to proving relations between overall likelihood boosters and uniform likelihood boosters. In this section, we complete this step with the following result:

▶ **Theorem 4.1.** *Assume that there is a depth-L $(0.1, 25)$-overall likelihood booster for* every *distribution pair $\mathcal{D}_0^k, \mathcal{D}_1^k$. Then there is a depth-$O(KL)$ $(0.1, 0.1, 100)$-uniform likelihood booster for every distribution pair $\mathcal{D}_0^K, \mathcal{D}_1^K$ whenever $K \ge 1000k(|\Sigma| + 1)^n$.*

We first show how to derive Theorem 1.2 from Theorem 4.1:

Proof of Theorem 1.2. We prove the inequality $\mathsf{R}(f) \le O(\mathsf{Corr}(f))$ (the converse inequality is trivial). Suppose we have a depth-L deterministic decision tree that solves the correlated samples problem on $\frac{1}{2}\mathcal{D}_0^k + \frac{1}{2}\mathcal{D}_1^k$ with success probability at least 0.999 (recall that the success probability can be amplified by Fact 1.1). That is, the decision tree accepts inputs drawn from $\mathcal{D}_1^k$ with probability at *least* 0.998 and accepts inputs drawn from $\mathcal{D}_0^k$ with probability at *most* 0.002. By Corollary 3.6, it is a $(0.1, 25)$-overall likelihood booster for $\mathcal{D}_0^k$ and $\mathcal{D}_1^k$.

By Theorem 4.1, for any pair of distributions $\mathcal{D}_0^K, \mathcal{D}_1^K$, there is a $(0.1, 0.1, 100)$-uniform likelihood booster with depth $O(KL)$. Then by Lemma 3.7, there is a $(1/3, 100)$-likelihood booster with depth $O(L)$ for $\mathcal{D}_0$ and $\mathcal{D}_1$, which by Lemma 3.4 implies a query algorithm for $\mathcal{D}_0$ and $\mathcal{D}_1$ with success probability at least 1/3. By the arbitrariness of $\mathcal{D}_0$ and $\mathcal{D}_1$, we have $\mathsf{R}_{1/3}(f) = O(L)$ via Yao's minimax, as desired. ◀

The rest of this section is dedicated to proving Theorem 4.1. We construct the desired uniform likelihood booster $T_{\text{bootstrap}}$, described in Subsection 4.1, by applying different overall likelihood boosters to appropriate sets of samples at different phases of computation. To quantify the progress made by $T_{\text{bootstrap}}$, we design a measure based on a "truncated" log likelihood ratio which handles samples that $T_{\text{bootstrap}}$ is confident about with special care. As the technical core of the proof, we show that under our carefully constructed measure, $T_{\text{bootstrap}}$ in expectation makes *positive* and *constant* progress during each phase of computation (Lemmas 4.2 and 4.3). Therefore, $T_{\text{bootstrap}}$ is able to achieve the desired guarantees after sufficiently many phases.

4.1 Bootstrapping algorithm

We describe our depth-$O(KL)$ $(0.1, 0.1, 100)$-uniform likelihood booster $T_{\text{bootstrap}}$ taking $K \geq 1000k(|\Sigma| + 1)^n$ samples. Recall that each vertex v of $T_{\text{bootstrap}}$ can be written as a Cartesian product $v = v_1 \times \cdots \times v_K$, where $v_j \subseteq \Sigma^n$ is the set of strings that are consistent with the queries made to the j-th sample so far. We say that the j-th sample is *settled* at v if

$$\mathsf{LR}(v_j) = \frac{\mathcal{D}_1(v_j)}{\mathcal{D}_0(v_j)} \notin [e^{-100}, e^{100}].$$

Note that it is possible for a sample to be settled in the wrong direction (e.g. $\mathsf{LR}(v_j) < e^{-100}$ on input drawn from $\mathcal{D}_1^K$), but we will show that this is not a serious issue.

The query algorithm $T_{\text{bootstrap}}$ proceeds in at most $C \cdot K$ phases (for some large constant $C > 0$). Each phase consists of at most L queries and is described as follows:

Phase $s = 1, \cdots, C \cdot K$:

1. If fewer than $k(|\Sigma| + 1)^n$ out of the K samples are unsettled, halt.
2. Else, since each v_j is determined by a string v_* in $(\Sigma \cup \{*\})^n$ recording the queries made so far to the j-th sample, by the Pigeonhole Principle there exist k unsettled samples $j_1, \cdots, j_k$ with $v_{j_1} = \cdots = v_{j_k} = v_*$.
3. Run the depth-L $(0.1, 25)$-overall likelihood booster $A^{(v_*)}$, assumed in Theorem 4.1 to exist, relative to the input-distribution pair

   $$(\mathcal{D}_0|_{v_*})^k \ , \quad (\mathcal{D}_1|_{v_*})^k$$

   on the samples

   $$(x^{j_1}, \ldots, x^{j_k}) \ .$$

   If any query causes one of these samples to become settled (i.e. $\mathsf{LR}(v_{j_i}) \notin [e^{-100}, e^{100}]$ for some $i \in \{1, \cdots, k\}$), halt $A^{(v_*)}$ and go to the next Phase. Otherwise we proceed to the next Phase after $A^{(v_*)}$ terminates. If fewer than L queries are made in the current phase, insert dummy vertices that do not make any query (see Section 2) to $T_{\text{bootstrap}}$ so that each phase corresponds to a path in $T_{\text{bootstrap}}$ with length exactly L.

