Property Testing of Boolean Functions

Jinyu Xie

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2018

ABSTRACT

Property Testing of Boolean Functions

Jinyu Xie

The field of property testing has been studied for decades, and Boolean functions are among the most classical subjects to study in this area. In this thesis we consider the property testing of Boolean functions: distinguishing whether an unknown Boolean function has some certain property (or equivalently, belongs to a certain class of functions), or is far from having this property, where a distance parameter $\epsilon \in (0, 1)$ is used to set the threshold for the latter statement. We study this problem under both the standard setting, where the distance between functions is measured with respect to the uniform distribution, as well as the distribution-free setting, where the distance is measured with respect to a fixed but unknown distribution. We obtain both new upper bounds and lower bounds for the query complexity of testing various properties of Boolean functions:

- Under the standard model of property testing, we prove a lower bound of $\tilde{\Omega}(n^{1/3})$ for the query complexity of any adaptive algorithm that tests whether an n-variable Boolean function is monotone, improving the previous best lower bound of $\tilde{\Omega}(n^{1/4})$ by Belov and Blais in [BB16]. We also prove a lower bound of $\tilde{\Omega}(n^{2/3})$ for adaptive algorithms, and a lower bound of $\tilde{\Omega}(n)$ for non-adaptive algorithms with one-sided errors that test unateness, a natural generalization of monotonicity. The latter lower bound matches the previous upper bound proved by Chakrabarty and Seshadhri [CS16], up to poly-logarithmic factors of n.
- We also study the distribution-free testing of k-juntas, where a function is a k-junta if it depends on at most k out of its n input variables. The standard property testing of k-juntas under the uniform distribution has been well understood: it has been shown that, for adaptive testing of k-juntas (and some constant ϵ) the optimal query complexity is $\tilde{\Theta}(k)$ [Bla09; CG04]; and for non-adaptive testing of k-juntas

it is $\tilde{\Theta}(k^{3/2})$ [Bla08; Che+17]. Both bounds are tight up to poly-logarithmic factors of k. However, this problem is far from clear under the more general setting of distribution-free testing. Previous results only imply an $O(2^k/\epsilon)$-query algorithm for distribution-free testing of k-juntas, and besides lower bounds under the uniform distribution setting that naturally extend to this more general setting, no other results were known from the lower bound side. We significantly improve these results with an $\tilde{O}(k^2/\epsilon)$-query adaptive distribution-free tester for k-juntas, as well as an exponential lower bound of $\Omega(2^{k/3})$ for the query complexity of non-adaptive distribution-free testers for this problem. These results illustrate the hardness of distribution-free testing and also the significant role of adaptivity under this setting.

- In the end we also study distribution-free testing of other basic Boolean functions. Under the distribution-free setting, a lower bound of $\tilde{\Omega}(n^{1/5})$ was proved for testing of (general) conjunctions, decision lists, and linear threshold functions by Glasner and Servedio in [GS09], and an $O(\sqrt{n}/\epsilon)$-query algorithm for testing monotone conjunctions was shown by Dolev and Ron in [DR11]. Building on techniques developed in [GS09] and [DR11], we improve these lower bounds to $\tilde{\Omega}(n^{1/3})$, and specifically for the class of conjunctions we present an adaptive algorithm with query complexity $\tilde{O}(n^{1/3}/\epsilon^5)$. For ϵ being the constant hidden in our lower bound proof, our lower and upper bounds are tight for testing conjunctions, up to poly-logarithmic factors of n.

Contents

List of Figures

Acknowledgements

First I want to thank my advisor, Xi Chen. It has been such a pleasure to be his student and to work with him. Thanks for admitting me to the Columbia University: none of the amazing things during the past five years would have happened without him. Thanks for the countless conversations and discussions about research as well as everyday life. He has been always so brilliant and supportive in guiding my research. I have become a much better researcher as well as a better person because of him.

Many thanks to my committee members, Rocco Servedio, Omri Weinstein, Eric Blais and Li-Yang Tan, for their time and valuable comments.

I also want to thank all the other people I have been working with during my graduate study: Erik Waingarten, Ying Sheng, Zhengyang Liu, Yu Cheng, Bo Tang, Xiaorui Sun, Anthi Orfanou, Yi-Hsiu Chen, and Arka Bhattacharya. Together we have completed a lot of nice work, which led to the results in this thesis. We also met some unsettled puzzles, and I'm sure we will figure them out one day.

Beyond these excellent researchers, I also met some amazing friends in New York City: Dingzeyu Li, Zhengqian Cheng, Kuangya Zhai, Xiaotang Wang, and many more. Thanks for all the fun we had together.

I thank my advisors during undergraduate study, Andrew Yao and Giulio Chiribella, who first introduced me to the exciting research field of Theoretical Computer Science.

I thank Yong Rui and Randy Carnevale, who were my supervisors during my internships at Microsoft and Transfix. They helped a lot in getting me connected to the industrial world.

Finally, and above all, I want to thank my parents, Jiaguo and Sihua, and my wife, Yi, for all the unconditional love and support they gave to me during the past years. I can make it all because of you. I love you.

Chapter 1

Introduction

Property testing of Boolean functions was first introduced by Blum, Luby, and Rubinfeld in [BLR93] and Rubinfeld and Sudan in [RS96]. It studies the problem of distinguishing whether an unknown Boolean function satisfies a certain property (or equivalently, belongs to a certain class of functions), or is far from having this property. More precisely, given an unknown Boolean function $f : \{0,1\}^n \to \{0,1\}$ and a class $\mathfrak{C}$ of functions over the same domain (e.g., the class of functions that are monotone, which means that flipping a variable from 0 to 1 can never change the output of the function from 1 to 0), we want to know if f belongs to $\mathfrak{C}$ or f is ϵ-far from $\mathfrak{C}$ for some constant $\epsilon \in (0,1)$. By ϵ-far we mean the distance between f and any function $g \in \mathfrak{C}$, $dist(f,g) := Pr_{\mathbf{x}}[f(\mathbf{x}) \neq g(\mathbf{x})]$, is always at least ϵ, with the probability taken over the uniform draw of $\mathbf{x}$ over $\{0,1\}^n$[1]. We are allowed to make queries to the black-box oracle of f: upon each query string $x \in \{0,1\}^n$, the oracle returns the value of $f(x)$ to us. The goal is to make as few queries as possible to distinguish the two cases above.

Under this setting, there has been long lines of works studying scenarios with respect to different choices of class $\mathfrak{C}$: [Dod+99; Gol+00; Fis+02; Fis04; BKR04; AC06; Ail+07; HK08; RS09; BBM12; Bri+12; Ron+12; CS13a; CS13b; CS14; BRY13; CST14; KMS15; Che+15; BB16] for monotonicity, [Gol+00; KS16; CS16; Bal+16; Bal+17] for unateness, [Fis+04; Bla08; Bla09; CG04; Buh+13; STW15] for k-junta, [Mat+09; Mat+10; BBM12] for threshold functions, and many more. There are still gaps remaining between best upper

[1] In some high-level discussions, we may simply say f is far from $\mathfrak{C}$, when there exists some unspecified constant ϵ such that $dist(f,g) \geq \epsilon$ for all function $g \in \mathfrak{C}$.

bounds and lower bounds (in terms of query complexity) for many of these problems. We will review previous results and our contribution for testing of monotonicity, unateness, and k-juntas in Sections 1.2 and 1.3.

A more general setting for this problem, called distribution-free testing, was introduced by Goldreich, Goldwasser, and Ron in 1998 [GGR98]. Instead of measuring distance between functions with respect to the uniform distribution over $\{0,1\}^n$, this new setting assumes there is a fixed but unknown distribution $\mathcal{D}$ over $\{0,1\}^n$ and the distance is calculated with respect to $\mathcal{D}$: $dist_{\mathcal{D}}(f,g) := Pr_{\mathbf{x}\sim\mathcal{D}}[f(\mathbf{x}) \neq g(\mathbf{x})]$. In order to gather the missing information regarding the distribution $\mathcal{D}$, we are also given the sampling oracle access to $\mathcal{D}$: each time the sampling oracle is triggered (with no input), it samples $\mathbf{x}$ from $\mathcal{D}$ and returns $\mathbf{x}$ as response. We call this type of queries as sampling queries, and queries to the black-box oralce as black-box queries. Fix a target class $\mathfrak{C}$, and given an unknown function f and an unknown distribution $\mathcal{D}$, we are interested in the total number of both black-box queries to the oracle of f as well as sampling queries to the oracle of $\mathcal{D}$ that one needs to complete the task: distinguishing whether f belongs to $\mathfrak{C}$, or it is far from $\mathfrak{C}$ with respect to $\mathcal{D}$. This model is clearly more general compared to the standard model, since we can always fix the distribution $\mathcal{D}$ to be the uniform distribution over $\{0,1\}^n$.

This generalized model of distribution-free testing has been studied in [HK05; AC06; HK07; GS09; HK04; HK08; DR11], and it was also inspired by similar models in computational learning theory like PAC learning model [Val84]. It has been shown in [GGR98] that any proper PAC learning algorithm can be used for constructing distribution-free testing algorithms for the same class. We will discuss more about previous results and our works for distribution-free testing in Sections 1.3 and 1.4.

Before going into more detailed introductions and proofs, let's first provide some preparation in the next section.

1.1 Preliminary

We introduce some formal definitions and notation in this section.

Throughout this thesis, we will restrict our attention to n-variable Boolean functions that map $\{0,1\}^n$ to $\{0,1\}$ for some positive integer n, unless otherwise specified. For clarity of this thesis we will hide poly-logarithmic factors of n (and k for the sections about testing k-juntas) as long as it won't interfere with our discussion.

For two Boolean functions $f, g : \{0,1\}^n \to \{0,1\}$ and a probability distribution $\mathcal{D}$ over $\{0,1\}^n$, the distance between f and g with respect to distribution $\mathcal{D}$ is defined as:

$$dist_{\mathcal{D}}(f,g) := \Pr_{\mathbf{z}\sim\mathcal{D}}[f(\mathbf{z}) \neq g(\mathbf{z})].$$

For a class $\mathfrak{C}$ of Boolean functions over $\{0,1\}^n$, we further define the distance between f and $\mathfrak{C}$ with respect to distribution $\mathcal{D}$ as:

$$dist_{\mathcal{D}}(f,\mathfrak{C}) := \min_{g\in\mathfrak{C}}\{dist_{\mathcal{D}}(f,g)\}.$$

Fix some constant $\epsilon \in (0,1)$, we say f is ϵ-far from $\mathfrak{C}$ with respect to $\mathcal{D}$ if $dist_{\mathcal{D}}(f,\mathfrak{C}) \geq \epsilon$. Under the standard setting of property testing when we don't specify the distribution $\mathcal{D}$, the distances are measured with respect to the uniform distribution by default.

Given the definition of distance, we can then define algorithms for testing certain Boolean function properties (usually called testers) as follows:

Fix a class $\mathfrak{C}$ of Boolean function and for some constant $\epsilon \in (0,1)$, we define a standard ϵ-tester for $\mathfrak{C}$ as:

Definition 1.1.1. *A randomized[2] algorithm A is an ϵ-tester for $\mathfrak{C}$, if given black-box oracle access to an unknown Boolean function f:*

[2]In this thesis we will always assume an algorithm is randomized, unless we state it is deterministic.

- *it accepts f with probability at least 2/3 if $f \in \mathfrak{C}$;*
- *it rejects f with probability at least 2/3 if f is ϵ-far from $\mathfrak{C}$.*

We say a tester has one-sided errors if it always accepts f when $f \in \mathfrak{C}$. It is called adaptive if the latter queries can depend on responses from previous queries, while on the other hand, a tester is non-adaptive if it has to pre-select a set of queries, submit them to the oracle at the same time, and make a decision (accept or reject) based on responses it collects from the oracle. We also say an algorithm A ϵ-tests the given class $\mathfrak{C}$ if it is an ϵ-tester for $\mathfrak{C}$, and we are interested in the query complexity of A, which is defined as how many queries A needs to make to the black-box oracle of f.

In the distribution-free setting, we define a distribution-free ϵ-tester for $\mathfrak{C}$ as:

Definition 1.1.2. *A randomized algorithm B is a distribution-free ϵ-tester for $\mathfrak{C}$, if given black-box oracle access to an unknown Boolean function f and sampling oracle access to an unknown distribution $\mathcal{D}$ over $\{0,1\}^n$:*

- *it accepts f with probability at least 2/3 if $f \in \mathfrak{C}$;*
- *it rejects f with probability at least 2/3 if f is ϵ-far from $\mathfrak{C}$ with respect to $\mathcal{D}$.*

Similarly, we say a distribution-free tester has one-sided errors if it always accepts f when $f \in \mathfrak{C}$. We also say the algorithm B above ϵ-tests the given class under distribution-free setting. The query complexity of B is usually defined as the total number of queries it makes to either the black-box oracle of f or the sampling oracle of $\mathcal{D}$, while for convenience of our arguments we in fact will use an alternate definition: the maximum between number of black-box queries used and number of sampling queries used, i.e., B is a q-query distribution-free tester if it makes at most q black-box queries and also at most q sampling queries. We may also assume that, upon each sampling query, the oracle of $\mathcal{D}$ returns $\mathbf{x} \sim \mathcal{D}$ as well as the value of $f(x)$. It's easy to see that both the new definition and new assumption won't change any asymptotic upper (or lower) bound we get.

Note that there is in fact no input from the algorithm side for queries to the sampling oracles. Upon each input pair $(f, \mathcal{D})$, one may assume without loss of generality that a distribution-free tester consists of two phases: in the first phase, the algorithm draws a certain number of sample pairs $(\mathbf{x}, f(\mathbf{x}))$ with $\mathbf{x} \sim \mathcal{D}$; in the second phase, it makes black-box queries to oracle of f. In general, a query $x \in \{0,1\}^n$ made by the algorithm in the second phase may depend on sample pairs it receives in the first phase (e.g. it can choose to query a string that is close to a sample string received in the first phase) and results of queries to f made so far. Later in Chapter 3 we will prove lower bounds on non-adaptive distribution-free testers. A distribution-free tester is said to be non-adaptive if its black-box queries made in the second phase do not depend on results of previous black-box queries, i.e., all queries during the second phase can be made in a single batch (though we emphasize that they may depend on sample pairs the algorithm received in the first phase).

We end this section with some basic notation that we will use throughout this thesis.

We will use lowercase letters like x, y to denote input strings of the functions, and uppercase letters like A, B to denote sets (of indices, strings, etc.). A special set we will use is $[n] = \{1, 2, \ldots, n\}$, which is usually referred to the set of all n indices for input (n-bit strings) of a Boolean function. There is a natural bi-jection between the ith input variable and the index i for each $i \in [n]$, therefore we may sometimes abuse the notation and also say a variable x_i is in $A \subset [n]$ when $i \in A$, for convenience of the proofs. There are also two special strings we will be using: 1^n and 0^n, which correspond to the all-1 and all-0 string in $\{0,1\}^n$.

For any set $A \subset [n]$, we write $\overline{A}$ to denote the complement of A with respect to $[n]$, and write $\{0,1\}^A$ as the set of all Boolean strings of length $|A|$ that are indexed by indices in A. For any strings $x, y \in \{0,1\}^n$, we use $x_A \in \{0,1\}^A$ to denote the projection of x over indices in A, and $x_A \circ y_{\overline{A}} \in \{0,1\}^n$ to denote the string concatenated from x_A and

$y_{\overline{A}}$. We also write $x^{(A)} \in \{0,1\}^n$ as the string obtained from x after negating all variables in A. For the case when $A = \{i\}$ for some $i \in [n]$, we just write $x_A = x_i$ (the ith bit of x) and $x^{(A)} = x^{(i)}$ for simplicity.

For two strings $x, y \in \{0,1\}^n$ that differ on exactly one index i (i.e. $y = x^{(i)}$), we say (x, y) is an edge on the Boolean hypercube over $\{0,1\}^n$, and define i as its direction. We also say this edge is a bi-chromatic edge of function f when $f(x) \neq f(y)$.

We write $|x|$ as the Hamming weight of a string $x \in \{0,1\}$, namely the number of 1's in x. For two strings $x, y \in \{0,1\}^n$, we write $d(x, y)$ as the Hamming distance between them, which is the number of indices that they differ on, and we use $\mathsf{diff}(x, y) \subset [n]$ to denote the set of such indices. We also write $x \vee y$ as the bitwise OR, $x \wedge y$ as the bitwise AND, and $x \oplus y$ as the bitwise EXCLUSIVE-OR of two strings x and y. We also extend such notation to be used between a string $x \in \{0,1\}$ and a partial string $z \in \{0,1\}^A$ for some $A \subset [n[$: for example, $x \oplus z$ is used to denote the n-bit string x' with $x'_i = x_i$ for all $i \notin A$ and $x'_i = x_i \oplus z_i$ for all $i \in A$.

We use boldface letters like $\mathbf{x}$ to denote random variables and calligraphic letters like $\mathcal{D}$ to denote a distribution. We write $\mathbf{u} \sim \mathcal{D}$ to indicate that the random variable $\mathbf{u}$ is drawn from the distribution $\mathcal{D}$. For any finite universe A (for example, $\{0,1\}^n$), we also use $\mathbf{u} \sim A$ to indicate that the random variable $\mathbf{u}$ is uniformly drawn from A . Given two distributions $\mathcal{P}, \mathcal{Q}$ over A, we define the total variation distance between $\mathcal{P}$ and $\mathcal{Q}$ as: $d_{TV}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2} \sum_{a \in A} |\mathcal{P}(a) - \mathcal{Q}(a)|$.

Now we are ready to dive into our discussion about testing for various Boolean function classes.

1.2 Testing of monotonicity and unateness

We start with the classical problem of monotonicity testing. The definition of monotonicity is given as follows:

Definition 1.2.1. *For any two strings $x, y \in \{0,1\}^n$, we write $x \prec y$ if and only if $x_i \leq y_i$ for all $i \in [n]$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is monotone, if and only if for any two strings $x, y \in \{0,1\}^n$ with $x \prec y$, we have $f(x) \leq f(y)$.*

Fix some function f, an obvious violation of f to monotonicity is a pair of two strings $x, y \in \{0,1\}^n$ such that either $x \prec y$ and $f(x) > f(y)$, or $x \succ y$ and $f(x) < f(y)$. We call (x, y) as a violation pair in this case, and further define it as a violation edge when (x, y) is an edge.

Property testing of monotonicity (or equivalently, testing of the class of monotone functions) studies the problem of distinguishing whether an unknown Boolean function is monotone or is far from monotone. It was first considered by Goldreich et al. in [Gol+00], where they proposed a simple "edge tester" for this problem. The algorithm keeps repeating the following test for $O(n/\epsilon)$ times: uniformly samples $\mathbf{x} \sim \{0,1\}^n$ and $\mathbf{i} \sim [n]$ at random, and rejects the input function f if $(\mathbf{x}, \mathbf{x}^{(\mathbf{i})})$ is a violation edge of f. If no violation is found the algorithm accepts the input in the end. Clearly this algorithm only has one-sided errors, and it was shown that an input function that is ϵ-far from being monotone gets rejected by this algorithm with high probability.

This was the best known algorithm for more than a decade, until Chakrabarty and Seshadhri proposed a more efficient algorithm with query complexity $\tilde{O}(n^{7/8}/\epsilon^{3/2})$ [CS13a]. Instead of looking for violation edges this algorithm looked for violation pairs, and by proving a new structural lemma regarding Boolean functions they were able to show that $\tilde{O}(n^{7/8}/\epsilon^{3/2})$ queries are enough to reject the input with high probability when the input function is far from monotonicity. This strategy was then improved to obtain an $\tilde{O}(n^{5/6}/\epsilon^4)$-query algorithm by Chen et al. in [CST14] by introducing a more careful way of picking the pairs.

This upper bound was further improved with an $\tilde{O}(\sqrt{n}/\epsilon^2)$-query algorithm by Khot et al. in [KMS15], which is the most efficient algorithm for testing monotonicity known to date. They continued to follow the "pair tester" strategy, but managed to show fewer

queries suffice by introducing a new isoperimetric-type theorem for far-from-monotone Boolean functions. Combined with the lower bound from [CST14] that will be discussed in the next part, this result (almost) solved the optimal query complexity of non-adaptive testing of monotonicity.

The algorithms mentioned above are all non-adaptive and have one-sided errors, since they only reject the input when finding a violation to monotonicity, and their current queries doesn't depend on responses from previous queries.

On the lower bound side of testing monotonicity, Fisher et al. proved the first lower bound of $\Omega(\log n)$ for non-adaptive algorithms, and a lower bound of $\Omega(\sqrt{n})$ when the algorithm is also restricted to have one-sided errors, for some constant distance parameter $\epsilon \in (0, 1)$ [Fis+02]. Chen et al. then improved the lower bound for general non-adaptive algorithms to $\tilde{\Omega}(n^{1/5})$ in [CST14] and then to $\tilde{\Omega}(n^{1/2-c})$ in [Che+15] for any positive constant c.

All these lower bounds are for non-adaptive algorithms, and only yield a lower bound of $\tilde{\Omega}(\log n)$ for adaptive algorithms. A recent breakthrough was made by Belovs and Blais [BB16], where they exploited properties of Talagrand functions and proved a lower bound of $\tilde{\Omega}(n^{1/4})$ for adaptive monotonicity testing.

For the more general setting of distribution-free testing, Halevy and Kushilevitz showed in [HK05] an exponential lower bound for monotonicity testing under this setting, which illustrates the hardness of distribution-free testing.

In Chapter 2 we will follow the work of [BB16] and present our new lower bound of testing monotonicity adaptively [CWX17a], which is formally stated as follows:

Theorem 1.2.2. *There exists some constant $\epsilon_0 \in (0, 1)$ such that any adaptive algorithm that ϵ_0-tests monotonicity must make at least $\tilde{\Omega}(n^{1/3})$ queries.*

We also want to mention that the techniques developed in the proof of Theorem 1.2.2 can be easily adapted to prove a tight $\tilde{\Omega}(n^{1/2})$ lower bound for non-adaptive monotonicity

testing, which removes the $-c$ in the exponent of [Che+15] and completely solves the optimal query complexity of this problem for constant ϵ, if we ignore the poly-logarithmic factors. We omit the details of this proof due to its similarity.

One generalized property from monotonicity is called unateness. A function f is unate if for every index $i \in [n]$ f is either non-decreasing on the ith direction (flipping the variable of input on index i from 0 to 1 won't flip the output from 1 to 0) or non-increasing on the ith direction, or in other words:

Definition 1.2.3. *A Boolean function f is unate if and only if there exists some vector $r \in \{0,1\}^n$ such that $g(x) = f(x \oplus r)$ is monotone.*

We say an edge $(x, x^{(i)})$ is monotone, if $x_i = 0$ and $f(x)$ goes from 0 to 1 after flipping x_i, or $x_i = 1$ and $f(x)$ goes from 1 to 0 after flipping x_i; an edge $(x, x^{(i)})$ is anti-monotone, if $x_i = 0$ and $f(x)$ goes from 1 to 0 after flipping x_i, or $x_i = 1$ and $f(x)$ goes from 0 to 1 after flipping x_i. Then we know a typical violation to unateness consists of a pair of edges $(x, x^{(i)})$ and $(y, y^{(i)})$ on the same ith direction for some $i \in [n]$, while $(x, x^{(i)})$ is monotone and $(y, y^{(i)})$ is anti-monotone.

The testing of unateness studies the problem of distinguishing whether an unknown Boolean function is unate or is far from being unate. It was also first considered in [Gol+00], where they proposed an $O(n^{3/2}/\epsilon)$-query algorithm for this task. The algorithm works in a similar fashion as the edge tester for monotonicity and only rejects the input when it finds a violation to unateness. Therefore this algorithm is non-adaptive and has one-sided errors.

Later, by introducing adaptiveness into the algorithm Khot and Shinkar [KS16] were able to design an $\tilde{O}(n/\epsilon)$-query tester for unateness, and this work was then improve by Chakrabarty and Seshadhri [CS16] with a new non-adaptive $\tilde{O}(n/\epsilon)$-query algorithm. This upper bound was further pushed to $O(n/\epsilon)$ (with a poly-logarithmic improvement)

by Baleshzar et al. in their later work [Bal+17] studying the testing of unateness of real-valued functions, and it was also shown to be optimal under this setting.

More recently, together with Chen and Waingarten, we showed in [CWX17b] an adaptive algorithm for testing unateness with query complexity $\tilde{O}(n^{3/4}/\epsilon^2)$. It has one-sided errors and is the most efficient algorithm known to date (when $1/\epsilon << n^{1/4}$).

On the lower bound side of unateness testing, the only result before our work was $\Omega(\sqrt{n}/\epsilon)$ by Baleshzar et al. [Bal+16] for non-adaptive tester with one-sided errors.