4.2 Sub-martingale property of progress measure

It's not hard to see that the overall likelihood ratio (OLR) is *not* an effective measure of progress for $T_{\text{bootstrap}}$: OLR can rocket to infinity even when there is only one settled sample. In this subsection, we introduce a better progress measure: *overall truncated log likelihood ratio* (OTLLR), and show that it is a sub-martingale along the computation path of *any* decision tree (Lemma 4.2). In other words, $T_{\text{bootstrap}}$ always makes *non-negative* progress in expectation. We will show that each phase of $T_{\text{bootstrap}}$ makes *positive* expected progress in the next subsection (Subsection 4.3).

Let T be a deterministic decision tree that takes as input K samples. For every vertex $v = v_1 \times \cdots \times v_K$ of T, we define the *truncated log likelihood ratio* of v_j as

$$\mathsf{TLLR}(v_j) := \begin{cases} \log(\mathsf{LR}(v_j)), & \text{if } |\log(\mathsf{LR}(v_j))| \le 100, \\ 500, & \text{otherwise.} \end{cases}$$

Note that if $\log(\mathsf{LR}(v_j))$ slightly exceeds the upper threshold 100, we set TLLR to a much higher value 500. Also, when $\log(\mathsf{LR}(v_j))$ drops below the lower threshold -100, we also set TLLR to 500. Thus, the j-th sample is *settled* at v if and only if $\mathsf{TLLR}(v_j) = 500$.

We define the *overall truncated log-likelihood-ratio* of v as the sum

$$\mathsf{OTLLR}(v) := \sum_{j=1}^{K} \mathsf{TLLR}(v_j).$$

The input x to T determines a computation path from the root of T to a leaf: $v^0 \to v^1 \to \cdots \to v^q$. The randomness in x transfers to the randomness in the path, so the path is a stochastic process. We now show that $\mathsf{OTLLR}(v^t)$ along the path is a sub-martingale when x is drawn from $\mathcal{D}_1^K$:

▶ **Lemma 4.2.** *Assume that T never queries a settled sample. Assume that the input x to T is drawn from $\mathcal{D}_1^K$, v is a non-leaf vertex with distance t from the root, and v is reachable (i.e.* $\Pr[v^t = v] > 0$ *on $\mathcal{D}_1^K$). Define $\Delta^t := \mathsf{OTLLR}(v^{t+1}) - \mathsf{OTLLR}(v^t)$. Then we have*

$$\mathbb{E}[\Delta^t | v^t = v] \ge 0.001 \cdot \mathbb{E}[(\Delta^t)^2 | v^t = v] \ge 0.$$

Proof. Let us condition on $v^t = v$ in the whole proof. If v is a dummy vertex that does not make any query, then $\Delta^t = 0$ deterministically and the lemma holds trivially. We assume that v is not a dummy vertex henceforth.

Suppose sample j is queried at vertex v. We have $\mathsf{OTLLR}(v^{t+1}) - \mathsf{OTLLR}(v^t) = \mathsf{TLLR}(v_j^{t+1}) - \mathsf{TLLR}(v_j^t)$. Since T never queries a settled sample, we know $\mathsf{TLLR}(v_j^t) = \log \frac{\mathcal{D}_1(v_j^t)}{\mathcal{D}_0(v_j^t)} \in [-100, 100]$.

Let $\sigma \in \Sigma$ denote the random outcome of the query, and let $p_0(\sigma), p_1(\sigma)$ denote the probability that the outcome to the query is σ under $\mathcal{D}_0|_{v_j^t}, \mathcal{D}_1|_{v_j^t}$, respectively. Let $H \subseteq \Sigma$ denote the set of $\sigma \in \Sigma$ with $|\mathsf{TLLR}(v_j^t) + \log \frac{p_1(\sigma)}{p_0(\sigma)}| > 100$. Note that $\mathcal{D}_0(v_j^{t+1}) = \mathcal{D}_0(v_j^t) p_0(\sigma)$ and $\mathcal{D}_1(v_j^{t+1}) = \mathcal{D}_1(v_j^t) p_1(\sigma)$, so

$$\mathsf{TLLR}(v_j^{t+1}) = \begin{cases} \mathsf{TLLR}(v_j^t) + \log \frac{p_1(\sigma)}{p_0(\sigma)}, & \sigma \notin H, \\ 500, & \sigma \in H. \end{cases}$$

Thus, H is precisely the set of outcomes $\sigma \in \Sigma$ that make sample j settled at v^{t+1}. Let $W = W(\sigma)$ denote the difference $\mathsf{TLLR}(v_j^{t+1}) - \mathsf{TLLR}(v_j^t)$. Our goal is to prove $\mathbb{E}[W] \ge 0.001 \cdot \mathbb{E}[W^2]$.

Note that $W(\sigma) \in [400, 600]$ when $\sigma \in H$ and $W(\sigma) = \log \frac{p_1(\sigma)}{p_0(\sigma)} \in [-200, 200]$ when $\sigma \notin H$. We have

$$\begin{aligned}\mathbb{E}[W] \geq & 400 \sum_{\sigma \in H} p_1(\sigma) + \sum_{\sigma \notin H} p_1(\sigma) \log \frac{p_1(\sigma)}{p_0(\sigma)} \\ = & 400 \sum_{\sigma \in H} p_1(\sigma) + \sum_{\sigma \notin H} p_0(\sigma) \cdot \frac{p_1(\sigma)}{p_0(\sigma)} \log \frac{p_1(\sigma)}{p_0(\sigma)}. \qquad (1)\end{aligned}$$

By a helper lemma (Lemma 4.4) proved in Subsection 4.4, we know that

$$\frac{p_1(\sigma)}{p_0(\sigma)} \log \frac{p_1(\sigma)}{p_0(\sigma)} \geq \left(\frac{p_1(\sigma)}{p_0(\sigma)} - 1\right) + \frac{1}{400} \cdot \frac{p_1(\sigma)}{p_0(\sigma)} \left(\log \frac{p_1(\sigma)}{p_0(\sigma)}\right)^2.$$

Plugging this into (1), we have

$$\begin{aligned}\mathbb{E}[W] \geq & 400 \sum_{\sigma \in H} p_1(\sigma) + \sum_{\sigma \notin H} p_1(\sigma) - \sum_{\sigma \notin H} p_0(\sigma) + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma) \left(\log \frac{p_1(\sigma)}{p_0(\sigma)}\right)^2 \\ \geq & 400 \sum_{\sigma \in H} p_1(\sigma) + \left(\sum_{\sigma \notin H} p_1(\sigma) - 1\right) + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma) \left(\log \frac{p_1(\sigma)}{p_0(\sigma)}\right)^2 \\ = & 400 \sum_{\sigma \in H} p_1(\sigma) - \sum_{\sigma \in H} p_1(\sigma) + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma) \left(\log \frac{p_1(\sigma)}{p_0(\sigma)}\right)^2 \\ = & 399 \sum_{\sigma \in H} p_1(\sigma) + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma) \left(\log \frac{p_1(\sigma)}{p_0(\sigma)}\right)^2 \\ = & 399 \sum_{\sigma \in H} p_1(\sigma) + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma)(W(\sigma))^2 \\ \geq & \frac{1}{1000} \sum_{\sigma \in H} p_1(\sigma)(W(\sigma))^2 + \frac{1}{400} \sum_{\sigma \notin H} p_1(\sigma)(W(\sigma))^2 \\ \geq & \frac{1}{1000} \mathbb{E}[W^2].\end{aligned}$$

◀

4.3 Bounding the conditional expectation of progress

In the previous subsection, we showed that OTLLR, as a progress measure, is a sub-martingale. Now we refine our progress measure to also include the natural measure *number of settled samples*, and show that each phase of $T_{\text{bootstrap}}$ makes *positive* progress in expectation.

Recall that we inserted dummy vertices in $T_{\text{bootstrap}}$ to ensure that each phase corresponds to a computation path of length exactly L. Therefore, an entire computation path of $T_{\text{bootstrap}}$ must have length divisible by L: $v^0 \to \cdots \to v^{qL}$. The sub-path $v^{tL} \to \cdots \to v^{(t+1)L}$ is the computation path of the $(t+1)$-th phase.

Define $\mathsf{S}(v)$ as the number of settled samples at vertex v. Our new measure of progress is

$$\mathsf{P}(v^t) := \mathsf{S}(v^t) + \mathsf{OTLLR}(v^t).$$

▶ **Lemma 4.3.** *Assume that the input x to $T_{\text{bootstrap}}$ is drawn from $\mathcal{D}_1^K$, v is a non-leaf vertex with distance tL from the root, and v is reachable (i.e.* $\Pr[v^{tL} = v] > 0$ *on $\mathcal{D}_1^K$). Then we have*

$$\mathbb{E}[\mathsf{P}(v^{(t+1)L}) - \mathsf{P}(v^{tL}) | v^{tL} = v] \geq 0.001.$$

Before proving the lemma, we first show how it implies Theorem 4.1.

Proof of Theorem 4.1. We consider an extended version of $T_{\text{bootstrap}}$ that always halts after exactly $C \cdot K$ phases: whenever it would halt at line 1, it instead enters dummy phases and increases its total progress P by 0.001 per phase (so that now $\mathsf{P} = \mathsf{S} + \mathsf{OTLLR} + 0.001 \cdot$ number of dummy phases). By Lemma 4.3, the extended algorithm finishes with expected total progress $\mathbb{E}[\mathsf{P}] \geq 0.001C \cdot K$ on input drawn from $\mathcal{D}_1^K$. However, P can never grow too large: before any dummy phase, P is at most $501K$, and there are at most $C \cdot K$ dummy phases, so $\mathsf{P} \leq 501K + 0.001C \cdot K$. By Markov's inequality on the non-negative random variable $(501K + 0.001C \cdot K) - \mathsf{P}$, we have $\Pr[\mathsf{P} \leq 501K] \leq \frac{501K}{0.001C \cdot K} = \frac{501}{0.001C}$. If we choose a large enough C, we know that with probability at least 0.99, the total progress exceeds $501K$, which means that the extended algorithm enters dummy phases before halting, and the original algorithm halts at line 1 with all but 0.001 fraction of the samples settled.