In Chapter 2 we will also present our new results regarding lower bounds of unateness testing [CWX17a]:

Theorem 1.2.4. *There exists some constant $\epsilon_0 \in (0,1)$ such that any adaptive algorithm that ϵ_0-tests unateness must make at least $\tilde{\Omega}(n^{2/3})$ queries.*

Theorem 1.2.5. *There exists some constant $\epsilon_0 \in (0,1)$ such that any non-adaptive algorithm with one-sided errors that ϵ_0-tests unateness must make at least $\tilde{\Omega}(n)$ queries.*

These two results, combined with the $\tilde{O}(\sqrt{n}/\epsilon^2)$-query algorithm from [KMS15] for testing monotoncity, show for the first time that testing unateness is computationally harder (with a polynomial gap) than testing monotonicity in both adaptive setting as well as non-adaptive setting with one-sided errors. The $\tilde{\Omega}(n)$ lower bound for the latter setting matches the non-adaptive upper bound of [CS16] up to poly-logarithmic factors, for some constant ϵ. Combined with the $\tilde{O}(n^{3/4})$-query algorithm given in our work [CWX17b], it also shows that adaptivity helps for testing unateness.

However, for adaptive testing of both monotonicity and unateness, we note there are still gaps remaining between the best upper bounds and lower bounds (for constant ϵ): $\tilde{O}(\sqrt{n})$ vs. $\tilde{\Omega}(n^{1/3})$ for monotonicity, and $\tilde{O}(n^{3/4})$ vs. $\tilde{\Omega}(n^{2/3})$ for unateness.

1.3 Testing of k-juntas

Another well-known class of Boolean functions is the class of k-juntas: functions that depend on at most k of its input variables.

Definition 1.3.1. *Given a positive integer $k < n$, a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a k-junta if there exists a subset $J = \{i_1, i_2, \ldots, i_k\} \subset [n]$ of size k and a function $g : \{0,1\}^k \to \{0,1\}$ such that $f(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_k})$ for all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.*

A crucial notion related to k-juntas is so-called relevant variables. Given some Boolean function f, the ith input variable, for some $i \in [n]$, is said to be relevant if f actually depends on this variable, or in other words, if there exists some string $x \in \{0,1\}^n$ such that $f(x) \neq f(x^{(i)})$. Clearly a Boolean function f is k-junta if and only if the number of f's relevant variables is at most k.

The testing of k-juntas studies the problem of distinguishing whether an unknown function is a k-junta. Usually the more interesting case is when $k << n$ (when there are only a few relevant variables), and because of that we would prefer to have testing algorithms with query complexity that is independent of n.

The testing of k-juntas was first considered in [Fis+04], where they proposed a non-adaptive tester for k-juntas with query complexity $\tilde{O}(k^2/\epsilon)$ [3]. This upper bound was then improved by Blais with an $\tilde{O}(k^{3/2}/\epsilon)$-query non-adaptive tester [Bla08] and an $O(k \log k + k/\epsilon)$-query adaptive tester [Bla09]. The main idea of these algorithms was randomly partitioning $[n]$ into $poly(k)$ many disjoint blocks, finding blocks that contain relevant variables, and rejecting the input when $(k+1)$ such blocks have been found (and we are guaranteed that there are at least $(k+1)$ different relevant variables).

[3] As mentioned, here the query complexity is independent of n, and we hide poly-logarithmic factors of k.

The problem of testing k-juntas adaptively was solved (up to poly-logarithmic factors) by combining Blais's adaptive algorithm [Bla09] with Chockler and Gutfreund's lower bound of $\Omega(k)$ [CG04] (for some constant distance parameter ϵ). For the non-adaptive case, Blais showed a lower bound of $\Omega(k/(\epsilon \log(k/\epsilon))$ for $\epsilon \geq k/2^k$ [Bla08] and Buhrman et al. showed a lower bound of $\Omega(k \log k)$ for constant ϵ [Buh+13]. More recently Servedio et al. showed a lower bound of $\Omega(\frac{k \log k}{\epsilon^c \log(\log(k)/\epsilon^c)})$ for $\epsilon \in [k^{-o_k(1)}, o_k(1)]$ and any absolute constant $c < 1$ [STW15]. This lower bound can be larger than the adaptive upper bound of $O(k \log k + k/\epsilon)$ obtained by Blais [Bla09] for certain choice of ϵ and it proved that adaptivity helps for testing k-juntas. The difference of performance between these non-adaptive lower bounds [Bla08; Buh+13; STW15] is however not big: they are essentially $\tilde{\Omega}(k)$ for constant ϵ, while the best upper bound for non-adaptive testing of k-juntas is $\tilde{O}(k^{3/2}/\epsilon)$ from [Bla08]. We made a breakthrough in [Che+17] by showing a lower bound of $\tilde{\Omega}(k^{3/2}/\epsilon)$ for this problem and matched the upper bound up to poly-logarithmic factors.

While standard property testing of k-juntas with respect to the uniform distribution was very well understood, there were little results regarding this problem when we switch to the more general setting of distribution-free testing.

The adaptive lower bound of $\Omega(k)$ [CG04] and non-adaptive lower bound of $\tilde{\Omega}(k^{3/2})$ [Che+17] for testing under uniform distribution naturally extend to this more general setting, but there was no other results on the lower bound side.

On the algorithm side, Halevy and Kushilevitz showed a way of building distribution-free tester based on uniform-distribution testers and self-correctors [HK07], and Alon and Weinstein proved that there is an $O(2^k)$-query self-corrector for k-juntas [AA12]. Combining these two results and the non-adaptive $poly(k, 1/\epsilon)$-query testers for k-juntas under uniform distribution we already know [Fis+04; Bla08], we get a non-adaptive $O(2^k/\epsilon)$-query distribution-free tester for k-juntas. That is the only algorithm we know prior to our work.

We significantly improve these bounds with the following new results [Che+18]:

Theorem 1.3.2. *Given $\epsilon \in (0,1)$ and any positive integer $k < n$, there is an adaptive distribution-free tester with one-sided errors that ϵ-tests k-junta and makes at most $\tilde{O}(k^2/\epsilon)$ queries.*

Theorem 1.3.3. *Given $k \leq n/200$ and $\epsilon_0 = 1/3$, any non-adaptive distribution-free tester that ϵ_0-tests k-junta must make at least $\Omega(2^{k/3})$ queries.*

The two results show that adaptivity provides an exponential improvement for the query complexity of distribution-free testing of k-juntas, which stands in sharp contrast to the case of testing k-juntas under uniform distribution, where there is only a polynomial gap. The exponential lower bound also illustrates the hardness of distribution-free testing, just like what we saw in the case of monotonicity testing [HK05].

We will present the proofs of these two results in Chapter 3.

1.4 Distribution-free testing of monotone conjunctions

There is another line of works studying distribution-free testing of the class of monotone conjunctions, defined as follows:

Definition 1.4.1. *A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a monotone conjunction if there exists a subset $S \subset [n]$ such that $f(x) = \bigwedge_{i \in S} x_i$. The special case when $S = \emptyset$ gives a constant function that always equals to 1.*

We use MCONJ to denote the class of all monotone conjunctions over $\{0,1\}^n$. While testing MCONJ under uniform distribution only needs $poly(1/\epsilon)$ (independent of n) many queries, Glasner and Servedio proved in [GS09] a lower bound of $\tilde{\Omega}(n^{1/5})$ for the query complexity of adaptive distribution-free testers of MCONJ with some constant distance parameter $\epsilon \in (0,1)$. Then Delov and Ron continued to attack this problem by giving an

$\tilde{O}(\sqrt{n}/\epsilon)$-query adaptive algorithm in [DR11]. There, however, remains a polynomial gap between these two bounds, and we manage to close this gap and pin down the optimal query complexity of this problem at $\tilde{\Theta}(n^{1/3})$ (for constant ϵ and with the poly-logarithmic factors ignored) [CX15], by proving the two theorems as follows:

Theorem 1.4.2. *Given $\epsilon \in (0,1)$, there is an adaptive distribution-free tester with one-sided errors that ϵ-tests MCONJ and makes at most $\tilde{O}(n^{1/3}/\epsilon^5)$ queries.*

Theorem 1.4.3. *There exists some constant $\epsilon_0 \in (0,1)$ such that any adaptive distribution-free tester that ϵ_0-tests MCONJ must make at least $\tilde{\Omega}(n^{1/3})$ queries.*

We will present the proofs of these two theorems in Chapter 4.

We note that our results can also be extended to other classes of Boolean functions. First, our upper bound can be extended to the class of general conjunctions (i.e. f is the conjunction of a subset of literals in $\{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$, and let's use CONJ to denote this class) via a reduction to the distribution-free testing of monotone conjucntions.

Theorem 1.4.4. *Given $\epsilon \in (0,1)$, there is an adaptive distribution-free tester with one-sided errors that ϵ-tests CONJ and makes at most $\tilde{O}(n^{1/3}/\epsilon^5)$ queries.*

Second, our lower bound can be extended to the distribution-free testing of general conjunctions, decision lists, as well as linear threshold functions (see their definitions in Chapter 4). We follow the same strategy from Glasner and Servedio's work [GS09] and improve their lower bounds of $\tilde{\Omega}(n^{1/5})$ to $\tilde{\Omega}(n^{1/3})$ for these classes. For general conjunctions, our bounds are also tight up to poly-logarithmic factors when ϵ is a constant.

Theorem 1.4.5. *There exists some constant $\epsilon_0 \in (0,1)$ such that any adaptive distribution-free tester that ϵ_0-tests CONJ must make at least $\tilde{\Omega}(n^{1/3})$ queries. The same lower bound holds for testing decision lists and testing linear threshold functions.*

We will also present proofs of these results in Chapter 4.

1.5 Organization

In this thesis, we will discuss several new results on the property testing of monotonicity, unateness, k-juntas, and classes of other basic Boolean functions like monotone conjunctions, under both the standard setting and the distribution-free setting.

In Chapter 2 we will focus on the standard property testing under uniform distribution and discuss about our new lower bounds for the query complexity of testing monotonicity and unateness. Then in Chapter 3 we will switch to the more general setting of distribution-free testing and present our new upper bound and lower bound for the query complexity of testing k-juntas under this setting. In Chapter 4 we will continue to work under the distribution-free setting and present our results on other basic Boolean functions like monotone conjunctions.

In the end we conclude our results and also discuss about open problems and possible directions for related future work.

This page intentionally left blank.

Chapter 2

Testing of Monotonicity and Unateness

In this chapter, we will focus on the standard property testing with respect to uniform distributions and present our new lower bounds for testing two classical properties, monotonicity and unateness . More precisely, we will show a lower bound of $\tilde{\Omega}(n^{1/3})$ for testing of monotonicity and prove Theorem 1.2.2 in Section 2.2. Then we will show two lower bounds for testing of uanteness and prove Theorem 1.2.4 and Theorem 1.2.5, in Section 2.3 and Section 2.4 respectively. We will end this chapter with some discussion in Section 2.5.

Let's first start with some preparation.

2.1 Preparation

In this section we introduce some notation and tools that will be useful.

2.1.1 Notation and definition

We study property testing of monotonicity and unateness in this chapter. Recall that a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is monotone, if for any two strings $x, y \in \{0,1\}^n$ such that $x \prec y$ (i.e., $x_i \leq y_i$ for every $i \in [n]$), we have $f(x) \leq f(y)$; f is unate if there exists a string $r \in \{0,1\}^n$ such that $g(x) = f(x \oplus r)$ is monotone. We use MONO and UNATE to denote the classes of monotone functions and unate functions, respectively.

Also recall that an edge $(x, x^{(i)})$ is monotone with respect to a function f, if it is bi-chromatic and satisfies the monotone conditions: either $f(x) = 0, f(x^{(i)}) = 1$, and

$x_i = 0$ or $f(x) = 1, f(x^{(i)}) = 0$, and $x_i = 1$; it is anti-monotone if it is bi-chromatic but not monotone.

Let's fix an integer N to be defined later (it will be always set to $2^{\sqrt{n}}$ in Section 2.2 and set to $(1 + 1/\sqrt{n})^{n/4}$ in Section 2.3). We use e_i, for each $i \in [N]$, to denote the string in $\{0,1\}^N$ with its kth entry being 0 if $k \neq i$ and 1 if $k = i$; we use $e_{i,i'}$, $i, i' \in [N], i < i'$, to denote the string in $\{0,1,*\}^N$ with its kth entry being 0 if $k < i'$ and $k \neq i$, 1 if $k = i$ or i', and $*$ if $k > i'$. We let $\overline{e}_i$ ($\overline{e}_{i,i'}$) denote the string obtained from e_i ($e_{i,i'}$) by flipping its 0-entries to 1 and 1-entries to 0.

2.1.2 Distance to monotonicity and unateness

Now we review some characterizations of distance to monotonicity and unateness from the literature:

Lemma 2.1.1 (Lemma 4 in [Fis+02]). *Let $f\colon \{0,1\}^n \to \{0,1\}$ be a Boolean function. Then $dist\big(f, \textsc{Mono}\big) = |M|/2^n$, where M is the maximal set of disjoint violating pairs of f.*

Lemma 2.1.2. *Given $f\colon \{0,1\}^n \to \{0,1\}$, let $(E_i^+, E_i^- : i \in [n])$ be a tuple of sets such that (1) each set E_i^+ consists of monotone bi-chromatic edges $(x, x^{(i)})$ along direction i with $x_i = 0$, $f(x) = 0$ and $f(x^{(i)}) = 1$; (2) each set E_i^- consists of anti-monotone bi-chromatic edges $(x, x^{(i)})$ along direction i with $x_i = 0$, $f(x) = 1$ and $f(x^{(i)}) = 0$; (3) all edges in these $2n$ sets are disjoint. Then $dist\big(f, \textsc{Unate}\big) \geq \frac{1}{2^n} \sum_{i=1}^n \min\big\{|E_i^+|, |E_i^-|\big\}$.*

Proof. By definition, the distance of f to unateness is given by

$$dist\big(f, \textsc{Unate}\big) = \min_{r \in \{0,1\}^n} dist\big(f_r, \textsc{Mono}\big),$$

where $f_r(x) = f(x \oplus r)$. On the other hand, since all edges in the $2n$ sets E_i^+ and E_i^- are

disjoint, it follows from Lemma 2.1.1 that

$$dist\big(f_r, \textsc{Mono}\big) \geq \frac{1}{2^n}\left(\sum_{i:r_i=0}\left|E_i^-\right| + \sum_{i:r_i=1}\left|E_i^+\right|\right) \geq \frac{1}{2^n}\sum_{i=1}^{n}\min\left\{|E_i^+|, |E_i^-|\right\}.$$

This finishes the proof of the lemma. □

2.1.3 Tree pruning lemmas

To prepare for our proofs later, now let's consider a rather general setup where a q-query deterministic algorithm A has oracle access to an object $\boldsymbol{O}$ drawn from a distribution $\mathcal{D}$: upon each query w, the oracle with an object O returns $\eta(w, O)$, an element from a finite set $\mathfrak{P}$. Such an algorithm can be equivalently viewed as a tree (let's abuse the notation a little bit and still call this tree A) of depth q, where each internal node u is labelled a query w to make and has $|\mathfrak{P}|$ edges (u, v) leaving u, each labelled a distinct element from $\mathfrak{P}$. (In general the degree of u can be much larger than two; this is the case for all our applications later since we will introduce new oracles that upon a query string $x \in \{0,1\}^n$ returns more information than just $f(x)$.) For this section we do not care about labels of leaves of A. Given A, we present two basic pruning techniques that will help our analysis of algorithms in our lower bound proofs.

Both lemmas share the following setup. Given A and a set E of edges of A we use L_E to denote the set of leaves ℓ that has at least one edge in E along the path from the root to ℓ. Each lemma below states that if E satisfies certain properties with respect to $\mathcal{D}$ that we are interested in, then

$$\Pr_{\boldsymbol{O}\sim\mathcal{D}}\left[\boldsymbol{O} \text{ reaches a leaf in } L_E\right] = o(1). \tag{2.1}$$

This will later allow us to focus on root-to-leaf paths that do not take any edge in E.

For each node u of tree A, we use $\Pr[u]$ to denote the probability of $\boldsymbol{O} \sim \mathcal{D}$ reaching u.

When u is an internal node with $\Pr[u] > 0$ we use $q(u)$ to denote the following conditional probability:

$$q(u) = \Pr_{\boldsymbol{O}\sim\mathcal{D}}\left[\boldsymbol{O} \text{ follows an edge in } E \text{ at } u \,\middle|\, \boldsymbol{O} \text{ reaches } u\right] = \frac{\sum_{(u,v)\in E}\Pr[v]}{\Pr[u]}.$$

We start with the first pruning lemma; it is trivially implied by the second pruning lemma, but we keep it because of its conceptual simplicity.

Lemma 2.1.3. *Given E, if $q(u) = o(1/q)$ for every internal node u with $\Pr[u] > 0$, then (2.1) holds.*

Proof. We can partition the set L_E of leaves into $L_E = \bigcup_{i\in[q]} L_i$, where L_i contains leaves with its first edge from E being the ith edge along its root-to-leaf path. We also write E_i as the set of edges in E at the ith level (i.e., they appear as the ith edge along root-to-leaf paths). Then for each i,

$$\Pr_{\boldsymbol{O}\sim\mathcal{D}}\left[\boldsymbol{O} \text{ reaches } L_i\right] \le \sum_{(u,v)\in E_i}\Pr[v] = \sum_{u}\sum_{(u,v)\in E_i}\Pr[v] = \sum_{u}\Pr[u]\cdot o(1/q).$$

Note that the sum is over certain nodes u at the same depth $(i-1)$. Therefore, $\sum_u \Pr[u] \le 1$ and the proof is completed by taking a union bound over L_i, $i \in [q]$. □

Next, for each leaf ℓ with $\Pr[\ell] > 0$ and the root-to-ℓ path being $u_1u_2\cdots u_{k+1} = \ell$, we let $q^*(\ell)$ denote $\sum_{i\in[k]} q(u_i)$. The second pruning lemma states that (2.1) holds if $q^*(\ell) = o(1)$ for all such ℓ.

Lemma 2.1.4. *If every leaf ℓ of A with $\Pr[\ell] > 0$ satisfies $q^*(\ell) = o(1)$, then (2.1) holds.*

Proof. The first part of the proof goes exactly the same as in the proof of the first lemma.

Let A' be the set of internal nodes u with $\Pr[u] > 0$. After a union bound over L_i, $i \in [q]$,

$$\Pr_{\boldsymbol{O}\sim\mathcal{D}}\left[\boldsymbol{O} \text{ reaches } L_E\right] \le \sum_{u\in A'}\Pr[u]\cdot q(u).$$

Let L_u be the leaves in the subtree rooted at $u \in A'$. We can rewrite $\Pr[u]$ as $\sum_{\ell \in L_u} \Pr[\ell]$. Thus,

$$\Pr_{\boldsymbol{O} \sim \mathcal{D}} \left[\boldsymbol{O} \text{ reaches } L_E \right] \leq \sum_{u \in A'} \sum_{\ell \in L_u} \Pr[\ell] \cdot q(u) = \sum_{\ell} \Pr[\ell] \cdot q^*(\ell),$$

where the last sum is over leaves ℓ with $\Pr[\ell] > 0$; the last equation follows by switching the order of the two sums. The lemma follows from $q^*(\ell) = o(1)$ and $\sum_\ell \Pr[\ell] = 1$. □

2.2 An $\tilde{\Omega}(n^{1/3})$ lower bound for testing of monotonicity

In this section we will show any (adaptive) algorithm for monotonicity testing must make at least $\tilde{\Omega}(n^{1/3})$ queries (for some constant distance parameter ϵ) and prove Theorem 1.2.2. The proof follows Yao's mini-max principle, and is heavily inspired by ideas from the lower bound of $\tilde{\Omega}(n^{1/4})$ for monotonicity testing in [BB16].

We will start with some high level intuition of our proof in Section 2.2.1. Following Yao's principle, we then define two distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ over Boolean functions in Section 2.2.2, such that each function drawn from distribution $\mathcal{D}_{yes}$ is monotone, and most functions drawn from distribution $\mathcal{D}_{no}$ are far from monotone. In the end we show in Section 2.2.3 that it's hard for a deterministic algorithm to distinguish whether an input function f is drawn from distribution $\mathcal{D}_{yes}$ or $\mathcal{D}_{no}$ with $\tilde{O}(n^{1/3})$ queries to the black-box oracle of f, which finishes the proof.

2.2.1 Intuition

We discuss about high level ideas behind our proof in this section. As mentioned, our proof is heavily inspired by the lower bound proof for monotonicity testing in [BB16], and let's start by reviewing the hard functions they used (i.e., Talagrand's random DNFs). Employing Yao's minimax principle as usual, the goal of [BB16] is to (1) construct a pair

of distributions $\mathcal{D}_{yes}{}^*$ and $\mathcal{D}_{no}{}^*$ over Boolean functions from $\{0,1\}^n$ to $\{0,1\}$ such that $\mathbf{f} \sim \mathcal{D}_{yes}{}^*$ is always monotone while $\mathbf{g} \sim \mathcal{D}_{no}{}^*$ is $\Omega(1)$-far from monotone with probability $\Omega(1)$; (2) show that no deterministic algorithm with a small number of queries can distinguish them (see Lemma 2.2.5 later).

Let $N = 2^{\sqrt{n}}$[1]. A function $\mathbf{f}$ from $\mathcal{D}_{yes}{}^*$ is drawn using the following procedure. We first sample a sequence of N random sub-hypercubes $\mathbf{H}_i$ in $\{0,1\}^n$. Each $\mathbf{H}_i$ is defined by a random term $\mathbf{T}_i$ with $x \in \mathbf{H}_i$ if $\mathbf{T}_i(x) = 1$, where $\mathbf{T}_i$ is the conjunction of $\sqrt{n}$ random variables sampled uniformly from $[n]$ (so each $\mathbf{H}_i$ has dimension $n - \sqrt{n}$). By a simple calculation most likely the $\mathbf{H}_i$'s have little overlap between each other and they together cover an $\Omega(1)$-fraction of $\{0,1\}^n$. Informally we consider $\mathbf{H}_i$'s together as a random *partition* of $\{0,1\}^n$ where each $x \in \{0,1\}^n$ belongs to a unique $\mathbf{H}_i$ (for now do not worry about cases when x lies in none or multiple $\mathbf{H}_i$'s). Next we sample for each $\mathbf{H}_i$ a random dictatorship function $\boldsymbol{h}_i(x) = x_{\boldsymbol{k}_i}$ with $\boldsymbol{k}_i$ drawn uniformly from $[n]$. The final function is $\mathbf{f}(x) = \boldsymbol{h}_i(x)$ for each $x \in \mathbf{H}_i$ (again do not worry about cases when x lies in none or multiple $\mathbf{H}_i$'s). A function $\mathbf{g}$ from $\mathcal{D}_{no}{}^*$ is drawn using the same procedure except that each $\boldsymbol{h}_i$ is now a random anti-dictatorship function $\boldsymbol{h}_i(x) = \overline{x_{\boldsymbol{k}_i}}$ with $\boldsymbol{k}_i$ sampled uniformly from $[n]$.

Note that the distributions sketched here are slightly different from [BB16] (see Section 2.5). For $\mathcal{D}_{no}{}^*$ in particular, instead of associating each $\mathbf{H}_i$ with an independent, random anti-dictatorship $\boldsymbol{h}_i$, [BB16] draws $\sqrt{n}$ anti-dictatorship functions *in total* and associates each $\mathbf{H}_i$ with one of them randomly.[2] While this gives a connection to the noise sensitivity results of [MO03] on Talagrand functions, it makes the functions harder to analyze and generalize due to the correlation between $\boldsymbol{h}_i$'s.

---

[1] $N = 2^{\sqrt{n}}$ is meant to be an integer. It won't change the correctness of the proof (at least asymptotically for large enough n) by switching this parameter to its nearest integer, and here we just assume N is an integer for convenience. We will make the same assumption for all our future parameters like $\sqrt{n}, n^{1/3}$ and $\sqrt{n}\log n$.

[2] Note that this is very close but also not exactly the same as the distributions used in [BB16]; see Section 2.5.

By handling the cases when x belongs to none or multiple $\mathbf{H}_i$'s properly, one can show $\mathbf{f}$ is always monotone. On the other hand, $\mathbf{g}$ is far from monotone as (intuitively) $\mathbf{H}_i$'s are mostly disjoint and within each $\mathbf{H}_i$, $\mathbf{g}$ is anti-monotone due to the anti-dictatorship $\boldsymbol{h}_i$.

At a high level one can view the terms $\mathbf{T}_i$ together as an *addressing function* in the construction of ${\mathcal{D}_{yes}}^*$ and ${\mathcal{D}_{no}}^*$, which maps each x to one of the N independent anti-dictatorship functions $\boldsymbol{h}_i$, by randomly partitioning $\{0,1\}^n$ using a long sequence of small hypercubes $\mathbf{H}_i$. Conceptually, this is the picture that we will follow to define our two-level Talagrand functions. They will also be built using a random partition of $\{0,1\}^n$ into a sequence of small(er) hypercubes, with the property that (i) if one places a dictatorship function in each hypercube independently at random, the resulting function is monotone, and (ii) if one places a random anti-dictatorship function in each of them, the resulting function is far from monotone with $\Omega(1)$ probability. The main difference lies in the way how the partition is done and how the hypercubes are sampled.

Before introducing the two-level Talagrand function, we explain at a high-level why the pair of distributions ${\mathcal{D}_{yes}}^*$ and ${\mathcal{D}_{no}}^*$ are hard to distinguish (this will allow us to compare them with our new functions and see why the latter are even harder to distinguish). Consider the situation when an algorithm is given an $x \in \mathbf{H}_i$'s with $\mathbf{g}(x) = \boldsymbol{h}_i(x) = 0$ and would like to identify the input function $\mathbf{g}$ in fact comes from ${\mathcal{D}_{no}}^*$. The most straightforward way to do this is to find a violating pair in $\mathbf{H}_i$ by flipping some 1's of x to 0 and hoping to see $\mathbf{g}(y) = 1$ in the new y obtained. The algorithm faces the following dilemma:

1. on the one hand, the algorithm wants to flip as many 1's of x as possible in order to flip the hidden variable $x_{\boldsymbol{k}_i}$ associated with the anti-dictatorship function $\boldsymbol{h}_i$ ($\boldsymbol{h}_i(x) = \overline{x_{\boldsymbol{k}_i}}$) and make $\mathbf{g}(y) = 1$;
2. on the other hand, it is very unlikely for the algorithm to flip many (say $\omega(\sqrt{n}\log n)$) 1's of x without moving y outside of $\mathbf{H}_i$ (which happens if one of the 1-entries flipped lies in $\mathbf{T}_i$), and when this happens, $\mathbf{g}(y)$ provides essentially no information about $\boldsymbol{k}_i$.

So $\mathbf{g}$ is very resilient against such attacks. However, consider the case when $x \in \mathbf{H}_i$ and $\boldsymbol{h}_i(x) = 1$; then, the algorithm may try to find a violating pair in $\mathbf{H}_i$ by flipping 0's of x to 1, and this time there is (almost) no limitation on how many 0's of x one can flip! In fact flipping 0's to 1's can never move y outside of $\mathbf{H}_i$.[3] In Section 2.5, we leverage this observation to find a violation with $\tilde{O}(n^{1/4})$ queries.

Now we describe the two-level Talagrand function. The random partitions we employ below are more complex; they allow us to upperbound not only the number of 1's of x that an algorithm can flip (without moving outside of the hypercube) but also the number of 0's as well. We use $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ to denote the two distributions.

To draw a function $\mathbf{f}$ from $\mathcal{D}_{yes}$, we partition $\{0,1\}^n$ into N^2 random sub-hypercubes as follows. First we sample as before N random $\sqrt{n}$-terms $\mathbf{T}_i$ to obtain $\mathbf{H}_i$. After that, we further partition each $\mathbf{H}_i$, by independently sampling N random $\sqrt{n}$-clauses $\mathbf{C}_{i,j}$, with each of them being the disjunction of $\sqrt{n}$ random variables sampled from $[n]$ uniformly. The terms $\mathbf{T}_i$ and clauses $\mathbf{C}_{i,j}$ together define N^2 sub-hypercubes $\mathbf{H}_{i,j}$: $x \in \mathbf{H}_{i,j}$ if $\mathbf{T}_i(x) = 1$ and $\mathbf{C}_{i,j}(x) = 0$. The rest is very similar. We sample a random dictatorship function $\boldsymbol{h}_{i,j}$ for each $\mathbf{H}_{i,j}$; the final function $\mathbf{f}$ has $\mathbf{f}(x) = \boldsymbol{h}_{i,j}(x)$ for $x \in \mathbf{H}_{i,j}$.[4] A function $\mathbf{g}$ from $\mathcal{D}_{no}$ is drawn using the same procedure except that $\boldsymbol{h}_{i,j}$'s are independent random anti-dictatorship functions. We call such functions two-level Talagrand functions, as one can view each of them as a two-level structure with the top being a Talagrand DNF and the bottom being N Talagrand CNFs, one attached with each term of the top DNF. See Figure 2.2 for a visual depiction.

By a simple calculation, (most likely) the $\mathbf{H}_{i,j}$'s have little overlap and cover an $\Omega(1)$-fraction of $\{0,1\}^n$. This is why $\mathbf{g}$ is far from monotone. It will become clear after the formal

[3] While we tried to keep the high-level description here simple, there is indeed a truncation that is always applied on $\mathbf{g}$, where one set $\mathbf{g}(x) = 1$ for $|x| > (n/2) + \sqrt{n}$, $\mathbf{g}(x) = 0$ for $|x| < (n/2) - \sqrt{n}$, and keep $\mathbf{g}(x)$ the same only when x lies in the middle layers with $|x|$ between $(n/2) - \sqrt{n}$ and $(n/2) + \sqrt{n}$. But even with the truncation in place, one can take advantage of this observation and find a violation in $\mathbf{g}$ using $\tilde{O}(n^{1/4})$ queries. See details in Section 2.5

[4] Again, do not worry about cases when x lies in none or multiple $\mathbf{H}_{i,j}$'s.

definition of $\mathcal{D}_{yes}$ that f is monotone; this relies on how exactly we handle cases when x lies in none or multiple $\mathbf{H}_i$'s.

Conceptually the construction of $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ follows the same high-level picture: the terms $\mathbf{T}_i$ and clauses $\mathbf{C}_{i,j}$ together serve as an addressing function, which we refer to as a *multiplexer* in the proof (see Figure 2.1 for a visual depiction). It maps each string x to one of the N^2 independent and random dictatorship or anti-dictatorship $\boldsymbol{h}_{i^*,j^*}$, depending on whether the function is from $\mathcal{D}_{yes}$ or $\mathcal{D}_{no}$. Terms $\mathbf{T}_i$ in the first level of multiplexing determines i^* and clauses $\mathbf{C}_{i^*,j}$ in the second level of multiplexing determines j^*. The new two-level Talagrand functions are harder than those of [BB16] since, starting with a string $x \in \mathbf{H}_{i,j}$, not only flipping $\omega(\sqrt{n}\log n)$ many 1's would move it outside of $\mathbf{H}_{i,j}$ with high probability (because the term $\mathbf{T}_i$ is most likely no longer satisfied), the same holds when flipping $\omega(\sqrt{n}\log n)$ many 0's to 1 (because the clause $\mathbf{C}_{i,j}$ is most likely no longer falsified). We will make the argument above more formal in the next two sections.

2.2.2 Distributions

As promised, in this section we present a pair of distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ supported on Boolean functions $f : \{0,1\}^n \to \{0,1\}$. We then show that every $\mathbf{f} \sim \mathcal{D}_{yes}$ is monotone, and $\mathbf{f} \sim \mathcal{D}_{no}$ is $\Omega(1)$-far from monotone with probability $\Omega(1)$.

Let $N = 2^{\sqrt{n}}$. A function $\mathbf{f} \sim \mathcal{D}_{yes}$ is drawn using the following procedure:

1. Sample a pair $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$. The pair $(\mathbf{T}, \mathbf{C})$ is then used to define a *multiplexer* map $\mathbf{\Gamma} = \mathbf{\Gamma}_{\mathbf{T},\mathbf{C}} : \{0,1\}^n \to (N \times N) \cup \{0^*, 1^*\}$.[5] Both definitions of $\mathcal{E}$ and $\mathbf{\Gamma}$ will be described next.
2. Sample $\boldsymbol{H} = (\boldsymbol{h}_{i,j} : i, j \in [N])$ from a distribution $\mathcal{E}_{\text{yes}}$, where each $\boldsymbol{h}_{i,j}$ is a random dictatorship Boolean function that maps $\{0,1\}^n$ to $\{0,1\}$: $\boldsymbol{h}_{i,j}(x) = x_{\boldsymbol{k}_{i,j}}$ with $\boldsymbol{k}_{i,j}$ sampled independently for each $\boldsymbol{h}_{i,j}$ and uniformly at random from $[n]$.

[5] We use 0^* and 1^* to denote two special symbols (instead of the Kleene closure of 0 and 1).

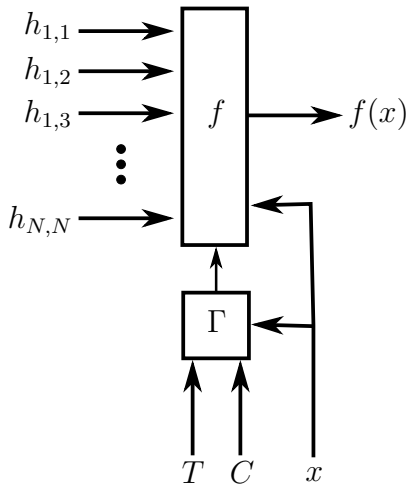

Figure 2.1: An illustration of the function $f = f_{T,C,H}$ and its dependency on T, C and H.

3. Finally, $\mathbf{f} = \mathbf{f}_{\mathbf{T},\mathbf{C},\boldsymbol{H}} : \{0,1\}^n \to \{0,1\}$ is defined as follows: $\mathbf{f}(x) = 1$ if $|x|$ is greater than $(n/2) + \sqrt{n}$; $\mathbf{f}(x) = 0$ if $|x|$ is less than $(n/2) - \sqrt{n}$; if $|x|$ is between $(n/2) - \sqrt{n}$ and $(n/2) + \sqrt{n}$ we define:

$$\mathbf{f}(x) = \begin{cases} 0 & \text{if } \boldsymbol{\Gamma}(x) = 0^* \\ 1 & \text{if } \boldsymbol{\Gamma}(x) = 1^* \\ \boldsymbol{h}_{\boldsymbol{\Gamma}(x)}(x) & \text{otherwise (i.e., } \boldsymbol{\Gamma}(x) \in N \times N) \end{cases}$$

On the other hand a function $\mathbf{f} = \mathbf{f}_{\mathbf{T},\mathbf{C},\boldsymbol{H}} \sim \mathcal{D}_{no}$ is drawn using the same procedure, with the only difference being that $\boldsymbol{H} = (\boldsymbol{h}_{i,j} : i,j \in [N])$ is drawn from $\mathcal{E}_{\text{no}}$ (instead of $\mathcal{E}_{\text{yes}}$): each $\boldsymbol{h}_{i,j}(x) = \overline{x_{\boldsymbol{k}_{i,j}}}$ is a random anti-dictatorship function with $\boldsymbol{k}_{i,j}$ drawn independently and uniformly from $[n]$.

Remark 1. *Given the same truncation done in both $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$, it suffices to show a lower bound against algorithms that query strings in the* middle layers *only:* $(n/2) - \sqrt{n} \leq |x| \leq (n/2) + \sqrt{n}$.

Next we describe the distribution $\mathcal{E}$ in details. $\mathcal{E}$ is uniform over all pairs (T, C) of the following form: $T = (T_i : i \in [N])$ with $T_i : [\sqrt{n}] \to [n]$ and $C = (C_{i,j} : i,j \in [N])$ with $C_{i,j} : [\sqrt{n}] \to [n]$. We call T_i's the (DNF) *terms* and $C_{i,j}$'s the (CNF) *clauses*. Equivalently,

to draw a pair $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$:

- For each $i \in [N]$, we sample a random term $\mathbf{T}_i$ by sampling $\mathbf{T}_i(k)$ independently and uniformly from $[n]$ for each $k \in [\sqrt{n}]$, with $\mathbf{T}_i(k)$ viewed as the kth variable of $\mathbf{T}_i$.
- For each $i, j \in [N]$, we sample a random clause $\mathbf{C}_{i,j}$ by sampling $\mathbf{C}_{i,j}(k)$ independently and uniformly from $[n]$ for each $k \in [\sqrt{n}]$, with $\mathbf{C}_{i,j}(k)$ viewed as the kth variable of $\mathbf{C}_{i,j}$.

Given a pair (T, C), we interpret T_i as a term and abuse the notation to write

$$T_i(x) = \bigwedge_{k \in [\sqrt{n}]} x_{T_i(k)}$$

as a Boolean function over n variables. We say x *satisfies* T_i when $T_i(x) = 1$. We interpret each $C_{i,j}$ as a clause and abuse the notation to write

$$C_{i,j}(x) = \bigvee_{k \in [\sqrt{n}]} x_{C_{i,j}(k)}$$

as a Boolean function over n variables. Similarly we say x *falsifies* $C_{i,j}$ when $C_{i,j}(x) = 0$.

Each pair (T, C) in the support of $\mathcal{E}$ defines a multiplexer map $\Gamma = \Gamma_{T,C} : \{0,1\}^n \to (N \times N) \cup \{0^*, 1^*\}$. Informally speaking, Γ consists of two levels: the first level uses the terms T_i in T to pick the first index $i' \in [N]$; the second level uses the clauses $C_{i',j}$ in C to pick the second index $j' \in [N]$. Sometimes Γ may choose to directly determine the value of the function by setting $\Gamma(x) \in \{0^*, 1^*\}$.

Formally, (T, C) defines Γ as follows. Given an $x \in \{0,1\}^n$ we have $\Gamma(x) = 0^*$ if $T_i(x) = 0$ for all $i \in [N]$ and $\Gamma(x) = 1^*$ if $T_i(x) = 1$ for at least two different i's in $[N]$. Otherwise there is a unique i' with $T_{i'}(x) = 1$, and the multiplexer enters the second level. Next, we have $\Gamma(x) = 1^*$ if $C_{i',j}(x) = 1$ for all $j \in [N]$ and $\Gamma(x) = 0^*$ if $C_{i',j}(x) = 0$ for at least two different j's in $[N]$. Otherwise there is a unique $j' \in [N]$ with $C_{i',j'}(x) = 0$ and in this case the multiplexer outputs $\Gamma(x) = (i', j')$.

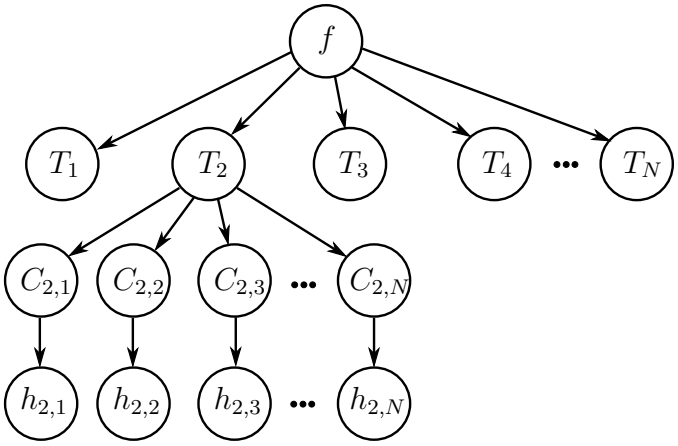

Figure 2.2: Picture of a function f in the support of $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$. We think of evaluating $f(x)$ as following the arrows down the tree. The first level represents multiplexing $x \in \{0,1\}^n$ with respect to the terms in T. If x satisfies no terms, or multiple terms, then f outputs 0, or 1, respectively. If x satisfies T_i for a *unique* term T_i (T_2 in the picture), then we follow the arrow to T_i and proceed to the second level. If x falsifies no clause, or multiple clauses, then f outputs 1, or 0, respectively. If x falsifies a unique clause $C_{i,j}$, then we follow the arrow to $C_{i,j}$ and output $h_{i,j}(x)$.

This finishes the definition of $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$. Figure 2.2 below gives a graphical representation of such functions. We now prove the properties of $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ promised at the beginning.

Lemma 2.2.1. *Every function f in the support of $\mathcal{D}_{yes}$ is monotone.*

Proof. Consider $f = f_{T,C,H}$ with (T, C) from the support of $\mathcal{E}$ and H from the support of $\mathcal{E}_{\text{yes}}$. Let $x \in \{0,1\}^n$ be a string with $f(x) = 1$ and $x_i = 0$ for some i. Let $x' = x^{(i)}$. We show that $f(x') = 1$.

First note that every term in T satisfied by x remains satisfied by x'; every clause satisfied by x remains satisfied by x'. As a result if $\Gamma(x) = 1^*$ then $\Gamma(x') = 1^*$ as well and $f(x') = 1$. If $\Gamma(x) = (i,j)$ for some $i, j \in [N]$, then $h_{i,j}(x) = f(x) = 1$. For this case we have either $\Gamma(x') = 1^*$ and $f(x') = 1$, or $f(x') = h_{i,j}(x')$ and $h_{i,j}(x') = h_{i,j}(x) = 1$ because $h_{i,j}$ here is a dictatorship function. This finishes the proof since $\Gamma(x)$ cannot be 0^* when $f(x) = 1$. □

Lemma 2.2.2. *A function* $\mathbf{f} \sim \mathcal{D}_{no}$ *is* $\Omega(1)$*-far-from monotone with probability* $\Omega(1)$*.*

Proof. Fix a pair (T, C) from the support of $\mathcal{E}$ and an H from the support of $\mathcal{E}_{\text{no}}$. Let $f = f_{T,C,H}$.

Consider the set $X \subset \{0,1\}^n$ consisting of strings x in the middle layers (i.e., $|x| \in [(n/2) \pm \sqrt{n}]$) with $f(x) = 1, \Gamma(x) = (i, j)$ for some $i, j \in [N]$ (instead of 0^* or 1^*), and $h_{i,j}$ being an anti-dictator function on the $k_{i,j}$th variable for some $k_{i,j} \in [n]$ (so $x_{k_{i,j}} = 0$). For each $x \in X$, we write $\eta(x)$ to denote the index $k_{i,j}$ for $h_{i,j}$ and use x^* to denote $x^{(\eta(x))}$. (Ideally, we would like to conclude that (x, x^*) is a violating edge of f as $h_{i,j}(x^*) = 0$. However, flipping one bit potentially may also change the value of the multiplexer map Γ. So we need to further refine the set X.)

Next we define the following two events with respect to a string $x \in X$ (with $\Gamma(x) = (i, j)$):

- $E_1(x)$: This event happens when $\eta(x) \neq C_{i,j}(\ell)$ for any $\ell \in [\sqrt{n}]$ (and thus, $C_{i,j}(x^*) = 0$);
- $E_2(x)$: This event happens when $T_{i'}(x^*) = 0$ for all $i' \neq i \in [N]$.

We use X' to denote the set of strings $x \in X$ such that both $E_1(x)$ and $E_2(x)$ happen. The following claim shows that (x, x^*) for every $x \in X'$ is a violating edge of f.

Claim 2.2.3. *For each* $x \in X'$*,* (x, x^*) *is a violating edge of* f*.*

Proof. It suffices to show that $f(x^*) = 0$. As x satisfies a unique term T_i (T_i cannot have $\eta(x)$ as a variable because $x_{\eta(x)} = 0$), it follows from $E_2(x)$ that x^* uniquely satisfies the same T_i. It follows from $E_1(x)$ that x^* uniquely (among all $C_{i,j'}, j' \in [N]$) falsifies the same clause $C_{i,j}$. As a result, $f(x^*) = h_{i,j}(x^*) = 0$. □

Furthermore, the violating edges (x, x^*) induced by strings $x \in X'$ are indeed disjoint. This is because, given x^*, one can uniquely reconstruct x by locating $h_{i,j}$ using $\Gamma(x^*)$ and

flipping the $k_{i,j}$th bit of x^* if $h_{i,j}$ is an anti-dictator function over the $k_{i,j}$th variable. Therefore, it suffices to show that $\boldsymbol{X}'$ (as a random set) has size $\Omega(2^n)$ with probability $\Omega(1)$, over choices of $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{\text{no}}$. The lemma then follows from the characterization of [Fis+02] as stated in Lemma 2.1.1.

Finally we work on the size of $\boldsymbol{X}'$. Fix a string $x \in \{0,1\}^n$ in the middle layers. The next claim shows that, when $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{\text{no}}$, $\boldsymbol{X}'$ contain x with $\Omega(1)$ probability.

Claim 2.2.4. *For each $x \in \{0,1\}^n$ with $(n/2) - \sqrt{n} \le |x| \le (n/2) + \sqrt{n}$, we have*

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\, \boldsymbol{H}\sim\mathcal{E}_{no}} \left[x \in \boldsymbol{X}'\right] = \Omega(1).$$

Proof. Fix an $x \in \{0,1\}^n$ in the middle layers. We calculate the probability of $x \in \boldsymbol{X}'$.

We equivalently partition the event of $x \in \boldsymbol{X}'$ into $\Theta(nN^2)$ subevents indexed by $i, j \in [N]$ and $k \in [n]$ with $x_k = 0$. Each subevent corresponds to 1) Condition on $\mathbf{T}$: both x and $x^{(k)}$ satisfy uniquely the term $\mathbf{T}_i$; 2) Condition on $\mathbf{C}$: both x and $x^{(k)}$ falsify uniquely the clause $\mathbf{C}_{i,j}$; 3) Condition on $\boldsymbol{H}$: $\boldsymbol{h}_{i,j}$ is the anti-dictatorship function over the kth variable. The probability that 3) happens is clearly $1/n$.

The probability that 1) happens is at least

$$\left(1 - \left(\frac{n/2 + \sqrt{n} + 1}{n}\right)^{\sqrt{n}}\right)^{N-1} \times \left(\frac{n/2 - \sqrt{n}}{n}\right)^{\sqrt{n}} = \Omega\left(\frac{1}{N}\right).$$

The probability that 2) happens is at least

$$\left(1 - \left(\frac{n/2 + \sqrt{n}}{n}\right)^{\sqrt{n}}\right)^{N-1} \times \left(\frac{n/2 - \sqrt{n} - 1}{n}\right)^{\sqrt{n}} = \Omega\left(\frac{1}{N}\right).$$

As a result, the probability of $x \in \boldsymbol{X}'$ is $\Omega(nN^2) \times \Omega(1/N) \times \Omega(1/N) \times \Omega(1/n) = \Omega(1)$. □

From Claim 2.2.4 and the fact that there are $\Omega(2^n)$ strings in the middle layer, the ex-

pected size of $\boldsymbol{X}'$ is $\Omega(2^n)$. Via Markov inequality we know $|\boldsymbol{X}'| = \Omega(2^n)$ with probability $\Omega(1)$. This finishes the proof. □

Given Lemma 2.2.1, Lemma 2.2.2 and same argument from [BB16], Theorem 1.2.2 follows directly from the following lemma which we show in the rest of the section. For the rest of the proof we fix the number of queries $q = n^{1/3}/\log^2 n$.

Lemma 2.2.5. *Let B be any q-query, deterministic algorithm with black-box oracle access to f. Then*

$$\Pr_{\mathbf{f}\sim\mathcal{D}_{yes}}\left[B \text{ accepts } \mathbf{f}\right] \leq \Pr_{\mathbf{f}\sim\mathcal{D}_{no}}\left[B \text{ accepts } \mathbf{f}\right] + o(1).$$

Since f is truncated in both distributions, we may assume WLOG that B queries strings in the middle layers only (i.e., strings x with $|x|$ between $(n/2) - \sqrt{n}$ and $(n/2) + \sqrt{n}$).

2.2.3 Proof of Lemma 2.2.5

We prove Lemma 2.2.5 in this section. Let's start with some more preparation for the proof.

2.2.3.1 Signatures and the new oracle

Let (T, C) be a pair from the support of $\mathcal{E}$ and H be a tuple from the support of $\mathcal{E}_{\text{yes}}$ or $\mathcal{E}_{\text{no}}$. Towards Lemma 2.2.5, we are interested in deterministic algorithms that have black-box oracle access to $f = f_{T,C,H}$ and attempt to distinguish $\mathcal{D}_{yes}$ from $\mathcal{D}_{no}$ (i.e., accept if H is from $\mathcal{E}_{\text{yes}}$ and reject if it is from $\mathcal{E}_{\text{no}}$).

For convenience of our lower bound proof, we assume below that the black-box oracle returns more than just $f(x)$ for each query $x \in \{0,1\}^n$; instead of simply returning $f(x)$, the oracle returns a 4-tuple (σ, τ, a, b) called the *full signature* of $x \in \{0,1\}^n$ with respect to (T, C, H) (see Definition 2.2.7 below). It will become clear later that $f(x)$ can always be derived correctly from the full signature of x and thus, query lower bounds against the

new oracle carry over to the standard oracle. Once the new oracle is introduced, we may actually ignore the function f and view any algorithm as one that has oracle access to the hidden triple (T, C, H) and attempts to tell whether H is from $\mathcal{E}_{\text{yes}}$ or $\mathcal{E}_{\text{no}}$.