It now suffices to show that the fraction of samples settled in the wrong direction (i.e., the likelihood ratio drops below e^{-100}) is at most 0.01 with probability at least 0.99. We first fix j and show that the probability that the j-th sample is settled in the wrong direction is at most e^{-100}, and then use the linearity of expectation and Markov's inequality to bound the overall wrong settlement.

Conditioning on all but the j-th sample, $T_{\text{bootstrap}}$ becomes a deterministic decision tree T' on a single sample. Let U denote the set of leaves ℓ of T' with $\mathsf{LR}(\ell) \leq e^{-100}$. We have $\sum_{\ell \in U} \mathcal{D}_1(\ell) \leq e^{-100} \sum_{\ell \in U} \mathcal{D}_0(\ell) \leq e^{-100}$. This means that the probability that a sample from $\mathcal{D}_1$ reaches leaves in U is at most e^{-100}. Thus the probability of wrong settlement for sample j in $T_{\text{bootstrap}}$ is at most e^{-100}.

By the linearity of expectation, the expected fraction of samples settled in the wrong direction is at most e^{-100}. Then by Markov's inequality, with probability at least 0.99, the fraction of wrong settlement is at most 0.01. ◀

Proof of Lemma 4.3. $\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL})$ is either 0 or 1, depending on whether or not a sample becomes settled in phase $t+1$.

In the case where $\Pr[\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL}) = 1 | v^{tL} = v] \geq 0.001$, we have $\mathbb{E}[\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL}) | v^{tL} = v] \geq 0.001$, and by Lemma 4.2 we have $\mathbb{E}[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL}) | v^{tL} = v] \geq 0$. Summing these two inequalities up proves the lemma.

From now on, we consider the harder case where $\Pr[\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL}) = 1 | v^{tL} = v] < 0.001$. We first prove that

$$\Pr[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL}) \geq 3 | v^{tL} = v] \geq 0.8. \tag{2}$$

Recall that in this phase $T_{\text{bootstrap}}$ runs the $(0.1, 25)$-overall likelihood booster $A^{(v_*)}$ for $(\mathcal{D}_0|_{v_*})^k$ and $(\mathcal{D}_1|_{v_*})^k$ on the samples $j_1, \ldots, j_k$. If $\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL}) = 0$, i.e. no sample becomes settled in this phase, then

$$\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL}) = \sum_{s=1}^{k} \left(\log \frac{\mathcal{D}_1(v_{j_s}^{(t+1)L})}{\mathcal{D}_0(v_{j_s}^{(t+1)L})} - \log \frac{\mathcal{D}_1(v_{j_s}^{tL})}{\mathcal{D}_0(v_{j_s}^{tL})} \right).$$

Conditioning on $v^{tL} = v$, we have $v_{j_s}^{tL} = v_*$, since $v_{j_1} = \cdots = v_{j_k} = v_*$. From $\frac{\mathcal{D}_b(v_{j_s}^{(t+1)L})}{\mathcal{D}_b(v_*)} = \mathcal{D}_b|_{v_*}(v_{j_s}^{(t+1)L})$, we see that

$$\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL}) = \log \prod_{s=1}^{k} \frac{\mathcal{D}_1|_{v_*}(v_{j_s}^{(t+1)L})}{\mathcal{D}_0|_{v_*}(v_{j_s}^{(t+1)L})}.$$

Recall that $A^{(v_*)}$ halts early only when a new sample becomes settled, which happens with probability < 0.001. Therefore, in order to prove (2) by a union bound, we only need to prove that the extended version of phase $t+1$ where $A^{(v_*)}$ gets to run without early halting achieves $\prod_{s=1}^{k} \frac{\mathcal{D}_1|_{v_*}(v_{j_s}^{(t+1)L})}{\mathcal{D}_0|_{v_*}(v_{j_s}^{(t+1)L})} \geq e^3$ with probability at least 0.9. This is indeed true because $A^{(v_*)}$ is a $(0.1, 25)$-overall likelihood booster for $(\mathcal{D}_0|_{v_*})^k$ and $(\mathcal{D}_1|_{v_*})^k$.

We now prove $\mathbb{E}[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL})|v^{tL} = v] \geq 0.001$. We prove it by contradiction. Suppose $\mathbb{E}[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL})|v^{tL} = v] < 0.001$. For $tL \leq s < (t+1)L$, define $\Delta(v^s)$ as the conditional expectation $\mathbb{E}[\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s)|v^s]$ and $\Delta_2(v^s)$ as the conditional variance $\mathbb{E}[(\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s) - \Delta(v^s))^2|v^s]$. Note that

$$\begin{aligned}\Delta_2(v^s) &= \mathbb{E}[((\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s))^2|v^s] - (\Delta(v^s))^2\\ &\leq \mathbb{E}[((\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s))^2|v^s].\end{aligned}$$

Thus by Lemma 4.2, we know that $\Delta(v^s) \geq 0.001 \cdot \Delta_2(v^s) \geq 0$. Now we have

$$\begin{aligned}0.001 >&\mathbb{E}[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL})|v^{tL} = v]\\ =& \sum_{tL \leq s < (t+1)L} \mathbb{E}[\Delta(v^s)|v^{tL} = v].\end{aligned}$$

By Markov's inequality, we have $\Pr\left[\sum_{tL \leq s<(t+1)L} \Delta(v^s) \geq 1|v^{tL} = v\right] \leq 0.001$. Now by a union bound with (2), we have

$$\begin{aligned}&\mathbb{E}\left[\left(\sum_{tL\leq s<(t+1)L} (\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s) - \Delta(v^s))\right)^2 \middle| v^{tL} = v\right]\\ =&\mathbb{E}\left[\left((\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL})) - \sum_{tL\leq s<(t+1)L} \Delta(v^s)\right)^2 \middle| v^{tL} = v\right]\\ \geq&(0.8 - 0.001) \times (3-1)^2\\ >&3. \end{aligned} \tag{3}$$

Since $\mathbb{E}\left[\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s) - \Delta(v^s)|v^s\right] = 0$, we have

$$\begin{aligned}&\mathbb{E}\big[(\mathsf{OTLLR}(v^{s_1+1}) - \mathsf{OTLLR}(v^{s_1}) - \Delta(v^{s_1}))\cdot\\ &\quad(\mathsf{OTLLR}(v^{s_2+1}) - \mathsf{OTLLR}(v^{s_2}) - \Delta(v^{s_2}))|v^{tL} = v\big] = 0\end{aligned}$$

whenever $s_1 < s_2$ by further conditioning on v^{s_2}. Thus expanding (3) we have

$$\begin{aligned}&\sum_{tL\leq s<(t+1)L} \mathbb{E}[\Delta_2(v^s)|v^{tL} = v]\\ =&\mathbb{E}\left[\sum_{tL\leq s<(t+1)L} (\mathsf{OTLLR}(v^{s+1}) - \mathsf{OTLLR}(v^s) - \Delta(v^s))^2 \middle| v^{tL} = v\right] \geq 3.\end{aligned}$$

Since $\Delta(v^s) \geq 0.001 \cdot \Delta_2(v^s)$, we have

$$0.001 > \sum_{tL\leq s<(t+1)L} \mathbb{E}[\Delta(v^s)|v^{tL} = v] \geq 0.001 \cdot \sum_{tL\leq s<(t+1)L} \mathbb{E}[\Delta_2(v^s)|v^{tL} = v] \geq 0.001 \times 3,$$

a contradiction.

Now we have shown $\mathbb{E}[\mathsf{OTLLR}(v^{(t+1)L}) - \mathsf{OTLLR}(v^{tL})|v^{tL} = v] \geq 0.001$. Adding it to the trivial inequality $\mathbb{E}[\mathsf{S}(v^{(t+1)L}) - \mathsf{S}(v^{tL})|v^{tL} = v] \geq 0$ proves the lemma. ◀

4.4 A helper inequality

▶ **Lemma 4.4.** *For all $M \geq 0, t \in (0, e^M]$, we have*

$$t \ln t - (t-1) \geq \frac{1}{M+2} \cdot t \ln^2 t.$$

Proof. Define function $h(t) = t \ln t - (t-1) - \frac{1}{M+2} \cdot t \ln^2 t$ on the interval $t \in (0, e^M]$. Our goal is to show $h(t) \geq 0$. Note that $h(1) = 0$, so we only need to show $h'(t) \geq 0$ for $t \geq 1$ and $h'(t) \leq 0$ for $t \leq 1$. We prove this by calculating $h'(t)$:

$$h'(t) = \ln t - \frac{1}{M+2} \cdot \ln^2 t - \frac{2}{M+2} \cdot \ln t = \left(1 - \frac{(\ln t) + 2}{M+2}\right) \ln t.$$

Note that $1 - \frac{(\ln t)+2}{M+2} \geq 0$ because $\ln t \leq M$. Therefore $h'(t) \geq 0$ when $t \geq 1$ and $h'(t) \leq 0$ when $t \leq 1$, as desired. ◀

5 Application 1: Selection Problem

5.1 Bi-correlated samples

To establish a relationship between correlated samples and selection, we first define an intermediate problem. The *bi-correlated samples problem* is defined by (here $\mathcal{D}_{ab} \coloneqq \mathcal{D}_a \times \mathcal{D}_b$):

$$\begin{aligned} \mathsf{biCorr}_\epsilon(f, \mathcal{D}) &\coloneqq \min\nolimits_{k \geq 1} \mathsf{D}_\epsilon(f^{2k}, \tfrac{1}{2}\mathcal{D}_{01}^k + \tfrac{1}{2}\mathcal{D}_{10}^k), \\ \mathsf{biCorr}_\epsilon(f) &\coloneqq \max\nolimits_{\mathcal{D}} \mathsf{biCorr}_\epsilon(f, \mathcal{D}). \end{aligned}$$

That is, the task is to decide whether f^{2k} outputs $(01)^k$ or $(10)^k$ as $k \to \infty$. We show this is as hard as correlated samples:

▶ **Lemma 5.1.** $\mathsf{Corr}(f, \mathcal{D}) = \Theta(\mathsf{biCorr}(f, \mathcal{D}))$.