We first give the syntactic definition of *full signatures*.

Definition 2.2.6. *We use $\mathfrak{P}$ to denote the set of all 4-tuples (σ, τ, a, b) with $\sigma \in \{0,1,*\}^N$ and $\tau \in \{0,1,*\}^N \cup \{\perp\}$ and $a, b \in \{0,1,\perp\}$ satisfying the following properties:*

1. *σ is either 1) the all-0 string 0^N; 2) e_i for some $i \in [N]$; or 3) $e_{i,i'}$ for some $i, i' \in [N]$, $i < i'$.*
2. *$\tau = \perp$ if σ is of case 1) or 3). Otherwise (when $\sigma = e_i$ for some i), $\tau \in \{0,1,*\}^N$ is either 1) the all-1 string 1^N; 2) $\overline{e}_j$ for some $j \in [N]$; or 3) $\overline{e}_{j,j'}$ for some $j, j' \in [N]$, $j < j'$.*
3. *$a = b = \perp$ unless: 1) If $\sigma = e_i$ and $\tau = \overline{e}_j$ for some $i, j \in [N]$, then $a \in \{0,1\}$ and $b = \perp$; or 2) If $\sigma = e_i$ and $\tau = \overline{e}_{j,j'}$ for some $i \in [N]$ and $j, j' \in [N], j < j'$, then $a, b \in \{0,1\}$.*

We next define semantically the full signature of $x \in \{0,1\}^n$ with respect to (T, C, H).

Definition 2.2.7 (Full signature). *We say (σ, τ, a, b) is the* full signature *of a string $x \in \{0,1\}^n$ with respect to (T, C, H) if it satisfies the following properties:*

1. *First, $\sigma \in \{0,1,*\}^N$ is determined by T according to one of the following three cases: 1) σ is the all-0 string 0^N if $T_i(x) = 0$ for all $i \in [N]$; 2) If there is a unique $i \in [N]$ with $T_i(x) = 1$, then $\sigma = e_i$; or 3) If there are more than one index $i \in [N]$ with $T_i(x) = 1$, then $\sigma = e_{i,i'}$ with $i, i' \in [N], i < i'$ being the smallest two such indices. We call σ the* term signature *of x.*
2. *Second, $\tau = \perp$ if σ is of case 1) or 3) above. Otherwise, assuming that $\sigma = e_i$, $\tau \in \{0,1,*\}^N$ is determined by $(C_{i,j} : j \in [N])$, according to one of the following cases: 1) τ is the all-1 string 1^N if $C_{i,j}(x) = 1$ for all $j \in [N]$; 2) If there is a unique*

$j \in [N]$ *with* $C_{i,j}(x) = 0$*, then* $\tau = \overline{e}_j$*; or 3) If there are more than one index* $j \in [N]$ *with* $C_{i,j}(x) = 0$*, then* $\tau = \overline{e}_{j,j'}$ *with* $j, j' \in [N], j < j'$ *being the smallest two such indices. We call* τ *the* clause signature *of* x*.*

3. *Finally,* $a = b = \perp$ *unless: 1) If* $\sigma = e_i$ *and* $\tau = \overline{e}_j$ *for some* $i, j \in [N]$*, then* $a = h_{i,j}(x)$ *and* $b = \perp$*; or 2) If* $\sigma = e_i$ *and* $\tau = \overline{e}_{j,j'}$ *for some* $i, j, j' \in [N]$ *and* $j < j'$*, then* $a = h_{i,j}(x)$ *and* $b = h_{i,j'}(x)$*.*

It follows from the definitions that the full signature of x with respect to (T, C, H) is in $\mathfrak{P}$. We also define the *full signature* of a set of strings Q with respect to (T, C, H).

Definition 2.2.8. *The* full signature (*map*) *of a set* $Q \subseteq \{0,1\}^n$ *with respect to a triple* (T, C, H) *is a map* $\phi \colon Q \to \mathfrak{P}$ *such that* $\phi(x)$ *is the full signature of* x *with respect to* (T, C, H) *for each* $x \in Q$*.*

For simplicity, we will write $\phi(x) = (\sigma_x, \tau_x, a_x, b_x)$ to specify the term and clause signatures of x as well as the values of a and b in the full signature $\phi(x)$ of x. Intuitively we may view ϕ as two levels of tables with entries in $\{0, 1, *\}$. The (unique) top-level table "stacks" the term signatures σ_x, where each row corresponds to a string $x \in Q$ and each column corresponds to a term T_i in T. In the second level a table appears for a term T_i if the term signature of some string $x \in Q$ is e_i. In this case the second-level table at T_i "stacks" the clause signatures τ_x for each $x \in Q$ with $\sigma_x = e_i$ where each row corresponds to such an x and each column corresponds to a clause $C_{i,j}$ in C. (The number of columns is still N since we only care about clauses $C_{i,j}$, $j \in [N]$ in the table at T_i.)

The lemma below shows that the new oracle is at least as powerful as the standard oracle.

Lemma 2.2.9. *Let* (T, C) *be from the support of* $\mathcal{E}$ *and* H *from the support of* $\mathcal{E}_{yes}$ *or* $\mathcal{E}_{no}$*. Given any string* $x \in \{0,1\}^n$*,* $f_{T,C,H}(x)$ *is determined by its full signature with respect to* (T, C, H)*.*

Proof. First if x does not lie in the middle layers, then $f(x)$ is determined by $|x|$. Below we assume that x lies in the middle layers. Let (σ, τ, a, b) be the full signature of x. There are five cases:

1. (No term satisfied) If $\sigma = 0^N$, then $f(x) = 0$.
2. (Multiple terms satisfied) If $\sigma = e_{i,i'}$ for some $i, i' \in [N]$, then $f(x) = 1$.
3. (Unique term satisfied, no clause falsified) If $\sigma = e_i$ for some $i \in [N]$ but $\tau = 1^N$, then $f(x) = 1$.
4. (Unique term satisfied, multiple clauses falsified) If $\sigma = e_i$ but $\tau = \overline{e}_{j,j'}$, for some $i, j, j' \in [N]$, then $f(x) = 0$.
5. (Unique term satisfied, unique clause satisfied) If $\sigma = e_i$ and $\tau = \overline{e}_j$ for some $i, j \in [N]$, then $f(x) = a$.

This finishes the proof of the lemma. □

Given Lemma 2.2.9, it suffices to consider deterministic algorithms with the new oracle access to a hidden triple (T, C, H), and Lemma 2.2.5 follows directly from the following lemma:

Lemma 2.2.10. *Let B be any q-query algorithm with the new oracle access to (T, C, H). Then*

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{yes}}\Big[B \text{ accepts } (\mathbf{T},\mathbf{C},\boldsymbol{H})\Big] \leq \Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{no}}\Big[B \text{ accepts } (\mathbf{T},\mathbf{C},\boldsymbol{H})\Big] + o(1).$$

Let's fix the algorithm B for the rest of the proof. Equivalently we can view B as a decision tree of depth q, and let's abuse the notation to also denote this tree by B. Each leaf of the tree B is labeled either "accept" or "reject." Each internal node u of B is labeled with a query string $x \in \{0,1\}^n$, and each of its outgoing edges (u, v) is labeled a tuple from $\mathfrak{P}$. We refer to such a tree as a *signature tree*.

As the algorithm executes, it traverses a root-to-leaf path down the tree making queries to the oracle corresponding to queries in the nodes on the path. For instance at node u,

after the algorithm queries x and the oracle returns the full signature of x with respect to the unknown (T, C, H), the algorithm follows the outgoing edge (u, v) with that label. Once a leaf ℓ is reached, B accepts if ℓ is labelled "accept" and rejects otherwise. With B fixed, here for simplicity for each valid triple (T, C, H) we just say (T, C, H) reaches a node (or leaf) u in B, if the algorithm traverses this tree and reaches u when given oracle access to (T, C, H).

Note that the number of children of each internal node is $|\mathfrak{P}|$, which is huge. Algorithms with the new oracle may adapt its queries to the full signatures returned by the oracle, while under the standard oracle, the queries may only adapt to the value of the function at previous queries. Thus, while algorithms making q queries in the standard oracle model can be described by a tree of size 2^q, q-query algorithms with this new oracle are given by signature trees of size $(2^{\Theta(\sqrt{n})})^q$.

We associate each node u in the tree B with a map $\phi_u : Q_u \to \mathfrak{P}$ where Q_u is the set of queries made along the path from the root to u so far (but not including the query in u), and $\phi_u(x)$ is the label of the edge that the root-to-u path takes after querying x. We will be interested in analyzing the following two quantities:

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } u\Big] \quad \text{and} \quad \Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } u\Big].$$

In particular, Lemma 2.2.10 would follow trivially if for every leaf ℓ of B:

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\Big] \leq (1+o(1)) \cdot \Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\Big]. \tag{2.2}$$

However, (2.2) above does not hold in general. Our plan for the rest of the proof is to prune an $o(1)$-fraction of leaves (measured in terms of their total probability under the yes-case) and show (2.2) for the rest.

To better understand these probabilities, we need to first introduce some useful notation.

2.2.3.2 Notation for full signature maps

Given a map $\phi : Q \to \mathfrak{P}$ for some $Q \subseteq \{0,1\}^n$, we write $\phi(x) = (\sigma_x, \tau_x, a_x, b_x)$ for each $x \in Q$ and use $\sigma_{x,i}, \tau_{x,j}$ to denote the ith entry and jth entry of σ_x and τ_x, respectively. Note that $\tau_{x,j}$ is not defined if $\tau_x = \perp$. (Below we will only be interested in $\tau_{x,j}$ if $\sigma_x = e_i$ for some $i \in [N]$.)

Let's introduce the following notation for ϕ: we say ϕ *induces a tuple* $(I; J; P; R; A; \rho)$, where

- The set $I \subseteq [N]$ is given by $I = \{i \in [N] : \exists x \in Q \text{ with } \sigma_{x,i} = 1\}$. (So in terms of the first-level table, I consists of columns that contain at least one 1-entry.)
- $J = (J_i \subseteq [N] : i \in I)$ is a tuple of sets indexed by $i \in I$. For each $i \in I$, we have

  $$J_i = \big\{j \in [N] : \exists x \in Q \text{ with } \sigma_x = e_i \text{ and } \tau_{x,j} = 0\big\}.$$

  (In terms of the second-level table at T_i, J_i consists of columns that contain at least one 0-entry.) By the definition of $\mathfrak{P}$, each x with $\sigma_x = e_i$ can contribute at most two j's to J_i. Also x does not contribute any j to J_i if $\sigma_x = e_{i,i'}$ or $e_{i',i}$, in which case $\tau_x = \perp$, or if $\sigma_x = e_i$ but $\tau_x = 1^N$. So in general J_i can be empty for some $i \in I$.
- $P = (P_i, P_{i,j} : i \in I, j \in J_i)$ is a tuple of two types of subsets of Q. For $i \in I$ and $j \in J_i$,

  $$P_i = \big\{x \in Q : \sigma_{x,i} = 1\big\} \quad \text{and} \quad P_{i,j} = \big\{x \in Q : \sigma_x = e_i \text{ and } \tau_{x,j} = 0\big\}.$$

  (In terms of the first-level table, P_i consists of rows that are 1 on the ith column; in terms of the second-level table for T_i, $P_{i,j}$ consists of rows that are 0 on the jth column.) Note that both P_i and $P_{i,j}$ are not empty by the definition of I and J_i.
- $R = (R_i, R_{i,j} : i \in I, j \in J_i)$ is a tuple of two types of subsets of Q. For $i \in I$ and $j \in J_i$,

  $$R_i = \big\{x \in Q : \sigma_{x,i} = 0\big\} \quad \text{and} \quad R_{i,j} = \big\{x \in Q : \sigma_x = e_i \text{ and } \tau_{x,j} = 1\big\}.$$

(In terms of the first-level table, R_i consists of rows that are 0 on the ith column; in terms of the second-level table at T_i, $R_{i,j}$ consists of rows that are 1 on the jth column.)

- $A = (A_{i,0}, A_{i,1}, A_{i,j,0}, A_{i,j,1} : i \in I, j \in J_i)$ is a tuple of subsets of $[n]$. For $i \in I$ and $j \in J_i$,

$$A_{i,1} = \big\{k \in [n] : \forall x \in P_i,\, x_k = 1\big\} \quad \text{and} \quad A_{i,0} = \big\{k \in [n] : \forall x \in P_i,\, x_k = 0\big\}$$

$$A_{i,j,1} = \big\{k \in [n] : \forall x \in P_{i,j},\, x_k = 1\big\} \quad \text{and} \quad A_{i,j,0} = \big\{k \in [n] : \forall x \in P_{i,j},\, x_k = 0\big\}.$$

  Note that all the sets are well-defined since P_i and $P_{i,j}$ are not empty.

- $\rho = (\rho_{i,j} : i \in I, j \in J_i)$ is a tuple of functions $\rho_{i,j} : P_{i,j} \to \{0, 1\}$. For each $x \in P_{i,j}$, we have $\rho_{i,j}(x) = a_x$ if $\tau_x = \overline{e}_j$ or $\tau_x = \overline{e}_{j,j'}$ for some $j' > j$; $\rho_{i,j}(x) = b_x$ if $\tau_x = \overline{e}_{j',j}$ for some $j' < j$.

Intuitively I is the set of indices of terms with some string $x \in Q$ satisfying the term T_i as reported in σ_x (σ_x equals to 1 rather than $*$, though the latter *may* also have x satisfies T_i), and P_i is the set of such strings while R_i is the set of strings x not satisfying T_i as reported in σ_x. For each $i \in I$, J_i is the set of indices of clauses with some string $x \in P_i$ satisfying T_i *uniquely* and falsifying the clause $C_{i,j}$ as reported in τ_x. $P_{i,j}$ is the set of such strings, and $R_{i,j}$ is the set of strings x which satisfy T_i uniquely but also satisfy $C_{i,j}$ as reported in τ_x. We collect the following facts which are immediate from the definition.

Fact 2.2.11. *Let $(I; J; P; R; A; \rho)$ be the tuple induced by a map $\phi\colon Q \to \Sigma$. Then we have*

- $|I| \le \sum_{i \in I} |P_i| \le 2|Q|$.
- *For each $i \in I$, $|J_i| \le \sum_{j \in J_i} |P_{i,j}| \le 2|P_i|$.*
- *For each $i \in I$ and $j \in J_i$, $|R_i|$ and $|R_{i,j}|$ are at most $|Q|$ (as they are subsets of Q).*
- *For each $i \in I$ and $j \in J_i$, $P_{i,j} \subseteq P_i$, $A_{i,0} \subseteq A_{i,j,0}$, and $A_{i,1} \subseteq A_{i,j,1}$.*

Note that $|I|$ and $\sum_{i \in I} |J_i|$ can be strictly larger than $|Q|$, as some x may satisfy more

than one (but at most two) term with $\sigma_x = e_{i,i'}$ and some x may falsify more than one clause with $\tau_x = \overline{e}_{j,j'}$.

The sets in A are important for the following reasons:

Fact 2.2.12. *Let $\phi : Q \to \mathfrak{P}$ be the full signature map of Q with respect to (T, C, H). Then*

- *For each $i \in I$, $T_i(k) \in A_{i,1}$ for all $k \in [\sqrt{n}]$ and $T_i(x) = 0$ for each $x \in R_i$.*
- *For each $i \in I$ and $j \in J_i$, $C_{i,j}(k) \in A_{i,j,0}$ for all $k \in [\sqrt{n}]$ and $C_{i,j}(x) = 1$ for each $x \in R_{i,j}$.*

Before moving back to the proof, we introduce the following consistency condition on P.

Definition 2.2.13. *Let $(I; J; P; R; A; \rho)$ be the tuple induced by a map $\phi : Q \to \mathfrak{P}$. We say that $P_{i,j}$ for some $i \in I$ and $j \in J_i$ is* 1-consistent *if $\rho_{i,j}(x) = 1$ for all $x \in P_{i,j}$, and* 0-consistent *if $\rho_{i,j}(x) = 0$ for all $x \in P_{i,j}$; otherwise we say $P_{i,j}$ is* inconsistent.

Let ϕ be the full signature map of Q with respect to (T, C, H). If $P_{i,j}$ is 1-consistent, the index k of the variable x_k in the dictatorship or anti-dictatorship function $h_{i,j}$ must lie in $A_{i,j,0}$ (when $h_{i,j}$ is an anti-dictator) or $A_{i,j,1}$ (when $h_{i,j}$ is a dictator); the situation is similar if $P_{i,j}$ is 0-consistent but would be more complicated if $P_{i,j}$ is inconsistent. Below we prune an edge whenever some $P_{i,j}$ in P becomes inconsistent. This way we make sure that $P_{i,j}$'s in every leaf left are consistent.

2.2.3.3 Tree pruning

Consider an edge (u, v) in B. Let $\phi_u \colon Q \to \mathfrak{P}$ and $\phi_v \colon Q \cup \{x\} \to \mathfrak{P}$ be the maps associated with u and v, with x being the query made at u and $\phi_v(x)$ being the label of (u, v). Let $(I; J; P; R; A; \rho)$ and $(I'; J'; P'; R'; A'; \rho')$ be the two tuples induced by ϕ_u and ϕ_v, respectively.

We list some easy facts about how $(I; J; P; R; A; \rho)$ is updated to obtain $(I'; J'; P'; R'; A'; \rho')$.

Fact 2.2.14. *Let $\phi_v(x) = (\sigma_x, \tau_x, a_x, b_x)$ for the string x queried at u. Then we have*

- *The new string x is placed in P'_i if $\sigma_{x,i} = 1$, and is placed in $P'_{i,j}$ if $\sigma_x = e_i$ and $\tau_{x,j} = 0$.*
- *Each new set in P' (i.e., P'_i with $i \notin I$ or $P'_{i,j}$ with either $i \notin I$ or $i \in I$ but $j \notin J_i$), if any, is $\{x\}$ and the corresponding set $A'_{i,1}$ or $A'_{i,j,1}$ is $\{k : x_k = 1\}$ and $A'_{i,0}$ or $A'_{i,j,0}$ is $\{k : x_k = 0\}$.*
- *Each old set in P' (i.e., P'_i with $i \in I$ or $P'_{i,j}$ with $i \in I$ and $j \in J_i$) either stays the same or has x added to the set. For the latter case, $\{k : x_k = 0\}$ is removed from $A_{i,1}$ or $A_{i,j,1}$ and $\{k : x_k = 1\}$ is removed from $A_{i,0}$ or $A_{i,j,0}$ to obtain the new sets in A'.*

Now we are ready to define a set of so-called *bad* edges of B, which will be used to prune B. In the rest of the proof we use α to denote a large enough positive constant.

Definition 2.2.15. *An edge (u, v) is called a* bad *edge if at least one of the following events happens at (u, v) and none of these events happen along the path from the root to u (letting ϕ_u and ϕ_v be the maps associated with u and v, x be the new query string at u, $(I; J; P; R; A; \rho)$ and $(I'; J'; P'; R'; A'; \rho')$ be the tuples that ϕ_u and ϕ_v induce, respectively):*

- *For some $i \in I$, $\left|A_{i,1} \setminus A'_{i,1}\right| \geq \alpha\sqrt{n}\log n$.*
- *For some $i \in I$ and $j \in J_i$, $\left|A_{i,j,0} \setminus A'_{i,j,0}\right| \geq \alpha\sqrt{n}\log n$.*
- *For some $i \in I$ and $j \in J_i$, $P_{i,j}$ is 0-consistent but $P'_{i,j}$ is inconsistent (meaning that x is added to $P_{i,j}$ with $\rho_{i,j}(y) = 0$ for all $y \in P_{i,j}$ but $\rho'_{i,j}(x)$ equals to 1 instead of 0).*
- *For some $i \in I$ and $j \in J_i$, $P_{i,j}$ is 1-consistent but $P'_{i,j}$ is inconsistent (meaning that x is added to $P_{i,j}$ with $\rho_{i,j}(y) = 1$ for all $y \in P_{i,j}$ but $\rho'_{i,j}(x$ equals to 0 instead of 1).*

Moreover, a leaf ℓ of B is bad *if one of the edges along the root-to-ℓ path is bad; ℓ is* good *otherwise.*

The following pruning lemma states that the probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ reaching a bad leaf of B is $o(1)$, when $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}$. We delay the proof to Section 2.2.3.5.

Lemma 2.2.16 (Pruning Lemma). $\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{yes}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H})\ \textit{reaches a bad leaf of}\ B\right] = o(1)$.

The pruning lemma allow us to focus on the good leaves ℓ of B only. In particular we know that along the root-to-ℓ path the sets $A_{i,1}$ and $A_{i,j,0}$ each cannot shrink by more than $\alpha\sqrt{n}\log n$ with a single query (otherwise the path contains a bad edge and ℓ is a bad leaf). Moreover every set $P_{i,j}$ in P at the end must remain consistent (either 0-consistent or 1-consistent).

We use these properties to prove the following lemma in Section 2.2.3.4 for good leaves.

Lemma 2.2.17. *For each good leaf ℓ of B, we have*

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{yes}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H})\ \textit{reaches}\ \ell\right] \le (1+o(1))\cdot \Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{no}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H})\ \textit{reaches}\ \ell\right].$$

Combining Lemma 2.2.16 and Lemma 2.2.17, we can prove Lemma 2.2.10 as follow:

Proof of Lemma 2.2.10. Let L be the set of leaves labeled "accept" in B and $L^*\subset L$ be the good leaves among them. Below we hide $(\mathbf{T},\mathbf{C})\sim\mathcal{E}$ in the subscript since it appears in every probability.

$$\begin{aligned}
\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[B \text{ accepts } (\mathbf{T},\mathbf{C},\boldsymbol{H})\right] &= \sum_{\ell\in L}\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\right]\\
&\le \sum_{\ell\in L^*}\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\right] + o(1)\\
&\le (1+o(1))\cdot\sum_{\ell\in L^*}\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\left[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\right] + o(1)\\
&\le \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\left[B \text{ accepts } (\mathbf{T},\mathbf{C},\boldsymbol{H})\right] + o(1),
\end{aligned}$$

where the second line follows from Lemma 2.2.16 and the third line follows from Lemma 2.2.17. □

2.2.3.4 Proof of Lemma 2.2.17 for good leaves

We prove Lemma 2.2.17 in this section. Let ℓ be a good leaf associated with ϕ_ℓ and $(I; J; P; R; A; \rho)$ be the tuple that ϕ_ℓ induces. Note that along the root-to-ℓ path, when the sets $A_{i,0}, A_{i,1}, A_{i,j,0}, A_{i,j,1}$ are created for the first time in A, their sizes are always between $(n/2) \pm \sqrt{n}$ (since all queries made by B lie in the middle layers). As a result, it follows from Definition 2.2.15 that for each $i \in I$ and $j \in J_i$:

i) $|A_{i,1}| \geq (n/2) - O(|P_i| \cdot \sqrt{n} \log n)$ and $|A_{i,j,0}| \geq (n/2) - O(|P_{i,j}| \cdot \sqrt{n} \log n)$;
ii) $|A_{i,0}|, |A_{i,1}|, |A_{i,j,0}|, |A_{i,j,1}| \leq (n/2) + \sqrt{n}$;
iii) $P_{i,j}$ is consistent (either 1-consistent or 0-consistent).

Then we have the following claim:

Claim 2.2.18. *For each $i \in I$ and $j \in J_i$, $|A_{i,j,1}| \geq (n/2) - O\big(|P_{i,j}|^2 \cdot \sqrt{n} \log n\big)$.*

Proof. For any two strings $x, y \in P_{i,j}$, we have

$$\big|\{k \in [n] : x_k = y_k = 0\}\big| \geq |A_{i,j,0}| \geq (n/2) - O\big(|P_{i,j}| \cdot \sqrt{n} \log n\big).$$

As a result, it follows from $|\{k : y_k = 0\}| \leq (n/2) + \sqrt{n}$ and $P_{i,j}$ being nonempty that

$$\big|\{k \in [n] : x_k = 1, y_k = 0\}\big| \leq O\big(|P_{i,j}| \cdot \sqrt{n} \log n\big).$$

Finally we have

$$|A_{i,j,1}| \geq \big|\{k : x_k = 1\}\big| - \sum_{y \in P_{i,j} \setminus \{x\}} \big|\{k : x_k = 1, y_k = 0\}\big| \geq (n/2) - O\big(|P_{i,j}|^2 \cdot \sqrt{n} \log n\big). \tag{2.3}$$

This finishes the proof of the claim. □

Additionally, notice that $A_{i,1} \subseteq A_{i,j,1}$; thus from i) we have

$$|A_{i,j,1}| \geq |A_{i,1}| \geq (n/2) - O\big(|P_i| \cdot \sqrt{n} \log n\big). \tag{2.4}$$

The following claim is an immediate consequence of this fact and Claim 2.2.18.

Claim 2.2.19. *For each $i \in I$ and $j \in J_i$, we have*

$$\big||A_{i,j,1}| - |A_{i,j,0}|\big| \leq O\left(\sqrt{n} \log n \cdot \min\left\{|P_{i,j}|^2, |P_i|\right\}\right)$$

Proof. We have from i) and ii) that

$$|A_{i,j,1}| - |A_{i,j,0}| \leq (n/2) + \sqrt{n} - \big((n/2) - O\big(|P_{i,j}| \cdot \sqrt{n} \log n\big)\big) = O\big(|P_{i,j}| \cdot \sqrt{n} \log n\big).$$

On the other hand, from ii), (2.3) and (2.4), we have

$$|A_{i,j,0}| - |A_{i,j,1}| \leq O\big(\sqrt{n} \log n \cdot \min\left\{|P_{i,j}|^2, |P_i|\right\}\big).$$

Note that $|P_{i,j}| \leq |P_i|$. The lemma then follows. □

We are now ready to prove Lemma 2.2.17.

Proof of Lemma 2.2.17. Let ℓ be a good leaf and let $\phi : Q \to \mathfrak{P}$ be the map associated with ℓ.

Let $|\mathcal{E}|$ denote the support size of $\mathcal{E}$. We are interested in the following two probabilities:

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\Big] = \frac{1}{|\mathcal{E}|}\sum_{(T,C)} \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T,C,\boldsymbol{H}) \text{ reaches } \ell\Big]$$
$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\Big[(\mathbf{T},\mathbf{C},\boldsymbol{H}) \text{ reaches } \ell\Big] = \frac{1}{|\mathcal{E}|}\sum_{(T,C)} \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\Big[(T,C,\boldsymbol{H}) \text{ reaches } \ell\Big],$$

where the sum is over the support of $\mathcal{E}$. Hence, it suffices to show that for each (T, C)

such that

$$\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[(T,C,\boldsymbol{H}) \text{ reaches } \ell\right] > 0, \tag{2.5}$$

we have the following inequality:

$$\frac{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\left[(T,C,\boldsymbol{H}) \text{ reaches } \ell\right]}{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[(T,C,\boldsymbol{H}) \text{ reaches } \ell\right]} \geq 1 - o(1). \tag{2.6}$$

Fix a pair (T, C) such that (2.5) holds. Recall that (T, C, H) reaches ℓ if and only if the signature of each $x \in Q$ with respect to (T, C, H) matches exactly $\phi(x) = (\sigma_x, \tau_x, a_x, b_x)$. Given (2.5), the term and clause signatures of x are already known to match σ_x and τ_x (otherwise the LHS of (2.5) is 0). The rest, i.e., a_x and b_x for each $x \in Q$, depends on $H = (h_{i,j} : i, j \in [N])$ only.

Since ℓ is consistent, there is a $\rho_{i,j} \in \{0, 1\}$ for each $P_{i,j}$ such that every $x \in P_{i,j}$ should satisfy $h_{i,j}(x) = \rho_{i,j}$. These are indeed the only conditions for H to match a_x and b_x for each $x \in Q$, and as a result, below we give the conditions on $H = (h_{i,j})$ for the triple (T, C, H) to reach ℓ:

- For $\mathcal{E}_{\text{yes}}$, (T, C, H) reaches ℓ, where $H = (h_{i,j})$ and $h_{i,j}(x) = x_{k_{i,j}}$, if and only if $k_{i,j} \in A_{i,j,\rho_{i,j}}$ for each $i \in I$ and $j \in J_i$ (so that each $x \in P_{i,j}$ has $h_{i,j}(x) = \rho_{i,j}$).
- For $\mathcal{E}_{\text{no}}$, (T, C, H) reaches ℓ, where $H = (h_{i,j})$ and $h_{i,j}(x) = \overline{x_{k_{i,j}}}$, if and only if $k_{i,j} \in A_{i,j,1-\rho_{i,j}}$ for each $i \in I$ and $j \in J_i$ (so that each $x \in P_{i,j}$ has $h_{i,j}(x) = \rho_{i,j}$).

With this characterization, we can rewrite the LHS of (2.6) as follows:

$$\begin{aligned}\frac{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}\left[(T,C,\boldsymbol{H}) \text{ reaches } \ell\right]}{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\left[(T,C,\boldsymbol{H}) \text{ reaches } \ell\right]} &= \prod_{i\in I, j\in J_i}\left(\frac{|A_{i,j,1-\rho_{i,j}}|}{|A_{i,j,\rho_{i,j}}|}\right) \\ &= \prod_{i\in I, j\in J_i}\left(1 + \frac{|A_{i,j,1-\rho_{i,j}}| - |A_{i,j,\rho_{i,j}}|}{|A_{i,j,\rho_{i,j}}|}\right).\end{aligned}$$

Thus, by applying Claim 2.2.19 and noting that $|A_{i,j,\rho_{i,j}}| \leq n$ (whether $\rho_{i,j} = 0$ or 1) we

have

$$\frac{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{no}}}[(T,C,\boldsymbol{H}) \text{ reaches } \ell]}{\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}[(T,C,\boldsymbol{H}) \text{ reaches } \ell]} \geq \prod_{i\in I, j\in J_i}\left(1 - O\left(\frac{\log n \cdot \min\{|P_{i,j}|^2, |P_i|\}}{\sqrt{n}}\right)\right)$$
$$\geq 1 - O\left(\frac{\log n}{\sqrt{n}}\right)\sum_{i\in I, j\in J_i}\min\left\{|P_{i,j}|^2, |P_i|\right\}.$$

As $\sum_j |P_{i,j}| \leq 2|P_i|$, $\sum_{j\in J_i}\min\left\{|P_{i,j}|^2, |P_i|\right\}$ is maximized if $|J_i| = 2\sqrt{|P_i|}$ and $|P_{i,j}| = \sqrt{|P_i|}$. Therefore we have

$$\sum_{i\in I, j\in J_i}\min\left\{|P_{i,j}|^2, |P_i|\right\} \leq \sum_{i\in I} 2|P_i|^{3/2} \leq O(q^{3/2}),$$

since $\sum_i |P_i| \leq 2q$. This finishes the proof of the lemma since q is chosen to be $n^{1/3}/\log^2 n$.

□

2.2.3.5 Proof of the pruning lemma

To complete our proof for the lower bound of monotonicity testing, we prove the pruning lemma, Lemma 2.2.16, in this section.

Let E be the set of bad edges as defined in Definition 2.2.15 (recall that if (u, v) is a bad edge, then the root-to-u path cannot have any bad edge). We split the proof of Lemma 2.2.16 into four lemmas, one lemma for each type of bad edges. To this end, we define four sets of bad edges E_1, E_2, E_3 and E_4 (we follow the same notation of Definition 2.2.15): an edge $(u, v) \in E$ belongs to

1. E_1 if $|A_{i,1} \setminus A'_{i,1}| \geq \alpha\sqrt{n}\log n$ for some $i \in I$;
2. E_2 if $|A_{i,j,0} \setminus A'_{i,j,0}| \geq \alpha\sqrt{n}\log n$ for some $i \in I$ and $j \in J_i$;
3. E_3 if it is not in E_2 and for some $i \in I$ and $j \in J_i$, $P_{i,j}$ is 0-consistent but $P'_{i,j}$ is inconsistent. When this happens we say (u, v) is E_3-bad at (i, j);
4. E_4 if it is not in E_1 or E_2 and for some $i \in I$ and $j \in J_i$, $P_{i,j}$ is 1-consistent but $P'_{i,j}$

is inconsistent. When this happens we say (u, v) is E_4-bad at (i, j).

It is clear that $E = E_1 \cup E_2 \cup E_3 \cup E_4$. (These four sets are not necessarily pairwise disjoint though we did exclude edges of E_2 from E_3 and edges of E_1 and E_2 from E_4 explicitly.) Each lemma below states that the probability of $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}$ taking an edge in E_i is $o(1)$. Lemma 2.2.16 then follows directly from a union bound over the four sets.

Lemma 2.2.20. *The probability of $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{yes}$ taking an edge in E_1 is $o(1)$.*

Proof. Let u be an internal node. We prove that, when $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ and $\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}$, either $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ reaches node u with probability 0 or

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(\mathbf{T}, \mathbf{C}, \boldsymbol{H}) \text{ takes an } E_1\text{-edge at } u \,\Big|\, (\mathbf{T}, \mathbf{C}, \boldsymbol{H}) \text{ reaches } u\Big] = o(1/q). \quad (2.7)$$

Lemma 2.2.20 then follows from Lemma 2.1.3. Below we assume that the probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ reaching node u is positive. Let $\phi : Q \to \mathfrak{P}$ be the map associated with u, and let $x \in \{0, 1\}^n$ be the string queried at u. Whenever we discuss a child node v of u below, we use $\phi' : Q \cup \{x\} \to \mathfrak{P}$ to denote the map associated with v and $(I; J; P; R; A; \rho)$ and $(I'; J'; P'; R'; A'; \rho')$ to denote the tuples ϕ and ϕ' induce. (Note that v is not a specific node but can be any child of u.)

Fix an $i \in I$. We upperbound by $o(1/q^2)$ the conditional probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ following an edge (u, v) with $|A_{i,1} \setminus A'_{i,1}| \geq \alpha\sqrt{n}\log n$. (2.7) follows directly from a union bound over $i \in I$.

With i fixed, observe that any edge (u, v) has either $A'_{i,1} = A_{i,1}$ or $A'_{i,1} = A_{i,1} \setminus \Delta_i$ with

$$\Delta_i = \big\{\ell \in A_{i,1} : x_\ell = 0\big\} \subseteq A_{i,1}.$$

The latter happens if and only if $P'_i = P_i \cup \{x\}$. Therefore, we assume WLOG that $|\Delta_i| \geq \alpha\sqrt{n}\log n$ (otherwise the conditional probability is 0 for i), and now it suffices

to upperbound by $o(1/q^2)$ the conditional probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ taking an edge (u, v) with $P_i' = P_i \cup \{x\}$.

To analyze this conditional probability for $i \in I$, we fix a triple (T_{-i}, C, H), where we use T_{-i} to denote a sequence of $N-1$ terms with only the ith term missing, such that

$$\Pr_{\mathbf{T}_i}\left[((T_{-i}, \mathbf{T}_i), C, H) \text{ reaches } u\right] > 0,$$

where $\mathbf{T}_i$ is a term drawn uniformly at random. It suffices to prove for any fixed such (T_{-i}, C, H):

$$\begin{aligned} &\Pr_{\mathbf{T}_i}\left[((T_{-i}, \mathbf{T}_i), C, H) \text{ reaches } u \text{ and } P_i' = P_i \cup \{x\}\right] \\ \leq\ & o(1/q^2) \cdot \Pr_{\mathbf{T}_i}\left[((T_{-i}, \mathbf{T}_i), C, H) \text{ reaches } u\right]. \end{aligned} \tag{2.8}$$

Recalling Fact 2.2.12, the latter event, $((T_{-i}, \mathbf{T}_i), C, H)$ reaching u, imposes two conditions on $\mathbf{T}_i$:

1. For each $y \in P_i$, $\mathbf{T}_i(y) = 1$, and
2. For each $z \in R_i$, $\mathbf{T}_i(z) = 0$.

Let U denote the set of all such terms $T : \sqrt{n} \to [n]$. Then equivalently $T \in U$ if and only if

$$U\text{: } T(k) \in A_{i,1} \text{ for all } k \in [\sqrt{n}] \text{ and each } z \in R_i \text{ has } z_{T(k)} = 0 \text{ for some } k \in [\sqrt{n}].$$

Regarding the former event in (2.8), i.e. $((T_{-i}, \mathbf{T}_i), C, H)$ reaching u and $P_i' = P_i \cup \{x\}$, a necessary condition over $\mathbf{T}_i$ is the same as above but in addition we require $\mathbf{T}_i(x) = 1$. (Note that this is not a sufficient condition since for that we also need $\mathbf{T}_i$ to be one of the first two terms that x satisfies, which depends on T_{-i}.) Let V denote the set of all such terms. Then $T \in V$ if

V: $T(k) \in A_{i,1} \setminus \Delta_i$ for all $k \in [\sqrt{n}]$ and each $z \in R_i$ has $z_{T(k)} = 0$ for some $k \in [\sqrt{n}]$.

In the rest of the proof we prove that $|V|/|U| = o(1/q^2)$, from which (2.8) follows. Let $m = \log n$. First we write U' to denote the following subset of U: $T' \in U$ is in U' if

$$\left|\{k \in [\sqrt{n}] : T'(k) \in \Delta_i\}\right| = m,$$

and it suffices to show $|V|/|U'| = o(1/q^2)$. Next we define the following bipartite graph G between U' and V (inspired by similar arguments of [BB16]): $T' \in U'$ and $T \in V$ have an edge if and only if $T'(k) = T(k)$ for all $k \in [\sqrt{n}]$ with $T'(k) \notin \Delta_i$. Each $T' \in U'$ has degree at most $|A_{i,1} \setminus \Delta_i|^m$, as one can only move each $T'(k) \in \Delta_i$ to $A_{i,1} \setminus \Delta_i$.

To lowerbound the degree of a $T \in V$, note that one only needs at most q many variables of T to kill all strings in R_i (such that they don't satisfy T). Let $H \subset [\sqrt{n}]$ be any set of size at most q such that for each string $z \in R_i$, there exists a $k \in H$ with $z_{T(k)} = 0$.[6] Then one can choose any m distinct indices $k_1, \ldots, k_\ell$ from $\overline{H}$, as well as any m (not necessarily distinct) variables $t_1, \ldots, t_\ell$ from Δ_i, and let T' be a term where

$$T'(k) = \begin{cases} t_i & k = k_i \text{ for some } i \in [m] \\ T(k) & \text{otherwise.} \end{cases}$$

The resulting T' is in U' and (T, T') is an edge in G. As a result, the degree of $T \in V$ is at least

$$\binom{\sqrt{n} - q}{m} \cdot |\Delta_i|^m.$$

By counting the number of edges in G from both sides and using $|A_{i,1}| \leq (n/2) + \sqrt{n}$,

$$\frac{|U'|}{|V|} \geq \binom{\sqrt{n} - q}{m} \cdot \left(\frac{|\Delta_i|}{|A_{i,1} \setminus \Delta_i|}\right)^m \geq \left(\frac{\sqrt{n}}{2m} \cdot \frac{\alpha\sqrt{n}m}{(n/2) + \sqrt{n}}\right)^m > \omega(q^2),$$

[6] For example, since $|R_i| \leq q$, one can set H to contain the smallest $k \in [\sqrt{n}]$ such that $z_{T(k)} = 0$, for each $z \in R_i$.

by choosing a large enough constant $\alpha > 0$. This finishes the proof of the lemma. □

Lemma 2.2.21. *The probability of* $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ *and* $\boldsymbol{H} \sim \mathcal{E}_{yes}$ *taking an edge in* E_2 *is* $o(1)$.

Proof. The proof of this lemma is similar to that of Lemma 2.2.20. Let u be any internal node of the tree. We prove that, when $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$, $\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}$, either $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ reaches u with probability 0 or

$$\Pr_{(\mathbf{T},\mathbf{C})\sim\mathcal{E},\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(\mathbf{T}, \mathbf{C}, \boldsymbol{H}) \text{ takes an } E_2\text{-edge at } u \;\Big|\; (\mathbf{T}, \mathbf{C}, \boldsymbol{H}) \text{ reaches } u\Big] = o(1/q). \quad (2.9)$$

Assume below WLOG that the probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ reaching u is positive.

Fix $i \in I$ and $j \in J_i$. We upperbound the conditional probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ taking an edge (u, v) with $|A_{i,j,0} \setminus A'_{i,j,0}| \geq \alpha\sqrt{n}\log n$ by $o(1/q^3)$. (2.9) then follows by a union bound over $i \in I$ and $j \in J_i$. Similarly let

$$\Delta_{i,j} = \big\{\ell \in A_{i,j,0} : x_\ell = 1\big\} \subseteq A_{i,j,0}, \quad (2.10)$$

and assume WLOG that $|\Delta_{i,j}| \geq \alpha\sqrt{n}\log n$ (as otherwise the conditional probability is 0 for i, j). Then it suffices to upperbound the conditional probability of $(\mathbf{T}, \mathbf{C}, \boldsymbol{H})$ going along an edge (u, v) with $P'_{i,j} = P_{i,j} \cup \{x\}$ by $o(1/q^3)$. The rest of the proof is symmetric to that of Lemma 2.2.20. □

Lemma 2.2.22. *The probability of* $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ *and* $\boldsymbol{H} \sim \mathcal{E}_{yes}$ *taking an edge in* E_3 *is* $o(1)$.

Proof. We fix any pair (T, C) from the support of $\mathcal{E}$ and prove that

$$\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\big[(T, C, \boldsymbol{H}) \text{ takes an } E_3\text{-edge}\big] = o(1). \quad (2.11)$$

The lemma follows by averaging (2.11) over all pairs (T, C) in the support of $\mathcal{E}$. To prove (2.11) we fix any internal node u such that the probability of $(T, C, \boldsymbol{H})$ reaching u is

positive, and prove that

$$\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T,C,\boldsymbol{H}) \text{ takes an } E_3\text{-edge leaving } u \;\Big|\; (T,C,\boldsymbol{H}) \text{ reaches } u\Big] = o(1/q). \quad (2.12)$$

(2.11) then follows by Lemma 2.1.3. Below we assume the probability of $(T,C,\boldsymbol{H})$ reaching u is positive.

We assume WLOG that there is no edge in E along the root-to-u path; otherwise, (2.12) is 0 (recall Definition 2.2.15). We follow the same notation used in the proof of Lemma 2.2.20, i.e., $\phi_u : Q \to \mathfrak{P}$ as the map associated with u, x as the query made at u, and $(I; J; P; R; A; \rho)$ as the tuple induced by ϕ_u. We also write F to denote the set of pairs (i,j) such that $i \in I$ and $j \in J_i$.

Observe that since (T,C) is fixed, the term and clause signatures of every string are fixed, and in particular the term and clause signatures (denoted σ_x and τ_x) of x are fixed. We assume WLOG that $\sigma_x = e_k$ for some $k \in [N]$ (otherwise x will never be added to any $P_{i,j}$ when $(T,C,\boldsymbol{H})$ leaves u and (2.12) is 0 by the definition of E_3). In this case we write D to denote the set of $\{(k,j) : \tau_{x,j} = 0\}$ with $|D| \le 2$. As a result, whenever $(T,C,\boldsymbol{H})$ takes an E_3-edge leaving from u, this edge must be E_3-bad at one of the pairs $(k,j) \in D$. Thus, the LHS of (2.12) is the same as

$$\sum_{(k,j)\in D} \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T,C,\boldsymbol{H}) \text{ takes a } (u,v) \text{ that is } E_3\text{-bad at } (k,j) \;\Big|\; (T,C,\boldsymbol{H}) \text{ reaches } u\Big]. \quad (2.13)$$

To bound the conditional probability for (k,j) above by $o(1/q)$, we assume WLOG that $(k,j) \in F$ (otherwise x would create a new $P_{k,j}$ whenever $(T,C,\boldsymbol{H})$ takes an edge (u,v) leaving u, and such an edge cannot be E_3-bad at (k,j)). Next we define ($A_{k,j,0}$ below is well defined since $(k,j) \in F$)

$$\Delta_{k,j} = \big\{\ell \in A_{k,j,0} : x_\ell = 1\big\}.$$

We may assume WLOG that $|\Delta_{k,j}| < \alpha\sqrt{n}\log n$; otherwise $(T, C, \boldsymbol{H})$ can never take an edge (u, v) in E_3 because E_2-edges are explicitly excluded from E_3. Finally, we assume WLOG $\rho_{k,j}(y) = 0$ for all $y \in P_{k,j}$; otherwise the edge (u, v) that $(T, C, \boldsymbol{H})$ takes cannot be E_3-bad at (k, j).

With all these assumptions on (k, j) in place, we prove the following inequality:

$$\begin{aligned} &\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T, C, \boldsymbol{H}) \text{ takes a } (u, v) \text{ that is } E_3\text{-bad at } (k, j)\Big] \\ &\le \frac{|\Delta_{k,j}|}{|A_{k,j,0}|} \cdot \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T, C, \boldsymbol{H}) \text{ reaches } u\Big]. \end{aligned} \tag{2.14}$$

Given $|\Delta_{k,j}| = O(\sqrt{n}\log n)$ and $|A_{i,j,0}| \ge (n/2) - O(q\sqrt{n}\log n) = \Omega(n)$ (since there is no bad edge particularly no E_2-edge, from the root to u), (2.12) follows by summing over D, with $|D| \le 2$.

We work on (2.14) in the rest of the proof. Fix any tuple $H_{-(k,j)}$ (with its entry $h_{k,j}$ missing) such that the probability of $(T, C, (H_{-(k,j)}, \boldsymbol{h}))$ reaching u is positive, where $\boldsymbol{h}(x) = x_{\mathbf{r}}$ is a random dictator function with the index $\mathbf{r}$ drawn from $[n]$ uniformly. Then (2.14) follows from

$$\begin{aligned} &\Pr_{\boldsymbol{h}}\Big[(T, C, (H_{-(k,j)}, \boldsymbol{h})) \text{ takes } (u, v) \text{ that is } E_3\text{-bad at } (k, j)\Big] \\ &\le \frac{|\Delta_{k,j}|}{|A_{k,j,0}|} \cdot \Pr_{\boldsymbol{h}}\Big[(T, C, (H_{-(k,j)}, \boldsymbol{h})) \text{ reaches } u\Big]. \end{aligned} \tag{2.15}$$

The event on the RHS, i.e., that $(T, C, (H_{-(k,j)}, \boldsymbol{h}))$ reaches u, imposes the following condition on the hidden index $\mathbf{r}$ associated with $\boldsymbol{h}$ ($\boldsymbol{h}(x) = x_{\mathbf{r}}$): $\mathbf{r} \in A_{k,j,0}$, since $\rho_{k,j}(y) = 0$ for all $y \in P_{k,j}$. Hence the probability on the RHS of (2.15) is $|A_{i,j,0}|/n$. On the other hand, the event on the LHS of (2.15), that $(T, C, (H_{-(i,j)}, \boldsymbol{h}))$ follows a (u, v) that is E_3-bad at (k, j), imposes the following necessary condition for $\mathbf{r}$: $\mathbf{r} \in \Delta_{k,j}$.[7] As a result, the

[7] Note that this is *not* a sufficient condition, because the other pair $(k, j') \in D$ may have $|\Delta_{k,j'}| \ge \alpha\sqrt{n}\log n$.

probability on the LHS of (2.15) is at most $|\Delta_{k,j}|/n$. (2.15) then follows. □

Lemma 2.2.23. *The probability of* $(\mathbf{T}, \mathbf{C}) \sim \mathcal{E}$ *and* $\boldsymbol{H} \sim \mathcal{E}_{yes}$ *taking an edge in* E_4 *is* $o(1)$.

Proof. We fix a pair (T, C) from the support of $\mathcal{E}$ and prove that

$$\Pr_{\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}} \left[(T, C, \boldsymbol{H}) \text{ takes an } E_4\text{-edge} \right] = o(1). \tag{2.16}$$

The lemma follows by averaging (2.16) over all (T, C) in the support of $\mathcal{E}$. To prove (2.16), fix a leaf ℓ such that the probability of $(T, C, \boldsymbol{H})$ reaching ℓ is positive. Let $u_1 \cdots u_{t'} u_{t'+1} = \ell$ be the root-to-ℓ path and let $q(u_s), s \in [t']$ denote the following conditional probability:

$$\Pr_{\boldsymbol{H} \sim \mathcal{E}_{\text{yes}}} \left[(T, C, \boldsymbol{H}) \text{ takes an } E_4\text{-edge leaving } u_s \,\middle|\, (T, C, \boldsymbol{H}) \text{ reaches } u_s \right].$$

It then suffices to show for every such leaf ℓ,

$$\sum_{s \in [t']} q(u_s) = o(1), \tag{2.17}$$

since (2.16) would then follow by Lemma 2.1.4. To prove (2.17), we use t to denote the smallest integer such that (u_{t+1}, u_{t+2}) is an edge in E_1 or E_2 with $t = t'$ by default if there is no such edge along the path. By the choice of t, there is no edge in E_1 or E_2 along $u_1 \cdots u_{t+1}$. For (2.17) it suffices to show

$$\sum_{s \in [t]} q(u_s) = o(1). \tag{2.18}$$

To see this we consider two cases. If there is no E_1, E_2 edge along the root-to-ℓ path, then the two sums in (2.17) and (2.18) are the same. If (u_{t+1}, u_{t+2}) is an edge in E_1 or E_2, then $q(u_s) = 0$ if $s \geq t + 2$ (since $(u, v) \notin E$ if there is already an edge in E along the path to u). We claim that $q(u_{t+1})$ must be 0 as well. This is because, given that (T, C) is fixed and that $(T, C, \boldsymbol{H})$ takes (u_{t+1}, u_{t+2}) with a positive probability, whenever $(T, C, \boldsymbol{H})$

follows an edge (u_{t+1}, v) from u_{t+1}, v has the same term and clause signatures (σ_x, τ_x) as u_{t+2} and thus, also has the same P and A (as part of the tuple its map induces). As a result (u_{t+1}, v) is also in E_1 or E_2 and cannot be an edge in E_4 (recall that we explicitly excluded E_1 and E_2 from E_4). Below we focus on u_s with $s \in [t]$ and upperbound $q(u_s)$.