Proof. It is obvious that $\mathsf{biCorr}(f, \mathcal{D}) \leq \mathsf{Corr}(f, \mathcal{D})$, so we focus on the converse, $\mathsf{Corr}(f, \mathcal{D}) \leq O(\mathsf{biCorr}(f, \mathcal{D}))$. The proof is via a hybrid argument. Let $T \colon (\{0,1\}^n)^{2k} \to \{0,1\}$ be an optimal algorithm for $\mathsf{biCorr}_{1/3}(f, \mathcal{D})$ that uses k sample pairs. Letting $d(-,-)$ denote the statistical distance between two distributions, the fact that T achieves error $\epsilon \coloneqq 1/3$ can be written as

$$d(T(\mathcal{D}_{01}^k), T(\mathcal{D}_{10}^k)) \;\geq\; 1 - 2\epsilon.$$

By the triangle inequality,

$$d(T(\mathcal{D}_{01}^k), T(\mathcal{D}_{00}^k)) \;+\; d(T(\mathcal{D}_{00}^k), T(\mathcal{D}_{10}^k)) \;\geq\; 1 - 2\epsilon.$$

Either the first or the second term is $\geq (1-2\epsilon)/2$. Say the first (second case is similar):

$$d(T(\mathcal{D}_{01}^k), T(\mathcal{D}_{00}^k)) \;\geq\; (1-2\epsilon)/2 \;=\; 1 - 2\epsilon' \qquad \text{where } \epsilon' \coloneqq 1/4 + \epsilon/2 = 5/12.$$

This means we can turn T into an 5/12-error algorithm for the correlated k-samples problem: the odd numbered input samples of T the algorithm can generate from $\mathcal{D}_0$ on its own; the even numbered input samples of T are taken from the input to the correlated k-samples problem. Finally, the error can be reduced to 1/3 via Fact 1.1. ◀

5.2 Proof of Theorem 1.3

First item. The following claim together with Lemma 5.1 implies the first item.

▷ Claim 5.2. $\mathsf{biCorr}_\epsilon(f,\mathcal{D}) \leq \mathsf{Sel}_\epsilon(f,\mathcal{D})$.

Proof. Let T_{Sel} be an optimal algorithm for $\mathsf{Sel}_\epsilon(f,\mathcal{D})$ using k samples. We describe an algorithm T_{biCorr} for bi-correlated k-samples with the same error and query cost. Let $x = (x_{ij})$ for $(i,j) \in [k] \times [2]$ be the random input to T_{biCorr}, that is, either *(i)* $x \sim \mathcal{D}_{01}^k$ or *(ii)* $x \sim \mathcal{D}_{10}^k$. The algorithm T_{biCorr} chooses a random string $z \in [2]^k$ and runs T_{Sel} on input $y \coloneqq (x_{iz_i})_{i\in[k]}$. Note that y is distributed as $\mathcal{D}^k$ in both cases *(i)* and *(ii)*. Suppose T_{Sel} outputs some $(i, f(x_{iz_i}))$. Assuming this output is correct for selection, and remembering our choice of z_i, we can deduce which case, *(i)* or *(ii)*, the input x came from, and let T_{biCorr} guess accordingly. Hence algorithm T_{biCorr} is correct every time T_{Sel} is, and so the error parameter is unaffected. ◁

Second and third item. For separating correlated samples from selection, we again consider the n-bit $\textsc{Xor}_n$ function. Define $x \sim \mathcal{D}$ by the following process:

1. Sample z uniformly from $\{0,1\}^{n-2}$ and let $a \coloneqq \textsc{Xor}_{n-2}(z)$.
2. Sample b uniformly from $\{0,1\}$.
3. With probability $\epsilon \coloneqq 1\%$, output $x \coloneqq aaz$; with probability $1-\epsilon = 99\%$, output $x \coloneqq bbz$.

Note that the first two bits of $x \sim \mathcal{D}$ are identical and hence $\textsc{Xor}_n(x) = \textsc{Xor}_{n-2}(z)$. Moreover, the first bit is ϵ-correlated with the function value $\textsc{Xor}_n(x)$. This makes $(\textsc{Xor}_n, \mathcal{D})$ easy for the correlated samples problem: The 1-query algorithm that guesses the function value based on the first bit of the first sample has error $\leq 1/2 - \epsilon/2$, and this error can be reduced to $1/3$ via Fact 1.1. This shows that $\mathsf{Corr}(\textsc{Xor}_n, \mathcal{D}) = O(1)$.

Next we prove the lower bound $\mathsf{Sel}(\textsc{Xor}_n, \mathcal{D}) = \Omega(n)$, which also proves the third item. Suppose for contradiction that T is a height-$(n-3)$ deterministic decision tree for k-selection for $(\textsc{Xor}_n, \mathcal{D})$. Consider any leaf ℓ that claims the i-th sample evaluates to $b \in \{0,1\}$. If we condition $\mathcal{D}^k$ by the $\leq n-3$ queries made by ℓ, we note that the function value is still only slightly biased away from $1/2$, that is, $\mathbb{E}_{x\sim\mathcal{D}^k|\ell}[\textsc{Xor}_n(x_i)] \in 1/2 \pm \epsilon$. Hence no leaf of T can compute selection to within error $\leq 1/3$. This concludes the proof of Theorem 1.3.

6 Application 2: Randomized Composition

Goal. In this section we prove Theorem 1.6, namely $\mathsf{R}(f \circ g) \geq \Omega(\mathsf{fbs}(f)\mathsf{R}(g))$. By Theorem 1.2 and Lemma 5.1 (from Subsection 5.1) it suffices to show

$$\mathsf{biCorr}(g) \;\leq\; O(\mathsf{R}(f\circ g)/\mathsf{fbs}(f)).$$

Let T be an optimal $1/10$-error algorithm for $f \circ g$ making $q \coloneqq O(\mathsf{R}(f\circ g))$ queries. Our goal is, given any balanced input distribution $\mathcal{D} \coloneqq \frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$ to the inner function g, to build a bounded-error algorithm T' solving the bi-correlated samples problem for $(g,\mathcal{D})$.

Rarely queried block. By the definition of $\mathsf{fbs}(f)$, there is an input $y \in \{0,1\}^n$ to f (say, $f(y) = 0$) with sensitive blocks $B_1, \cdots, B_N \subseteq [n]$ and weights $w_1, \cdots, w_N \in [0,1]$ such that

$$\textstyle\sum_{j\in[N]} w_j \;=\; \mathsf{fbs}(f), \tag{4}$$
$$\textstyle\sum_{j:B_j\ni i} w_j \;\leq\; 1, \qquad \forall i \in [n]. \tag{5}$$

For any $z \in \{0,1\}^n$, define $\mathcal{D}_z$ as the distribution over $(x_1, \cdots, x_n) \in (\{0,1\}^m)^n$ where each x_i is drawn independently from $\mathcal{D}_{z_i}$. Hence we have $g^n(x) = z$ for $x \sim \mathcal{D}_z$. We define

$$q_j \;\coloneqq\; \text{expected \# of queries } T \text{ makes to block } B_j \text{ on input } \mathcal{D}_y.$$

That is, if we denote by $i_t \in [n]$ the block that T queries at time t, then q_j is the expected number of time steps t with $i_t \in B_j$. By linearity of expectation and (5), we have

$$\textstyle\sum_{j\in[N]} w_j q_j \;=\; \mathbb{E}\Big[\sum_{j\in[N]} w_j \sum_{t:i_t\in B_j} 1\Big] \;=\; \mathbb{E}\Big[\sum_{t\in[q]} \sum_{j:B_j\ni i_t} w_j\Big] \;\le\; \mathbb{E}\Big[\sum_{t\in[q]} 1\Big] \;\le\; q.$$

Combining this with (4), we know there exists $j \in [N]$, say $j = 1$ for simplicity, such that

$$q_1 \;\le\; \frac{q}{\mathsf{fbs}(f)}.$$

Truncated T. Next we modify T so that it makes at most $5q_1$ queries to block B_1 for *every* input (not just on average over $\mathcal{D}_y$). Namely, if T makes more than $5q_1$ queries to block B_1, we simply let T halt and output 1; otherwise its behavior is unchanged. We denote this "truncated" algorithm by T^{tr}. We claim that T^{tr} still computes $f \circ g$ correctly on average over both $\mathcal{D}_y$ and $\mathcal{D}_{y^{B_1}}$ (recall that y^{B_1} is y but with the block B_1 flipped; note that $f(y^{B_1}) = 1$ and hence $(f \circ g)(x) = 1$ for each $x \sim \mathcal{D}_{y^{B_1}}$)

$$\begin{aligned}
&\textit{Correct for } x \sim \mathcal{D}_{y^{B_1}}\textit{:} \qquad & \Pr[T^{\mathrm{tr}}(x) = 1] \;&\ge\; \Pr[T(x) = 1] \\
& & &\ge\; 4/5. && (6)
\end{aligned}$$

$$\begin{aligned}
&\textit{Correct for } x \sim \mathcal{D}_y\textit{:} \qquad & \Pr[T^{\mathrm{tr}}(x) = 0] \;&\ge\; \Pr[T(x) = 0] \\
& & &\quad - \Pr[T(x) \text{ makes} > 5q_1 \text{ queries to } B_1] && (7)\\
& & &\ge\; 4/5 - 1/5 && (8)\\
& & &=\; 3/5, && (9)
\end{aligned}$$

where (7) uses the Union Bound and (8) uses the Markov Bound.