For each $s \in [t]$ we write x^s to denote the string queried at u_s and let $(I^s; J^s; P^s; Q^s; R^s; \rho^s)$ be the tuple induced by the map associated with u_s. We also write F_s to denote the set of pairs (i, j) with $i \in I^s, j \in J^s_i$. Following the same arguments used to derive (2.13) in the proof of Lemma 2.2.22, let $D_s \subseteq F_s$ denote the set of at most two pairs (i, j) such that x^s is added to $P^s_{i,j}$ when $(T, C, \boldsymbol{H})$ reaches u_s. Note that if x^s just creates a new $P_{i,j}$ (so $(i, j) \notin F_s$), $(T, C, \boldsymbol{H})$ won't take an E_4-edge in this case and we do not include it in D_s. As a result, whenever $(T, C, \boldsymbol{H})$ takes an E_4-edge (u, v), the latter must be E_4-bad at one of $(i, j) \in D_s$.

Next for each pair $(i, j) \in D_s$, we can follow the analysis of (2.14) to show that

$$\begin{aligned}&\Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T, C, \boldsymbol{H}) \text{ takes a } (u, v) \text{ that is } E_4\text{-bad at } (i, j)\Big]\\ \le\ &\frac{|\Delta^s_{i,j}|}{|A^s_{i,j,1}|} \cdot \Pr_{\boldsymbol{H}\sim\mathcal{E}_{\text{yes}}}\Big[(T, C, \boldsymbol{H}) \text{ reaches } u\Big],\end{aligned}$$

where the set $\Delta^s_{i,j}$ is defined as $\Delta^s_{i,j} = \Big\{k \in A^s_{i,j,1} : x^s_k = 0\Big\}$.

As there is no E_1 or E_2 edge along the path to u_s, we have by (2.4) that $A^s_{i,j,1}$ has size $\Omega(n)$. Therefore we know

$$q(u_s) \le O(1/n) \cdot \sum_{(i,j)\in D_s} \left|\Delta^s_{i,j}\right| \quad \text{and} \quad \sum_{s\in[t]} q(u_s) \le O(1/n) \cdot \sum_{s\in[t]} \sum_{(i,j)\in D_s} \left|\Delta^s_{i,j}\right|. \tag{2.19}$$

Let $(I^*; J^*; P^*; R^*; A^*; \rho^*)$ be the tuple induced by the map associated with u_{t+1} and let F^* be the set of (i, j) with $i \in I^*$ and $j \in J^*_i$. We upperbound the second sum in (2.19)

above by focusing on any fixed pair $(i,j) \in F^*$ and observing that

$$\left|A^*_{i,j,1}\right| + \sum_{s:(i,j)\in D_s} \left|\Delta^s_{i,j}\right| \leq (n/2) + \sqrt{n}.$$

This is because each $\Delta^s_{i,j}$ and $A^*_{i,j,1}$ are pairwise disjoint and their union is indeed exactly the number of 1-entries of the query string along the path that first creates $P_{i,j}$. The latter is at most $(n/2)+\sqrt{n}$ because we assumed that strings queried in the tree lie in the middle layers. On the other hand,

$$\left|A^*_{i,j,1}\right| \geq (n/2) - O\big(\sqrt{n}\log n \cdot \min\left\{|P^*_{i,j}|^2, |P^*_i|\right\}\big).$$

This follows directly from (2.3) and (2.4) and our choice of t at the beginning of the proof so that there is no E_1 or E_2 edge from u_1 to u_{t+1}. We finish the proof by plugging the two inequalities into (2.19) and follow the same arguments used at the end of the proof of the lemma for good leaves. □

2.3 An $\tilde{\Omega}(n^{2/3})$ lower bound for testing of unateness

In this section we will show any (adaptive) algorithm for unateness testing must make at least $\tilde{\Omega}(n^{2/3})$ queries (for some constant distance parameter ϵ) and prove Theorem 1.2.4. The proof follows Yao's mini-max priciple and the same idea from our lower bound proof for monotonicity testing, with some adaptation to the case of unateness testing. We will give some more intuition how we can push the lower bound of $\tilde{\Omega}(n^{1/3})$ for monotonicity testing further to $\tilde{\Omega}(n^{2/3})$ after we define the new hard distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ for the proof in this section.

We will present our distributions $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ in Section 2.3.1, and prove it is hard to distinguish them with $\tilde{O}(n^{2/3})$ queries in Section 2.3.2 to complete the proof.

2.3.1 Distributions

In this section we describe a pair of distributions, $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$, supported on Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$ that will be used to obtain an adaptive lower bound for unateness testing by applying Yao's mini-max principle. In order to apply this principle we also show in this section that any $\mathbf{f} \sim \mathcal{D}_{yes}$ is unate, and $\mathbf{f} \sim \mathcal{D}_{no}$ is $\Omega(1)$-far from being unate with probability $\Omega(1)$. Let N be the following parameter:

$$N = \left(1 + \frac{1}{\sqrt{n}}\right)^{n/4} \approx e^{\sqrt{n}/4}.$$

A function $\mathbf{f} \sim \mathcal{D}_{yes}$ is drawn using the following procedure:

1. Sample a subset $\mathbf{M} \subset [n]$ uniformly at random from all subsets of size $n/2$.
2. Sample $\mathbf{T} \sim \mathcal{E}(\mathbf{M})$, which is a sequence of terms $(\mathbf{T}_i : i \in [N])$. $\mathbf{T}$ is then used to define a multiplexer map $\mathbf{\Gamma} = \mathbf{\Gamma}_{\mathbf{T}}\colon \{0,1\}^n \to [N] \cup \{0^*, 1^*\}$. Both definitions of $\mathcal{E}(\mathbf{M})$ and $\mathbf{\Gamma}_{\mathbf{T}}$ will be described next.
3. Sample $\mathbf{H} = (\boldsymbol{h}_i : i \in [N]) \sim \mathcal{E}_{\text{yes}}(\mathbf{M})$ according to the follows: for each $i \in [N]$, $\boldsymbol{h}_i\colon \{0,1\}^n \to \{0,1\}$ is a dictatorship function $\boldsymbol{h}_i(x) = x_{\boldsymbol{k}_i}$ with $\boldsymbol{k}_i$ sampled independently and uniformly from $\overline{\mathbf{M}}$. We will refer to $\boldsymbol{h}_i$ as the dictatorship function and $x_{\boldsymbol{k}_i}$ (or simply its index $\boldsymbol{k}_i$) as the *special* variable associated with the ith term $\mathbf{T}_i$.
4. Sample two strings $\mathbf{r} \in \{0,1\}^{\mathbf{M}}$ and $\mathbf{s} \in \{0,1\}^{\mathbf{M}}$ uniformly at random. Finally, we define $\mathbf{f} = \mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}}\colon \{0,1\}^n \to \{0,1\}$ as $\mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}}(x) = f_{\mathbf{M},\mathbf{T},\mathbf{H}}\big(x \oplus (\mathbf{r} \circ \mathbf{s})\big)$, where $\mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H}}$ is defined as follows (with the truncation done first):

$$\mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H}}(x) = \begin{cases} 0 & \text{if } |x_{\mathbf{M}}| < (n/4) - \sqrt{n} \\ 1 & \text{if } |x_{\mathbf{M}}| > (n/4) + \sqrt{n} \\ 0 & \text{if } \mathbf{\Gamma}(x) = 0^* \\ 1 & \text{if } \mathbf{\Gamma}(x) = 1^* \\ h_{\mathbf{\Gamma}(x)}(x) & \text{otherwise (i.e., when } \mathbf{\Gamma}(x) \in [N]) \end{cases}$$

A function $\mathbf{f} = \mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}} \sim \mathcal{D}_{no}$ is drawn using a similar procedure, with the only difference being that $\mathbf{H} = (\boldsymbol{h}_i : i \in [N])$ is sampled from $\mathcal{E}_{\text{no}}(\mathbf{M})$ instead of $\mathcal{E}_{\text{yes}}(\mathbf{M})$: each $\boldsymbol{h}_i$ is a dictatorship function $\boldsymbol{h}_i(x) = x_{\boldsymbol{k}_i}$ with probability $1/2$ and an anti-dictatorship $\boldsymbol{h}_i(x) = \overline{x_{\boldsymbol{k}_i}}$ with probability $1/2$, where $\boldsymbol{k}_i$ is chosen independently and uniformly at random from $\overline{\mathbf{M}}$. We will also refer to $\boldsymbol{h}_i$ as the dictatorship or anti-dictatorship function and $x_{\boldsymbol{k}_i}$ as the special variable associated with $\mathbf{T}_i$.

Remark 2. *Note that the truncation in* $\mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}}$ *is done after sampling* $\mathbf{r}$*. As a result, we may* not *assume all queries are made in the middle layers, like we did in Section 2.2.*

Fixing an $M \subset [n]$ of size $n/2$, we now describe $\mathbf{T} \sim \mathcal{E}(M)$ and $\boldsymbol{\Gamma}_{\mathbf{T}} \colon \{0,1\}^n \to [N] \cup \{0^*, 1^*\}$ to finish the description of the two distributions. Each term $\mathbf{T}_i$ in $\mathbf{T}$, $i \in [N]$, is drawn independently and is a random *subset* of M with each $j \in M$ included with probability $1/\sqrt{n}$ independently. We also abuse the notation and interpret each term $\mathbf{T}_i$ as a Boolean function that is the conjunction of its variables:

$$\mathbf{T}_i(x) = \bigwedge_{j \in \mathbf{T}_i} x_j.$$

Note that, for some technical reason that will become clear later in the proof of Lemma 2.3.21, the definition of terms here is slightly different from that used in the monotonicity lower bound, though both are the conjunction of roughly $\sqrt{n}/2$ ($\sqrt{n}$ in monotonicity) variables. Given $\mathbf{T}$, the multiplexer map $\boldsymbol{\Gamma}_{\mathbf{T}} \colon \{0,1\}^n \to [N] \cup \{0^*, 1^*\}$ indicates the index of the term $\mathbf{T}_i$ that is satisfied by x, if there is a unique one; it returns 0^* if no term is satisfied, or 1^* if more than one terms are satisfied:

$$\boldsymbol{\Gamma}_{\mathbf{T}}(x) = \begin{cases} 0^* & \forall\, i \in [N],\ \mathbf{T}_i(x) = 0 \\ 1^* & \exists\, i \neq j \in [N],\ \mathbf{T}_i(x) = \mathbf{T}_j(x) = 1 \\ i & \mathbf{T}_i(x) = 1 \text{ for a unique } i \in [N] \end{cases}$$

We give some intuition for the reason why the two distributions are hard to distinguish and can be used to obtain a much better lower bound for unateness testing, despite of being much simpler than the two-level construction used in the previous section. Note that $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$ are exactly the same except that (1) in $\mathcal{D}_{yes}$, $\boldsymbol{h}_i$'s are random dictatorship or anti-dictatorship functions (if one takes $\mathbf{s}$ into consideration) but are *consistent* in the sense that all $\boldsymbol{h}_i$'s with the same special variable are either all dictatorship or anti-dictatorship functions; (2) in contrast, whether $\boldsymbol{h}_i$ is a dictatorship or anti-dictatorship is independent for each $i \in [N]$ in $\mathcal{D}_{no}$. Informally, the only way for an algorithm to be sure that f is from $\mathcal{D}_{no}$ (instead of $\mathcal{D}_{yes}$) is to find two terms with the same special variable x_k (for some $k \in [n]$) but one with a dictatorship and the other with an anti-dictatorship function over x_k. As a result, one can interpret our $\tilde{\Omega}(n^{2/3})$ lower bound (at a high level) as the product of two quantities: the number of queries one needs to *breach* a term $\mathbf{T}_i$ (see Section 2.3.2.2 for details) and find its special variable, and the number of terms one needs to breach in order to find two with the same special variable. This is different from monotonicity testing since we are done once a term is breached there, and enables us to obtain a much better lower bound for unateness testing.

Next we prove that $\mathbf{f} \sim \mathcal{D}_{yes}$ is unate and $\mathbf{f} \sim \mathcal{D}_{no}$ is far from unate with high probability.

Lemma 2.3.1. *Every f in the support of $\mathcal{D}_{yes}$ is unate.*

Proof. Given the definition of $f = f_{M,T,H,r,s}$ using $f_{M,T,H}$, it suffices to show that $f_{M,T,H}$ is monotone. The rest of the proof is similar to that of Lemma 2.2.1. □

Lemma 2.3.2. *A function $\mathbf{f} \sim \mathcal{D}_{no}$ is $\Omega(1)$-far from unate with probability $\Omega(1)$.*

Proof. Consider a fixed subset $M \subset [n]$ of size $n/2$. It suffices to prove that, when $\mathbf{T} \sim \mathcal{E}(M)$ and $\mathbf{H} \sim \mathcal{E}_{\text{no}}(M)$, the function $\mathbf{f} = \mathbf{f}_{M,\mathbf{T},\mathbf{H}}$ is $\Omega(1)$-far from unate. This is due to the fact that flipping variables of a function (as we apply EXLUSIVE-OR between input strings with $\mathbf{r} \circ \mathbf{s}$) does not change its distance to unateness.

Fix T in the support of $\mathcal{E}(M)$ and H in the support of $\mathcal{E}_{\text{no}}(M)$. We let $X \subset \{0,1\}^n$ denote the set of $x \in \{0,1\}^n$ in the middle layers (i.e. $|x_M|$ is within $n/4 \pm \sqrt{n}$) such that $\Gamma_T(x) = i$ for some $i \in [N]$ (rather than 0^* or 1^*). For each $x \in X$ with $\Gamma_T(x) = i$, we also let $\rho(x) = k$ be the special variable associated with T_i (i.e., $h_i(x) = x_k$ or $h_i(x) = \overline{x_k}$). As $\rho(x) \in \overline{M}$ and $\Gamma_T(x)$ depends only on variables in M, we have that

$$\Gamma_T\big(x^{(\rho(x))}\big) = \Gamma_T(x),$$

i.e., after flipping the $\rho(x)$th bit of x, the new string still uniquely satisfies the same term as x.

Let $x^* = x^{(\rho(x))}$ for each string $x \in X$. Then it's easy to see $x^* \in X$ and $(x^*)^* = x$. The claim below shows that (x, x^*) is a bi-chromatic edge along the $\rho(x)$th direction. As a result, one can decompose $|X|$ into $|X|/2$ many *disjoint* bi-chromatic edges (x, x^*).

Claim 2.3.3. *For all $x \in X$, (x, x^*) is a bi-chromatic edge of $f_{M,T,H}$.*

Proof. Let $k = \rho(x) \in \overline{M}$. Then $f_{M,T,H}(x)$ and $f_{M,T,H}(x^*)$ are either x_k and x_k^*, or $\overline{x_k}$ and $\overline{x_k^*}$. The claim follows directly from $x^* = x^{(k)}$ and thus, $x_k^* = \overline{x_k}$. □

For each $k \in \overline{M}$, we partition strings $x \in X$ with $\rho(x) = k$ and $f(x) = 0$ into

$$X_k^+ = \big\{x \in X : \rho(x) = k,\, x_k = 0,\, f(x) = 0\big\}$$
$$\text{and} \quad X_k^- = \big\{x \in X : \rho(x) = k,\, x_k = 1,\, f(x) = 0\big\}.$$

Note that for each $x \in X_k^+$, (x, x^*) is a monotone (bi-chromatic) edge; for each $x \in X_k^-$, (x, x^*) is an anti-monotone edge. Since all these $|X|/2$ edges are disjoint, by Lemma 2.1.2 we have:

$$dist\big(f_{M,T,H}, \textsc{Unate}\big) \geq \frac{1}{2^n} \cdot \sum_{k \in \overline{M}} \min\big\{|X_k^+|, |X_k^-|\big\}.$$

Therefore, it suffices to show that with probability $\Omega(1)$ over $\mathbf{T} \sim \mathcal{E}(M)$ and $\mathbf{H} \sim \mathcal{E}_{\text{no}}(M)$, both $\mathbf{X}_k^+$ and $\mathbf{X}_k^-$ (as random variables derived from $\mathbf{T}$ and $\mathbf{H}$) have size $\Omega(2^n/n)$ for every $k \in \overline{M}$.

To simplify the proof we introduce a new distribution $\mathcal{E}'(M)$ that is the same as $\mathcal{E}(M)$ but conditioned on that every T_i in T contains at least $n^{1/3}$ elements. Our goal is to show that

$$\Pr_{\mathbf{T}\sim\mathcal{E}'(M),\,\mathbf{H}\sim\mathcal{E}_{\text{no}}(M)}\left[\forall k \in \overline{M}, \text{ both } \mathbf{X}_k^+ \text{ and } \mathbf{X}_k^- \text{ have size } \Omega(2^n/n)\right] = \Omega(1). \tag{2.20}$$

This implies the desired claim over $\mathbf{T} \sim \mathcal{E}(M)$ as the probability of $\mathbf{T} \sim \mathcal{E}(M)$ lying in the support of $\mathcal{E}'(M)$ is at least $1 - \exp\left(-\Omega(\sqrt{n})\right)$. To see this is the case, the probability of $\mathbf{T}_i$ having less than $n^{1/3}$ many elements can be bounded from above by

$$\begin{aligned}\mathbf{P}r\left[|\mathbf{T}_i| \le n^{1/3}\right] &= \sum_{j\le n^{1/3}} \binom{n/2}{j} \cdot \left(1 - \frac{1}{\sqrt{n}}\right)^{n/2-j} \cdot \left(\frac{1}{\sqrt{n}}\right)^{j} \\ &\le (n^{1/3}+1) \cdot \binom{n/2}{n^{1/3}} \cdot \left(1 - \frac{1}{\sqrt{n}}\right)^{n/2-n^{1/3}} < e^{-0.49\sqrt{n}}.\end{aligned}$$

Taking a union bound over all $N \approx e^{\sqrt{n}/4}$ terms, we conclude that $\mathbf{T} \sim \mathcal{E}(M)$ lies in the support of $\mathcal{E}'(M)$ with probability at least $1 - \exp(-0.24\sqrt{n})$.

In Claim 2.3.4, we prove a lower bound for the expectation of $|\mathbf{X}|$ (note that $\mathbf{X}$ only depends on $\mathbf{T}$):

Claim 2.3.4. *We have*

$$\mathop{\mathbf{E}}_{\mathbf{T}\sim\mathcal{E}(M)}\left[|\mathbf{X}|\right] = \Omega(2^n) \quad \textit{and} \quad \mathop{\mathbf{E}}_{\mathbf{T}\sim\mathcal{E}'(M)}\left[|\mathbf{X}|\right] = \Omega(2^n). \tag{2.21}$$

Proof. By linearity of expectation, we have

$$\mathop{\mathbf{E}}_{\mathbf{T}\sim\mathcal{E}(M)}\left[|\mathbf{X}|\right] = \sum_{x \text{ in middle layers}} \Pr_{\mathbf{T}\sim\mathcal{E}(M)}\left[x \in \mathbf{X}\right].$$

Fix a string $x \in \{0,1\}^n$ in the middle layers (i.e., $|x_M|$ lies in $n/4 \pm \sqrt{n}$). We decompose the probability on the RHS for x into N disjoint subevents. The ith subevent corresponds to $\mathbf{T}_i$ being the unique term which x satisfies. The probability of the ith subevent is at least

$$\left(1 - \frac{1}{\sqrt{n}}\right)^{\frac{n}{4}+\sqrt{n}} \times \left(1 - \left(1 - \frac{1}{\sqrt{n}}\right)^{\frac{n}{4}-\sqrt{n}}\right)^{N-1} = \Omega\left(\frac{1}{N}\right).$$

As a result, the probability of $x \in \mathbf{X}$ is $N \cdot \Omega(1/N) = \Omega(1)$. The first part of (2.21) follows from the fact that there are $\Omega(2^n)$ many strings x in the middle layers.

The second part of (2.21) follows from the first part and the fact that $|\mathbf{X}| \leq 2^n$ and $\mathbf{T} \sim \mathcal{E}(M)$ does not lie in the support of $\mathcal{E}'(M)$ with probability $o(1)$ as shown above. □

Let $\mu^* = \Omega(2^n)$ be the expectation of $|\mathbf{X}|$ over $\mathbf{T} \sim \mathcal{E}'(M)$, and let p be the probability that $|\mathbf{X}| \geq \mu^*/2$. Then we have

$$\mu^* \leq p \cdot 2^n + (1-p) \cdot (\mu^*/2) \leq p \cdot 2^n + \mu^*/2$$

and thus, $p = \Omega(1)$. As a result, it suffices to consider a T in the support of $\mathcal{E}'(M)$ that satisfies $|X| \geq \mu^*/2$ and show that, over $\mathbf{H} \sim \mathcal{E}_{\text{no}}(M)$, all $|\mathbf{X}_k^+|$ and $|\mathbf{X}_k^-|$ are $\Omega(2^n/n)$ with probability $\Omega(1)$. To this end, we focus on $\mathbf{X}_k^+$ and then use symmetry and a union bound on all the n sets.

Given $k \in \overline{M}$, T and its corresponding X (since $\mathbf{X}$ only depends on $\mathbf{T}$) with $|X| \geq \mu^*/2$, we note that half of $x \in X$ have $x_k = 0$ (since whether $x \in X$ only depends on x_M) and for each $x \in X$ with $x_k = 0$, the probability of $x \in \mathbf{X}_k^+$ (over $\mathbf{H}$) is $1/(2n)$. Hence, the expectation of $|\mathbf{X}_k^+|$ is $|X|/4n \geq \mu^*/8n = \Omega(2^n/n)$. Let $\mu = |X|/4n$. To obtain a concentration bound on $|\mathbf{X}_k^+|$, we apply Hoeffding's inequality over $\mathbf{H} \sim \mathcal{E}_{\text{no}}(M)$ in the next claim.

Claim 2.3.5. *For each $k \in \overline{M}$, we have*

$$\Pr_{\mathbf{H} \sim \mathcal{E}_{no}(M)}\left[\mu - |\mathbf{X}_k^+| \geq \mu/2\right] \leq \exp\left(-\Omega\left(2^{n^{1/3}}/n^2\right)\right).$$

Proof. Consider the size of $\mathbf{X}_k^+$ as a function over $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_N$ for a particular fixed T in the support of $\mathcal{E}'(M)$ with $|X| \geq \Omega(2^n)$. We have that $|\mathbf{X}_k^+|$ is a sum of independent random variables taking values between 0 and $2^{n-n^{1/3}}$. This is because each term T_i has length at least $n^{1/3}$, and it can add at most $b_i \leq 2^{n-n^{1/3}}$ strings to $|\mathbf{X}_k^+|$. Then we have:

$$\sum_{i\in[N]} b_i^2 \leq 2^{n-n^{1/3}} \sum_{i\in[N]} b_i \leq 2^{2n-n^{1/3}}.$$

Also the expectation of $|\mathbf{X}_k^+|$ is μ because the choices in $\mathbf{H}$ partitions half of X into $2n$ disjoint parts. We can now apply Hoeffding's inequality:

$$\mathop{\mathbf{P}r}_{\mathbf{H}\sim\mathcal{E}_{\text{no}}(M)}\left[\mu - |\mathbf{X}_k^+| \geq \frac{\mu}{2}\right] \leq \exp\left(-\frac{\Omega(2^{2n}/n^2)}{2^{2n-n^{1/3}}}\right) = \exp\left(-\Omega\left(2^{n^{1/3}}/n^2\right)\right)$$

This finishes the proof of the claim. □

The same argument works for $|\mathbf{X}_k^-|$. (2.20) then follows from a union bound on $k \in \overline{M}$ and both sets $\mathbf{X}_k^+$ and $\mathbf{X}_k^-$. This finishes the proof of Lemma 2.3.2. □

Given Lemma 2.3.1 and Lemma 2.3.2, our lower bound for testing unateness (Theorem 1.2.4) follows directly from the lemma below. We fix $q = n^{2/3}/\log^3 n$ as the number of queries in the rest of the proof. The remainder of this section will prove the following lemma.

Lemma 2.3.6. *Let B be any q-query deterministic algorithm with black-box oracle access to f. Then*

$$\Pr_{\mathbf{f}\sim\mathcal{D}_{no}}\left[B \text{ rejects } \mathbf{f}\right] \leq \Pr_{\mathbf{f}\sim\mathcal{D}_{yes}}\left[B \text{ rejects } \mathbf{f}\right] + o(1).$$

2.3.2 Proof of Lemma 2.3.6

We prove Lemma 2.3.6 in this section. Similar with the proof for lower bound of monotonicity testing, here we also start with a few more definitions.

2.3.2.1 Balanced decision trees

Let B be a q-query deterministic algorithm, i.e., a binary decision tree of depth at most q in which each internal node is labeled a query string $x \in \{0,1\}^n$ and each leaf is labelled "accept" or "reject." Each internal node u has one 0-child and one 1-child. For each internal node u, we use Q_u to denote the set of strings queried so far (not including the query x to be made at u).

Next we give the definition of a q-query tree B being *balanced* with respect to a subset $M \subset [n]$ of size $n/2$ and a string $r \in \{0,1\}^M$ (as the **M** and **r** in the procedure that generates $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$). After the definition we show that, when both **M** and **r** are drawn uniformly at random (as in the procedure), B is balanced with respect to **M** and **r** with probability at least $1 - o(1)$.

Definition 2.3.7 (Balance). *We say B is* balanced *with respect to a subset $M \subset [n]$ of size $n/2$ and $r \in \{0,1\}^M$ if for every internal node u of B (letting x be the query at u) and every $Q \subseteq Q_u$, with*

$$A = \left\{k \in [n] : \forall y, y' \in Q,\, y_k = y'_k\right\} \quad \textit{and} \quad A' = \left\{k \in [n] : \forall y, y' \in Q \cup \{x\},\, y_k = y'_k\right\}, \tag{2.22}$$

the set $\Delta = A \setminus A'$ having size at least $n^{2/3} \log n$ implies that

$$\Delta_1 = \left\{k \in \Delta \cap M : x_k \oplus r_k = 0 \textit{ and } \forall y \in Q,\, y_k \oplus r_k = 1\right\} \tag{2.23}$$

has size at least $n^{2/3} \log n/8$.

Lemma 2.3.8. *Let B be a q-query decision tree. Then B is balanced with respect to a subset $\mathbf{M} \subset [n]$ of size $n/2$ and an $\mathbf{r} \in \{0,1\}^{\mathbf{M}}$, both drawn uniformly at random, with probability at least $1 - o(1)$*

Proof. Fix an internal node u and a $Q \subseteq Q_u$ such that $|\Delta| \geq n^{2/3} \log n$. Then the

probability over the draw of $\mathbf{M}$ and $\mathbf{r}$ of $\mathbf{\Delta}_1$ being smaller than $n^{2/3}\log n/8$ is at most $\exp(-\Omega(n^{2/3}\log n))$ using the Chernoff bound. The lemma follows by a union bound as there are at most $O(2^q)$ choices for u and 2^q choices for Q. □

Lemma 2.3.6 follows from the following lemma.

Lemma 2.3.9. *Fix M and r. Let B be a q-query tree that is balanced with respect to M and r. Then we have*

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{no}(M),\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{M,\mathbf{T},\mathbf{H},r,\mathbf{s}}\right] \leq \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{yes}(M),\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{M,\mathbf{T},\mathbf{H},r,\mathbf{s}}\right] + o(1). \tag{2.24}$$

where $\mathbf{T}\sim\mathcal{E}(M)$ and $\mathbf{s}\sim\{0,1\}^{\overline{M}}$.

Proof of Lemma 2.3.6 assuming Lemma 2.3.9. To simplify the notation, in the sequence of equations below we ignore in the subscripts names of distributions from which certain random variables are drawn when it is clear from the context. Using Lemma 2.3.8 and Lemma 2.3.9, we have

$$\begin{aligned}
&\Pr_{\mathbf{M},\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}}(M),\mathbf{r},\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}}\right] \\
&\leq \frac{1}{2^{n/2}\cdot\binom{n}{n/2}}\cdot\sum_{M,r}\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}}(M),\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{M,\mathbf{T},\mathbf{H},r,\mathbf{s}}\right] \\
&\leq \frac{1}{2^{n/2}\cdot\binom{n}{n/2}}\cdot\sum_{M,r:\text{ balanced } B}\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}}(M),\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{M,\mathbf{T},\mathbf{H},r,\mathbf{s}}\right] + o(1) \\
&\leq \frac{1}{2^{n/2}\cdot\binom{n}{n/2}}\cdot\sum_{M,r:\text{ balanced } B}\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}}(M),\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{M,\mathbf{T},\mathbf{H},r,\mathbf{s}}\right] + o(1) \\
&\leq \Pr_{\mathbf{M},\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}}(M),\mathbf{r},\mathbf{s}}\left[B \text{ rejects } \mathbf{f}_{\mathbf{M},\mathbf{T},\mathbf{H},\mathbf{r},\mathbf{s}}\right] + o(1).
\end{aligned}$$

This finishes the proof of Lemma 2.3.6. □

To prove Lemma 2.3.9, we may consider an adversary that has M of size $n/2$ and $r\in\{0,1\}^M$ in hand and can come up with any q-query decision tree B as long as B is

balanced with respect to M and r. Our goal is to show that any such tree B satisfies (2.24). This inspires us to introduce the definition of *balanced* decision trees.

Definition 2.3.10 (Balanced Decision Trees). *A q-query tree B is said to be* balanced *(without specifying M and r here) if it is balanced with respect to $M^* = [n/2]$ and $r^* = 0^{[n/2]} \in \{0,1\}^M$. Equivalently, for every internal node u of B and every $Q \subseteq Q_u$ (letting A and A' denote the sets as defined in (2.22)), if $\Delta = A \setminus A'$ has size at least $n^{2/3} \log n$, then the set Δ_1 as defined in (2.23) using M^* and r^* has size at least $n^{2/3} \log n/8$.*

With Definition 2.3.10 in hand, we use the following lemma to prove Lemma 2.3.9.

Lemma 2.3.11. *Let B' be a balanced q-query decision tree. Then we have*

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{no}(M^*),\mathbf{s}}\left[B' \text{ rejects } f_{M^*,\mathbf{T},\mathbf{H},r^*,\mathbf{s}}\right] \leq \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{yes}(M^*),\mathbf{s}}\left[B' \text{ rejects } f_{M^*,\mathbf{T},\mathbf{H},r^*,\mathbf{s}}\right] + o(1), \quad (2.25)$$

where $\mathbf{T} \sim \mathcal{E}(M^*)$ and $\mathbf{s} \sim \{0,1\}^{\overline{M^*}}$.

Proof of Lemma 2.3.9 assuming Lemma 2.3.11. Let B be a q-query tree from the statement of Lemma 2.3.9 that is balanced with respect to M and $r \in \{0,1\}^M$, which are not necessarily the same as M^* and r^*. Then we use B, M and r to define a new q-query tree B' that is balanced (with respect to M^* and r^*): B' is obtained by replacing every query x made in B by x', where x' is obtained by first doing an XOR of x with r over indices in M and then reordering the indices of the new x using a bijection between M and M^*. Then it's easy to see B' is balanced, and the LHS of (2.24) for B is the same as the LHS of (2.25) for B'. The same holds for the RHS of (2.24) and (2.25) as well. Lemma 2.3.9 then follows from Lemma 2.3.11. □

For simplicity in notation, we fix M and r to be $[n/2]$ and $0^{[n/2]}$ in the rest of the section. We also write $\mathcal{E}$ for $\mathcal{E}(M)$, $\mathcal{E}_{\text{yes}}$ for $\mathcal{E}_{\text{yes}}(M)$, and $\mathcal{E}_{\text{no}}$ for $\mathcal{E}_{\text{no}}(M)$. Given T in the

support of $\mathcal{E}$, H from the support of $\mathcal{E}_{\text{yes}}$ or $\mathcal{E}_{\text{no}}$, and $s \in \{0,1\}^{\overline{M}}$, we write

$$f_{T,H,s} =: f_{M,T,H,r,s}$$

for convenience. Then the goal (2.25) of Lemma 2.3.11 becomes

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}},\mathbf{s}}\left[B \text{ rejects } f_{\mathbf{T},\mathbf{H},\mathbf{s}}\right] \leq \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}},\mathbf{s}}\left[B \text{ rejects } f_{\mathbf{T},\mathbf{H},\mathbf{s}}\right] + o(1),$$

where $\mathbf{T} \sim \mathcal{E}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$ in both probabilities.

Remark 3. *Since B works on $f_{\mathbf{T},\mathbf{H},\mathbf{s}}$ and r is all-0, the multiplexer $\mathbf{\Gamma}_{\mathbf{T}}$ is first truncated according to $|x_M|$, the number of 1's in the first $n/2$ indices. As a consequence, we may now assume without loss generality from now on that B only queries strings x that have $|x_M|$ lying between $n/4 \pm \sqrt{n}$. We will refer to them as strings in the middle layers in the rest of the section.*

2.3.2.2 Balanced signature trees

At a high level we proceed in a similar fashion as in the proof of monotonicity lower bound. We first define a new and stronger oracle model that returns more than just $f(x) \in \{0,1\}$ for each query $x \in \{0,1\}^n$. Upon each query $x \in \{0,1\}^n$, the oracle returns the so-called *signature* of $x \in \{0,1\}^n$ with respect to (T,H,s) when hidden function is $f_{T,H,s}$ (and it will become clear that $f_{T,H,s}(x)$ is determined by the signature of x); in addition, the oracle also reveals the special variable k of a term T_i when the latter is *breached* (see Definition 2.3.17). Note that the revelation of special variables is new in the proof of unateness lower bound compared to that of monotonicity lower bound. On the other hand, the definition of signatures in this section is much simpler due to the single-level construction of the multiplexer map.

After the introduction of the stronger oracle model, ideally we would like to prove that

every q-query deterministic algorithm C with access to the new oracle can only have at most $o(1)$ advantage in rejecting the function $f_{\mathbf{T},\mathbf{H},\mathbf{s}}$ when $\mathbf{T} \sim \mathcal{E}$, $\mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$ as compared to $\mathbf{T}$, $\mathbf{H} \sim \mathcal{E}_{\text{yes}}$ and $\mathbf{s}$. It turns out that we are only able to prove this when C is represented by a so-called *balanced signature tree*, a definition closely inspired by that of balanced decision trees in Definition 2.3.10. This suffices for us to prove Lemma 2.3.11 since only balanced decision trees are considered there.

Recall the definition of e_i and $e_{i,i'}$ from Section 2.1.1 (but with $N = (1 + 1/\sqrt{n})^{n/4}$ now). We first define signatures syntactically and then semantically. The two definitions below are simpler than their counterparts in Section 2.2 (as we only have one level of multiplexing in Γ_T). By Remark 3, we can assume without loss of generality that every string queried lies in the middle layers.

Definition 2.3.12. *We use $\mathfrak{P}$ to denote the set of all triples (σ, a, b), where $\sigma \in \{0,1,*\}^N$ and $a, b \in \{0,1,\perp\}$ satisfy the following properties:*

1. *σ is either 1) the all 0-string 0^N, 2) e_i for some $i \in [N]$, or 3) $e_{i,i'}$ for some $i, i' \in [N]$, $i < i'$.*
2. *If σ is of case 1), then $a = b = \perp$. If σ is of case 2), then $a \in \{0,1\}$ and $b = \perp$. Lastly, if σ is of case 3), then we have $a, b \in \{0,1\}$.*

Definition 2.3.13. *We say $(\sigma, a, b) \in \mathfrak{P}$ is the* signature *of a string $x \in \{0,1\}^n$ in the middle layers with respect to (T, H, s) if it satisfies the following properties:*

1. *$\sigma \in \{0,1,*\}^N$ is set according to the following three cases: 1) $\sigma = 0^N$ if $T_i(x) = 0$ for all $i \in [N]$; 2) $\sigma = e_i$ if $T_i(x) = 1$ is the unique term that is satisfied by x; 3) $\sigma = e_{i,i'}$ if $i < i'$ and $T_i(x) = T_{i'}(x) = 1$ are the first two terms that are satisfied by x.*
2. *If σ is in case 1), then $a = b = \perp$; if σ is in case 2) with $\sigma = e_i$, then $a = h_i(x \oplus s)$ [8] and $b = \perp$; if σ is in case 3) with $\sigma = e_{i,i'}$, then $a = h_i(x \oplus s)$ and $b = h_{i'}(x \oplus s)$.*

[8] Recall that $x \oplus s$ is the n-bit string obtained from x after an XOR with s over indices in $\overline{M}$.

The signature map of a set $Q \subset \{0,1\}^n$ of strings in the middle layers with respect to (T, H, s) is the map $\phi\colon Q \to \mathfrak{P}$ such that $\phi(x)$ is the signature of x with respect to (T, H, s) for each $x \in Q$.

Next we show that $f_{T,H,s}(x)$ is uniquely determined by the signature of x. Thus, the new oracle is at least as powerful as the standard one. The proof is similar to that of Lemma 2.2.9.

Lemma 2.3.14. *Let T be from the support of $\mathcal{E}$, H be from the support of $\mathcal{E}_{yes}$ or $\mathcal{E}_{no}$ and $s \in \{0,1\}^{\overline{M}}$. Given an $x \in \{0,1\}^n$ in the middle layers, $f_{T,H,s}(x)$ is uniquely determined by the signature (σ, a, b) of x with respect to (T, H, s).*

Proof. Let $f = f_{T,H,s}$. We consider the following three cases:

1. (No term is satisifed) If $\sigma = 0^N$, then $f(x) = 0$.
2. (Unique term satisfied) If If $\sigma = e_i$ for some $i \in [N]$, then $f(x) = h_i(x \oplus s) = a$.
3. (Multiple terms satisfied) If $\sigma = e_{i,i'}$ for some $i < i' \in [N]$, then $f(x) = 1$.

This finishes the proof of the lemma. □

We have defined the signature of x with respect to (T, H, s), which is the first thing that the new oracle returns upon a query x. Let $Q \subset \{0,1\}^n$ be a set of strings in the middle layers (and consider Q as the set of queries made so far by an algorithm). Next we define terms *breached* by Q with respect to a triple (T, H, s). Upon a query x, the new oracle checks if there is any term(s) newly breached after x is queried; if so, the oracle also reveals its special variable in $\overline{M}$.

For this purpose, let $\phi : Q \to \mathfrak{P}$ be the signature map of Q with respect to (T, H, s), where $\phi(x) = (\sigma_x, a_x, b_x)$. We say ϕ induces a 5-tuple $(I; P; R; A; \rho)$ if it satisfies the following properties:

- The set $I \subseteq [N]$ is given by

$$I = \big\{i \in [N] : \exists x \in Q \text{ with } \sigma_{x,i} = 1\big\}.$$

- $P = (P_i : i \in I)$ and $R = (R_i : i \in I)$ are two tuples of subsets of Q. For each $i \in I$,
$$P_i = \big\{x \in Q : \sigma_{x,i} = 1\big\} \quad \text{and} \quad R_i = \big\{x \in Q : \sigma_{x,i} = 0\big\}.$$
- $A = (A_i, A_{i,0}, A_{i,1} : i \in I)$ is a tuple of subsets of $[n]$, such that for each $i \in I$, $A_i = A_{i,0} \cup A_{i,1}$ and
$$A_{i,1} = \big\{k \in [n] : \forall x \in P_i, x_k = 1\big\} \quad \text{and} \quad A_{i,0} = \big\{k \in [n] : \forall x \in P_i, x_k = 0\big\}.$$
- $\rho = (\rho_i : i \in I)$ is a tuple of functions $\rho_i : P_i \to \{0,1\}$ with $\rho_i(x) = a_x$ if either $\sigma_x = e_i$ or $\sigma_x = e_{i,i'}$ for some $i' > i$, and $\rho_i(x) = b_x$ if $\sigma_x = e_{i',i}$ for some $i' < i$, for each $x \in P_i$ ($\rho_i(x)$ gives us the value of $h_i(x \oplus s)$ for each $x \in P_i$).

The following fact is reminiscent of Fact 2.2.12.

Fact 2.3.15. *Let $\phi : Q \to \mathfrak{P}$ be the signature map of Q with respect to (T, H, s). Then for each $i \in I$, we have $T_i \subseteq A_{i,1} \cap M$, $T_i(x) = 0$ for all $x \in R_i$, and $h_i(x \oplus s) = \rho_i(x)$ for each $x \in P_i$.*

We introduce the similar concept of consistency as in Definition 2.2.13.

Definition 2.3.16. *Let $(I; P; R; A; \rho)$ be the tuple induced by $\phi : Q \to \mathfrak{P}$. For each $i \in I$, we say P_i is* 1-consistent *if $\rho_i(x) = 1$ for all $x \in P_i$, and* 0-consistent *if $\rho_i(x) = 0$ for all $x \in P_i$. We say P_i is* consistent *if it is either 1-consistent or 0-consistent; we say P_i is* inconsistent *otherwise.*

We are now ready to define terms breached by Q with respect to (T, H, s).

Definition 2.3.17 (Breached Terms). *Let $Q \subset \{0,1\}^n$ be a set of strings in the middle layers. Let T be from the support of $\mathcal{E}$, H be from the support of $\mathcal{E}_{yes}$ or $\mathcal{E}_{no}$, and $s \in \{0,1\}^{\overline{M}}$. Let $(I; P; R; A; \rho)$ be the tuple induced by the signature map of Q with respect to (T, H, s). We say the ith term is breached by Q with respect to (T, H, s), for some $i \in I$, if at least one of*

the following two events happens: (1) P_i is inconsistent or (2) $|A_i \cap \overline{M}| \leq n/10$. We say the ith term is safe *if it is not breached.*

We can now finish the formal definition of our new oracle model. Upon each query x, the oracle first returns the signature of x with respect to the hidden triple (T, H, s). It then examines if there is any newly breached term(s) (by Definition 2.3.17 there can be at most two such terms since x can be added to at most two P_i's) and return the special variable $k \in \overline{M}$ of the newly breached term(s). As a result, if Q is the set of queries made so far, the information returned by the new oracle can be summarized as a 6-tuple $(I; P; R; A; \rho; \delta)$, where

1. $(I; P; R; A; \rho)$ is the tuple induced by the signature map of Q with respect to (T, H, s);
2. Let $I_B \subseteq I$ be the set of indices of terms breached by Q, and let $I_S = I \setminus I_B$ denote the safe terms. Then $\delta : I_B \to \overline{M}$ satisfies that $k = \delta(i)$ is the special variable of the ith term in h_i.

We view a q-query deterministic algorithm C with access to the new oracle as a *signature tree*, in which each leaf is labeled "accept" or "reject" and each internal node u is labeled a query string $x \in \{0,1\}^n$ in the middle layers. Each internal node u has $|\mathfrak{P}| \cdot O(n^2)$ children with each of its edges (u, v) labeled by (1) a triple $(\sigma, a, b) \in \mathfrak{P}$ as the signature of x with respect to the hidden (T, H, s), and (2) the special variable of any newly breached (at most two) term(s). Each node u is associated with a set Q_u as the set of queries made so far (not including x), its signature map $\phi : Q_u \to \mathfrak{P}$, and a tuple $(I; P; R; A; \rho; \delta)$ (induced by ϕ) as the summary of all information received from the oracle so far. (Note that one can fully reconstruct the signature map ϕ from $(I; P; R; A; \rho)$ so it is redundant to keep ϕ. We keep it because sometimes it is (notation-wise) easier to work with ϕ directly.) Similar with the proof of monotonicity lower bound, for simplicity we also say the triple (T, H, s)

reaches a node (or leaf) u in C if the algorithm reaches u given access to the new oracle with respect to this triple.

Finally we define *balanced* signature trees.

Definition 2.3.18 (Balanced Signature Trees). *We say that a signature tree C is* balanced *if for any internal node u of C (letting x be the query to make and $(I; P; R; A; \rho; \delta)$ be the summary so far) and any $i \in I$, $\Delta = \{j \in A_i : x_j$ disagrees with y_j of $y \in P_i\}$ having size at least $n^{2/3} \log n$ implies that $\Delta_1 = \{k \in \Delta \cap M : x_k = 0$ and $\forall\, y \in P_i,\ y_k = 1\}$ has size at least $n^{2/3} \log n/8$.*

Note that the definition above is weaker compared to Definition 2.3.10 of balanced decision trees, in the sense that the condition on Δ_1 in the latter applies to any subset of queries $Q \subseteq Q_u$ (instead of only P_i's). Lemma 2.3.11 follows from the lemma below on balanced signature trees.

Lemma 2.3.19. *Let C be a q-query balanced signature tree. Then we have*

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{no},\mathbf{s}}\left[C \text{ rejects } (\mathbf{T},\mathbf{H},\mathbf{s})\right] \leq \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{yes},\mathbf{s}}\left[C \text{ rejects } (\mathbf{T},\mathbf{H},\mathbf{s})\right] + o(1). \tag{2.26}$$

Proof of Lemma 2.3.11 assuming Lemma 2.3.19. Let B be a q-query balanced decision tree. We use B to obtain a q-query algorithm C with access to the new oracle by simulating B as follows: each time a string x is queried, C uses the signature of x returned by the oracle to extract $f(x)$ (using Lemma 2.3.14) and then continue the simulation of B. One can verify that the corresponding signature tree of C is balanced and the probabilities of C rejecting $(\mathbf{T},\mathbf{H},\mathbf{s})$ in both cases are the same as B. □

Before moving on to the proof of Lemma 2.3.19, let us remark on how the new oracle may help an algorithm distinguish between functions in $\mathcal{D}_{yes}$ and $\mathcal{D}_{no}$. Suppose that a deterministic algorithm C is at some internal node u with a tuple $(I; P; R; A; \rho; \delta)$. For each breached $i \in I_B$, the algorithm knows that h_i is either a dictator or anti-dictator

with special variable x_k with $k = \delta(i)$. By inspecting the y_k of a $y \in P_i$ and $\rho_i(y)$, the algorithm can also deduce whether $h_i(x \oplus s)$ is x_k or $\overline{x_k}$. The former suggests that x_k is monotone and the latter suggests that x_k is anti-monotone.

However, unlike monotonicity testing, observing $h_i(x \oplus s) = \overline{x_k}$ has no indication on whether f is drawn from $\mathcal{D}_{yes}$ or $\mathcal{D}_{no}$: indeed $h_i(x \oplus s)$ is equally possible to be x_k or $\overline{x_k}$ in both distributions because of the random bit s_k. But if the algorithm observes a so-called *collision*, i.e. $i, i' \in I_B$ such that $h_i(x \oplus s) = x_k$ and $h_{i'}(x \oplus s) = \overline{x_k}$, then one can safely assert that the hidden function belongs to $\mathcal{D}_{no}$. This gives us the crucial insight (as sketched earlier in Section 2.3.1) that leads to a higher unateness testing lower bound than monotonicity testing: for testing monotonicity, deducing that a variable goes in an anti-monotone direction suffices for a violation; for testing unateness, however, one needs to find a collision in order to observe a violation. While the proof of Lemma 2.3.19 is quite technical, it follows the intuition that with q queries, it is hard for a balanced signature tree to find a collision in breached terms I_B, and when no collision is found, it is hard to tell where the hidden function is drawn from.

2.3.2.3 Tree pruning

To prove Lemma 2.3.19 on a given balanced q-query signature tree C, we start by identifying a set of *bad edges* of C and using them to prune the tree.

Definition 2.3.20. *An edge (u, v) in C is a* bad *edge if at least one of the following events happens at (u, v) and none of these events happens along the root-to-u path (letting x be the string queried at u, and $(I_B \cup I_S; P; R; A; \rho; \delta)$ and $(I'_B \cup I'_S; P'; R'; A'; \rho'; \delta')$ be the summaries at u and v, respectively):*

1. *For some $i \in I_S$, $|A_i \setminus A'_i| \geq n^{2/3} \log n$;*
2. *$|I'_B| > n^{1/3}/\log n$; or*
3. *There exist two distinct indices $i, j \in I'_B$ with $\delta'(i) = \delta'(j)$.*

We say a leaf ℓ of C is a *good* leaf if there is no bad edge along the root-to-ℓ path; otherwise, ℓ is *bad*. The following lemma allows us to focus on good leaves. We defer the proof to Section 2.3.2.5.