Algorithm T'. We are ready to define the algorithm T' for the bi-correlated samples problem for $(g, \mathcal{D})$. The random input to this problem is $z = (z_{ij})$, $(i,j) \in [n] \times \{0,1\}$, sampled either from *(i)* $\mathcal{D}_{01}^n$ or *(ii)* $\mathcal{D}_{10}^n$. On input z the algorithm T' simply runs T^{tr} on the input $(x_1, \ldots, x_n) \in (\{0,1\}^m)^n$ defined by

$$x_i \;\coloneqq\; \begin{cases} z_{iy_i} & \text{for } i \in B_1, \\ \sim \mathcal{D}_{y_i} & \text{for } i \notin B_1. \end{cases}$$

That is, for $i \in B_1$ the algorithm T' simply copies its input bits in z to the bits of x. For $i \notin B_1$ the algorithm T' uses its own randomness to generate an independent sample from either $\mathcal{D}_0$ or $\mathcal{D}_1$. The key observation is that in case *(i)* we have $x \sim \mathcal{D}_y$, and in case *(ii)* we have $x \sim \mathcal{D}_{y^{B_1}}$. But T^{tr} can distinguish these two cases to within bounded error by (6) and (9). Hence T' is a bounded-error algorithm for bi-correlated samples with query cost $5q_1 \le O(q/\mathsf{fbs}(f))$. This completes the proof of Theorem 1.6.

References

1 Scott Aaronson. Quantum certificate complexity. *Journal of Computer and System Sciences*, 74(3):313–322, 2008. `doi:10.1016/j.jcss.2007.06.020`.

2 Andris Ambainis, Martins Kokainis, Krišjānis Prūsis, and Jevgēnijs Vihrovs. All classical adversary methods are equivalent for total functions. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 96, pages 8:1–8:14, 2018. `doi:10.4230/LIPIcs.STACS.2018.8`.

3 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity. In *Proceedings of the 37th Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 93, pages 10:1–10:13, 2018. `doi:10.4230/LIPIcs.FSTTCS.2017.10`.

4 Amos Beimel, Sebastian Ben Daniel, Eyal Kushilevitz, and Enav Weinreb. Choosing, agreeing, and eliminating in communication complexity. *Computational Complexity*, 23:1–42, 2014. `doi:10.1007/s00037-013-0075-7`.

5 Shalev Ben-David and Eric Blais. A new minimax theorem for randomized algorithms. *arXiv preprint*, 2020. `arXiv:2002.10802`.

6 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions. *arXiv preprint*, 2020. `arXiv:2002.10809`.

7 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55, pages 60:1–60:14, 2016. `doi:10.4230/LIPIcs.ICALP.2016.60`.

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

9 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity via max-conflict complexity. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 132, pages 64:1–64:13, 2019. `doi:10.4230/LIPIcs.ICALP.2019.64`.

10 Justin Gilmer, Michael Saks, and Srikanth Srinivasan. Composition limits and separating examples for some boolean function complexity measures. *Combinatorica*, 36(3):265–311, 2016. `doi:10.1007/s00493-014-3189-x`.

11 Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication versus partition number. *ACM Transactions on Computation Theory*, 10(1), 2018. `doi:10.1145/3170711`.

12 Peter Høyer, Troy Lee, and Robert Špalek. Negative weights make adversaries stronger. In *Proceedings of the 39th Symposium on Theory of Computing (STOC)*, STOC '07, page 526–535, 2007. `doi:10.1145/1250790.1250867`.

13 Raghav Kulkarni and Avishay Tal. On fractional block sensitivity. *Chicago Journal of Theoretical Computer Science*, 2016(8), 2016. `doi:10.4086/cjtcs.2016.008`.

14 Ashley Montanaro. A composition theorem for decision tree complexity. *Chicago Journal of Theoretical Computer Science*, 2014(6), 2014. `doi:10.4086/cjtcs.2014.006`.

15 Noam Nisan. CREW PRAMs and decision trees. *SIAM Journal on Computing*, 20(6):999–1007, 1991. `doi:10.1137/0220062`.

16 Ben Reichardt. Reflections for quantum query algorithms. In *Proceedings of the 22nd Symposium on Discrete Algorithms (SODA)*, pages 560–569, 2011.

17 Petr Savický. On determinism versus unambiquous nondeterminism for decision trees. Technical Report TR02-009, Electronic Colloquium on Computational Complexity (ECCC), 2002. URL: `http://eccc.hpi-web.de/report/2002/009/`.

18 Ronen Shaltiel. Towards proving strong direct product theorems. *Computational Complexity*, 12(1/2):1–22, 2004. `doi:10.1007/s00037-003-0175-x`.

19 Avishay Tal. Properties and applications of boolean function composition. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science (ITCS)*, pages 441–454, 2013. doi:10.1145/2422436.2422485.

20 Andrew Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Symposium on Foundations of Computer Science (SFCS 1977)*, pages 222–227, October 1977. doi:10.1109/SFCS.1977.24.