Lemma 2.3.21 (Pruning Lemma). *Let C be a balanced q-query signature tree. Then*

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{no},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \textit{ reaches a bad leaf}\right] = o(1).$$

We prove the following lemma for good leaves in Section 2.3.22:

Lemma 2.3.22. *For any good leaf ℓ of C, we have*

$$\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{no},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \textit{ reaches } \ell\right] \le (1+o(1)) \cdot \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{yes},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \textit{ reaches } \ell\right].$$

Assuming Lemma 2.3.21 and Lemma 2.3.22, we can prove Lemma 2.3.19:

Proof of Lemma 2.3.19 assuming Lemma 2.3.21 and Lemma 2.3.22. Let L be the set of leaves of C that are labeled "reject" and let $L^* \subseteq L$ be the good ones in L. Then we have

$$\begin{aligned}
\Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}},\mathbf{s}}\left[C \text{ reject } (\mathbf{T},\mathbf{H},\mathbf{s})\right] &= \sum_{\ell\in L} \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \text{ reaches } \ell\right] \\
&\le \sum_{\ell\in L^*} \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{no}},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \text{ reaches } \ell\right] + o(1) \\
&\le (1+o(1)) \cdot \sum_{\ell\in L^*} \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}},\mathbf{s}}\left[(\mathbf{T},\mathbf{H},\mathbf{s}) \text{ reaches } \ell\right] + o(1) \\
&\le (1+o(1)) \cdot \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}},\mathbf{s}}\left[C \text{ rejects } (\mathbf{T},\mathbf{H},\mathbf{s})\right] + o(1) \\
&\le \Pr_{\mathbf{T},\mathbf{H}\sim\mathcal{E}_{\text{yes}},\mathbf{s}}\left[C \text{ rejects } (\mathbf{T},\mathbf{H},\mathbf{s})\right] + o(1),
\end{aligned}$$

where we used Lemma 2.3.21 in the second line and Lemma 2.3.22 in the third line. □

2.3.2.4 Proof of Lemma 2.3.22 for good leaves

The proof of Lemma 2.3.22 is similar in spirit to Lemma 2.2.17 for monotonicity.

Fix a good leaf ℓ in C. We let Q be the set of queries made along the root-to-ℓ path, $\phi : Q \to \mathfrak{P}$ be the signature map of Q with $\phi(x) = (\sigma_x, a_x, b_x)$ for each $x \in Q$, and let $(I_B \cup I_S; P; R; A; \rho; \delta)$ be the summary associated with ℓ. Since ℓ is a good leaf, there are no bad edges along the root-to-ℓ path. Combining this with the definition of breached/safe terms, we have the following list of properties:

1. For each $i \in I_S$, $|A_i \cap \overline{M}| \geq n/10$;
2. Every $i \in I_S$ is either 1-consistent or 0-consistent;
3. $|I_B| \leq n^{1/3}/\log n$; and
4. For any two distinct indices $i, j \in I_B$, we have $\delta(i) \neq \delta(j)$.

Let $D = \{\delta(i) : i \in I_B\} \subset \overline{M}$ be the special variables of breach terms. We have $|D| = |I_B|$.

Next we fix a tuple T from the support of $\mathcal{E}$ such that the probability of $(T, \mathbf{H}, \mathbf{s})$ reaching ℓ is positive, when $\mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$. It then suffices to show that

$$\mathop{\mathbf{P}r}_{\mathbf{H} \sim \mathcal{E}_{\text{yes}}, \mathbf{s}} \left[(T, \mathbf{H}, \mathbf{s}) \text{ reaches } \ell \right] \geq (1 - o(1)) \mathop{\mathbf{P}r}_{\mathbf{H} \sim \mathcal{E}_{\text{no}}, \mathbf{s}} \left[(T, \mathbf{H}, \mathbf{s}) \text{ reaches } \ell \right]. \tag{2.27}$$

The properties below follow directly from the assumption that the probability of $(T, \mathbf{H}, \mathbf{s})$ reaching ℓ is positive when $\mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$:

1. For every $x \in Q$ and $i \in [N]$ such that $\sigma_{x,i} \in \{0,1\}$, we have $T_i(x) = \sigma_{x,i}$; and
2. For each $i \in I_B$, letting $k = \delta(i)$, there exists a bit b such that $\rho_i(x) = x_k \oplus b$ for all $x \in P_i$.

For each $i \in I_B \cup I_R$ we pick a string y_i from P_i arbitrarily as a representative and let $\alpha_i = \rho_i(y_i)$.

We first derive an explicit expression for the probability over $\mathcal{E}_{\text{no}}$ in (2.27). To this end, we note that, given properties listed above, $(T, \mathbf{H}, \mathbf{s})$ (with $\mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$) reaches ℓ iff

1. For each $i \in I_S$, let $\boldsymbol{k}$ be the special variable of $\boldsymbol{h}_i$. Then we have $\boldsymbol{k} \in A_i \cap \overline{M}$. Furthermore, $\boldsymbol{h}_i$ is a dictatorship function if $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} = \alpha_i$ or an anti-dictatorship if $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} \neq \alpha_i$;
2. For each $i \in I_B$, the special variable of $\boldsymbol{h}_i$ is the same as $\boldsymbol{k} = \delta(i)$ and similarly, $\boldsymbol{h}_i$ is a dictatorship function if $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} = \alpha_i$ or an anti-dictatorship if $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} \neq \alpha_i$.

Thus, once s is fixed, there is exactly one choice of $\boldsymbol{h}_i$ for each $i \in I_B$ and $|A_i \cap \overline{M}|$ choices of $\boldsymbol{h}_i$ for each $i \in I_S$. Since there are $(n/2) \cdot 2$ choices overall for each $\boldsymbol{h}_i$, the probability over $\mathcal{E}_{\text{no}}$ in (2.27) is

$$\left(\frac{1}{n}\right)^{|I_B|} \cdot \prod_{i \in I_S} \left(\frac{|A_i \cap \overline{M}|}{n}\right).$$

Next we work on the more involved probability over $\mathcal{E}_{\text{yes}}$ in (2.27). Given properties listed above $(T, \mathbf{H}, \mathbf{s})$ (with $\mathbf{s} \sim \{0,1\}^{\overline{M}}$ and $\mathbf{H} \sim \mathcal{E}_{\text{yes}}$ so every $\boldsymbol{h}_i$ is a dictatorship function) reaches ℓ iff

1. For each $i \in I_S$, let $\boldsymbol{k}$ be the special variable of the dictatorship function $\boldsymbol{h}_i$. Then we have $\boldsymbol{k} \in A_i \cap \overline{M}$ and $s_{\boldsymbol{k}}$ satisfies that $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} = \alpha_i$;
2. For each $i \in I_B$, the special variable of $\boldsymbol{h}_i$ is the same as $\boldsymbol{k} = \delta(i)$ and we always have $y_{i,\boldsymbol{k}} \oplus s_{\boldsymbol{k}} = \alpha_i$.

Note that once $\mathbf{s} = s$ is fixed, these are independent conditions over $\boldsymbol{h}_i$'s (among the overall $n/2$ choices for each $\boldsymbol{h}_i$). As a result, we can rewrite the probability for $\mathcal{E}_{\text{yes}}$ as

$$\mathop{\mathbf{E}}_{\mathbf{s} \sim \{0,1\}^{\overline{M}}} \left[\prod_{i \in I} \mathbf{Z}_i \right], \tag{2.28}$$

where $\mathbf{Z}_i$'s are (correlated) random variables that depend on $\mathbf{s}$. For each $i \in I_B$, $\mathbf{Z}_i = 2/n$ if

$$\alpha_i = y_{i,\delta(i)} \oplus \mathbf{s}_{\delta(i)}$$

and $\mathbf{Z}_i = 0$ otherwise. For each $i \in I_S$, we have

$$\mathbf{Z}_i = \frac{|\{k \in A_i \cap \overline{M} : y_{i,k} \oplus \mathbf{s}_k = \alpha_i\}|}{n/2}$$

For some technical reason, for each $i \in I_S$, let $\mathbf{B}_i$ be the following random set that depends on $\mathbf{s}$:

$$\mathbf{B}_i = \left\{k \in (A_i \cap \overline{M}) \setminus D : y_{i,k} \oplus \mathbf{s}_k = \alpha_i\right\}.$$

Using $|D| = |I_B|$, we may now simplify (2.28) by:

$$\begin{aligned}\mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}}}\left[\prod_{i\in I}\mathbf{Z}_i\right] &= \frac{1}{2^{|I_B|}}\cdot\left(\frac{2}{n}\right)^{|I_B|}\mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}\setminus D}}\left[\prod_{i\in I_S}\mathbf{Z}_i\right] \\ &\geq \left(\frac{1}{n}\right)^{|I_B|}\mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}\setminus D}}\left[\prod_{i\in I_S}\left(\frac{|\mathbf{B}_i|}{n/2}\right)\right].\end{aligned}$$

Therefore, it remains to show that

$$\mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}\setminus D}}\left[\prod_{i\in I_S}\left(\frac{2|\mathbf{B}_i|}{|A_i\cap\overline{M}|}\right)\right] \geq 1 - o(1). \tag{2.29}$$

Next we further simplify (2.29) by introducing new, simpler random variables. We may re-write

$$|\mathbf{B}_i| = \sum_{k\in(A_i\cap\overline{M})\setminus D}\mathbf{X}_{i,k}, \quad \text{where} \quad \mathbf{X}_{i,k} = \begin{cases} 1 & \text{if } y_{i,k}\oplus\mathbf{s}_k = \alpha_i \\ 0 & \text{otherwise}\end{cases}$$

For each $i \in I_S$ and $k \in (A_i \cap \overline{M}) \setminus D$, let $\mathbf{Y}_{i,k}$ and $\mathbf{Y}_i$ be the following random variables:

$$\mathbf{Y}_{i,k} = \frac{1 - 2\mathbf{X}_{i,k} + 2\tau_i}{|A_i \cap \overline{M}|} \quad \text{and} \quad \mathbf{Y}_i = \sum_{k\in A_i\cap M\setminus D}\mathbf{Y}_{i,k}, \quad \text{where} \quad \tau_i = \frac{|A_i\cap\overline{M}\cap D|}{2|(A_i\cap\overline{M})\setminus D|}.$$

(Note that $|(A_i \cap \overline{M}) \setminus D|$ is $\Omega(n)$ so τ_i's are well-defined.) A simple derivation shows

that

$$\prod_{i\in I_S}\left(\frac{2|\mathbf{B}_i|}{|A_i\cap\overline{M}|}\right)=\prod_{i\in I_S}\left(1-\sum_{k\in(A_i\cap M)\setminus D}\mathbf{Y}_{i,k}\right)=\prod_{i\in I_S}\big(1-\mathbf{Y}_i\big). \tag{2.30}$$

Using the fact that each fraction on the LHS is between 0 and 2, we have that $\mathbf{Y}_i$ always satisfies $|\mathbf{Y}_i|\le 1$. The difficulty in lowerbounding (2.30) is that $\mathbf{Y}_i$'s are not independent. But with a fixed i, $\mathbf{Y}_{i,k}$'s are indeed independent with respect to the randomness in $\mathbf{s}$, and each $\mathbf{Y}_{i,k}$ is either

$$\frac{1}{|A_i\cap\overline{M}|}+O\left(\frac{1}{n^{5/3}\log n}\right)\quad\text{or}\quad-\frac{1}{|A_i\cap\overline{M}|}+O\left(\frac{1}{n^{5/3}\log n}\right)$$

with equal probabilities, where we used the fact that $|A_i\cap\overline{M}|=\Omega(n)$ and $|D|\le n^{1/3}/\log n$.

For each $i\in I_S$, let $\mathbf{W}_i$ be the random variable defined as

$$\mathbf{W}_i=\begin{cases}\mathbf{Y}_i & \text{if } |\mathbf{Y}_i|\le \log^2 n/\sqrt{n}\\ 2|I_S| & \text{otherwise}\end{cases}$$

We prove the following claim that helps us avoid the correlation between $\mathbf{Y}_i$'s.

Claim 2.3.23. *The following inequality always holds:*

$$\prod_{i\in I_S}\big(1-\mathbf{Y}_i\big)\ge\big(1-o(1)\big)\cdot\left(1-\sum_{i\in I_S}\mathbf{W}_i\right).$$

Proof. The inequality holds trivially if $|\mathbf{Y}_j|\ge\log^2 n/\sqrt{n}$ for some $j\in I_S$. This is because $|\mathbf{Y}_i|\le 1$ and thus, the LHS is nonnegative. On the other hand $\mathbf{W}_j=2|I_S|$ implies that the RHS is negative even when every other $\mathbf{W}_i$ is -1. So we may assume that $|\mathbf{Y}_i|\le\log^2 n/\sqrt{n}$ for every i. We defer the proof for this case to Claim 5.1.1 in Section 5.1. □

Given Claim 2.3.23, it suffices to upperbound the expectation of each $\mathbf{W}_i$ over $\mathbf{s}\sim$

$\{0,1\}^{\overline{M}\setminus D}$:

$$\mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}\setminus D}}\left[\mathbf{W}_i\right] \le \mathop{\mathbf{E}}_{\mathbf{s}\sim\{0,1\}^{\overline{M}\setminus D}}\left[\mathbf{Y}_i\right] + \left(2|I_S|+1\right)\cdot \Pr_{\mathbf{s}}\left[\mathbf{Y}_i \ge \log^2 n/\sqrt{n}\right] = O\left(\frac{1}{n^{2/3}\log n}\right) \tag{2.31}$$

where we used $|I_S| \le n^{2/3}$ and that the probability of $\mathbf{Y}_i \ge \log^2 n/\sqrt{n}$ is superpolynomially small, by a Chernoff bound. Our goal, (2.29), then follows directly from (2.31) and Claim 2.3.23.

2.3.2.5 Proof of the pruning lemma

Let E be the set of bad edges in C. We start by partitioning E into three (disjoint) subsets E_1, E_2 and E_3 according the the event that happens at $(u,v) \in E$. Let $(u,v) \in E$ and let $(I_B \cup I_S; P; R; A; \rho; \delta)$ and $(I'_B \cup I'_S; P'; R'; A'; \rho'; \delta')$ be the summaries associated with u and v, respectively. Then

1. $(u,v) \in E_1$ if for some $i \in I_S$, we have $|A_i \setminus A'_i| \ge n^{2/3}\log n$;
2. $(u,v) \in E_2$ if $(u,v) \notin E_1$ and $|I'_B| \ge n^{1/3}/\log n$; or
3. $(u,v) \in E_3$ if $(u,v) \notin E_1 \cup E_2$ and for two different indices $i,j \in I'_B$, we have $\delta(i) = \delta(j)$.

Note that E_1, E_2 and E_3 are disjoint. Moreover, by the definition of bad edges none of these events happens at any edge along the root-to-u path.

Our plan is to show that the probability of $(\mathbf{T},\mathbf{H},\mathbf{s})$, as $\mathbf{T} \sim \mathcal{E}, \mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$, passing through an edge in E_i is $o(1)$ for each i. The pruning lemma then follows from a union bound.

For edge sets E_1 and E_3, we show that for any internal node u of C, the probability of $(\mathbf{T},\mathbf{H},\mathbf{s})$ taking an edge (u,v) that belongs to E_1 or E_3 is at most $o(1/q)$, conditioning on $(\mathbf{T},\mathbf{H},\mathbf{s})$ reaching u when $\mathbf{T} \sim \mathcal{E}, \mathbf{H} \sim \mathcal{E}_{\text{no}}$ and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$. This allows us to apply Lemma 2.1.3. We handle E_2 using a different argument by showing that, roughly speaking,

I_B goes up with very low probability after each round of query and thus, the probability of $|I_B|$ reaching $n^{1/3}/\log n$ is $o(1)$.

Edge Set E_1. Fix an internal node u of C. We show that the probability of $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ leaving u with an E_1-edge, conditioning on it reaching u, is $o(1/q)$. It then follows from Lemma 2.1.3 that the probability of $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ passing through an E_1-edge is $o(1)$.

Let x be the query made at u, and let $(I_B \cup I_S; P; R; A; \rho; \delta)$ be the summary associated with u. Fix an index $i \in I_S$. We upperbound by $o(1/q^2)$ the conditional probability of $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ taking an E_1-edge with $|A_i \setminus A'_i| \geq n^{2/3} \log n$. The claim follows by a union bound on $i \in I_S$ (as $|I| = O(q)$).

Note that either $A'_i = A_i$ or $A'_i = A_i \setminus \Delta$, where

$$\Delta = \big\{k \in A_i : x_k \text{ disagrees with } y_k \text{ of } y \in P_i\big\}.$$

Thus, a necessary condition for $|A_i \setminus A'_i| \geq n^{2/3} \log n$ to happen is $|\Delta| \geq n^{2/3} \log n$ and $\mathbf{T}_i(x) = 1$.

Since C is balanced, $|\Delta| \geq n^{2/3} \log n$ implies that

$$\Delta_1 = \big\{k \in A_i \cap M : x_k = 0 \text{ and } y_k = 1, y \in P_i\big\}$$

has size at least $n^{2/3} \log n/8$. On the other hand, fix any triple (T_{-i}, H, s), where T_{-i} is a tuple of $N-1$ terms with T_i missing, H is from the support of $\mathcal{E}_{\text{no}}$ and $s \in \{0,1\}^{\overline{M}}$ such that

$$\Pr_{\mathbf{T}_i}\big[((T_{-i}, \mathbf{T}_i), H, s) \text{ reaches } u\big] > 0, \tag{2.32}$$

where $\mathbf{T}_i$ is drawn by including each index in M with probability $1/\sqrt{n}$. It suffices to show that

$$\Pr_{\mathbf{T}_i}\big[((T_{-i}, \mathbf{T}_i), H, s) \text{ reaches } u \text{ and } \mathbf{T}_i(x) = 1\big] \leq o(1/q^2) \cdot \Pr_{\mathbf{T}_i}\big[((T_{-i}, \mathbf{T}_i), H, s) \text{ reaches } u\big]. \tag{2.33}$$

For this purpose, note that given (2.32), the event on the RHS of (2.33) happens at T_i if and only if T_i is a subset of $A^*_{i,1} = A_{i,1} \cap M$ and $T_i(y) = 0$ for every $y \in R_i$; we use U to denote the set of all such terms T_i (U cannot be empty by (2.32)). On the other hand, the event on the LHS of (2.33) happens if and only if T_i further avoids picking variables from Δ_1, i.e. $T_i \subseteq A^*_{i,1} \setminus \Delta_1$. We use V to denote the set of all such T_i's. To prove (2.33), note that we can take any T_i in V, add an arbitrary subset of Δ_1, and the result must be a set in U. As a result we have (note that the bound is very loose here)

$$\frac{\Pr[\mathbf{T}_i \in V]}{\Pr[\mathbf{T}_i \in U]} \leq \left(1 - \frac{1}{\sqrt{n}}\right)^{|\Delta_1|} = o(1/q^2).$$

This finishes the proof for E_1. Next we work on the edge set E_3.

Edge set E_3. Fix an internal node u of C. We show that the probability of $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ leaving u with an E_3-edge, conditioning on it reaching u, is $o(1/q)$. By definition, we can assume that there is no bad edge along the root-to-u path and thus, $|I_B| \leq n^{1/3}/\log n$ and I_B has no collision, i.e. there are no distinct $i, j \in I_B$ such that $\delta(i) = \delta(j)$. For $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ to leave u with an E_3-edge, it must be the case that some (at most two) terms are breached after the query x and a collision happens (either between a newly breached term and a term in I_B, or between the two newly breached terms).

Fix a pair (T, s), where T is from the support of $\mathcal{E}$ and $s \in \{0,1\}^{\overline{M}}$, such that $(T, \mathbf{H}, s)$ reaches u with a non-zero probability when $\mathbf{H} \sim \mathcal{E}_{\text{no}}$. It suffices to show that

$$\Pr_{\mathbf{H}}\left[(T, \mathbf{H}, s) \text{ reaches } u \text{ and a collision happens}\right] \leq o(1/q) \cdot \Pr_{\mathbf{H}}\left[(T, \mathbf{H}, s) \text{ reaches } u\right]. \tag{2.34}$$

Note that the set of (at most two) $i \in I_S$ such that x is added to P_i after it is queried is determined by T (if x starts a new P_i, then this i is safe for sure). If there exists no such i, then the probability on the LHS of (2.34) is 0 since no term is newly breached and we are done. Below we prove (2.34) for the case when $i \in I_S$ is the only index such that x is

added to P_i. The case when there are two such i's can be handled similarly.

The proof of (2.34) easily follows from the following simple but useful claim:

Claim 2.3.24. *Let T and s be such that $(T, \mathbf{H}, s)$ reaches u with non-zero probability when $\mathbf{H} \sim \mathcal{E}_{no}$. Then conditioning on reaching u, $\boldsymbol{h}_i$ has its special variable uniformly distributed in $A_i \cap \overline{M}$.*

Proof. As $i \in I_S$, P_i is consistent. For (T, H, s) to reach u, the only condition on h_i and its special variable k is that (1) if $y_k \oplus s_k = \rho_i(y)$ for some $y \in P_i$, then h_i is a dictatorship function x_k; (2) if $y_k \oplus s_k \neq \rho_i(y)$ for some $y \in P_i$, then h_i is an anti-dictatorship function $\overline{x_k}$. Given T and s, there are $|A_i \cap \overline{M}|$ choices for h_i among the $2 \cdot (n/2)$ choices and they are all equally likely. □

Our goal, (2.34), follows easily from $|A_i \cap \overline{M}| = \Omega(n)$ since $i \in I_S$, Claim 2.3.24, $|I_B| \leq n^{1/3}/\log n$, our choice of $q = n^{2/3}/\log^3 n$, and the fact that, for the event on the LHS to happen, the special variable of $\boldsymbol{h}_i$ must fall inside I_B.

Edge set E_2. Let (u, v) be a bad edge in E_2 with $|I'_B| \geq n^{1/3}/\log n$. We decompose I'_B into K and L: $i \in I'_B$ is in K if *at the edge (u', v^*) along the root-to-v path where i becomes newly breached*, we have $|A^*_i \cap \overline{M}| \leq n/10$, where A^*_i is the set at v^*, and $i \in I'_B$ is in L otherwise (i.e. $|A^*_i \cap \overline{M}| > n/10$ but P^*_i at v^* becomes inconsistent after the query at u'). The claim below shows that K is small:

Claim 2.3.25. *For every E_2-bad edge (u, v), we have $|K| \leq O(n^{1/3}/\log^2 n)$.*

Proof. Fix an $i \in K$ and let (u', v^*) be the edge along the root-to-v path where i becomes breached. Note that when A_i is first created along the path, $A_i = \overline{M}$ and $|A_i \cap \overline{M}| = n/2$ (since at that time P_i consists of a single string). As we walk down the root-to-v^* path, every time a string is added to P_i, the size of A_i can only drop by $n^{2/3} \log n$ (otherwise, this edge is an E_1-edge, contradicting with the assumption that $(u, v) \in E_2$ since edges in E_1 are excluded from E_2 and (u, v) is the first bad edge along the root-to-v path) and

thus, $|A_i \cap \overline{M}|$ can drop by at most $n^{2/3} \log n$ for each new query. Combine this with $|A_i^* \cap \overline{M}| \leq n/10$, we have that $|P_i^*|$ at v^* is at least

$$1 + \frac{n/2 - n/10}{n^{2/3} \log n} = \Omega\left(\frac{n^{1/3}}{\log n}\right).$$

Using the fact that each of the q queries can be added to at most two P_i's, we have

$$|K| \leq \frac{2q}{\Omega(n^{1/3}/\log n)} = O\left(\frac{n^{1/3}}{\log^2 n}\right).$$

This finishes the proof of the claim. □

It follows directly from Claim 2.3.25 that every bad $(u, v) \in E_2$ has $|L| \geq n^{1/3}/(2 \log n)$. This inspires us to consider the following random process of walking down the tree C from its root, with respect to $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ over $\mathbf{T} \sim \mathcal{E}$, $\mathbf{H} \sim \mathcal{E}_{\text{no}}$, and $\mathbf{s} \sim \{0,1\}^{\overline{M}}$. As we walk down an edge (u, v) of C, letting $(I_B \cup I_S; P; R; A; \rho; \delta)$ and $(I'_B \cup I'_S; P'; R'; A'; \rho'; \delta')$ be the summaries associated with u and v, if $|A_i \backslash A'_i| \geq n^{2/3} \log n$ for some $i \in I_S$, then we fail and terminate the random process; if not we add the newly breached term(s) i with $|A'_i \cap \overline{M}| > n/10$ (so P'_i becomes inconsistent), if any, to $\mathbf{L}$. We succeed if $|\mathbf{L}| \geq n^{1/3}/(2 \log n)$, and it suffices for us to show that we succeed with probability $o(1)$ over $\mathbf{T}, \mathbf{H}$ and $\mathbf{s}$.

For the analysis, let u be an internal node of C, and fix any pair (T, s) such that $(T, \mathbf{H}, s)$ can reach u with a non-zero probability. As discussed earlier, the set of (at most two) P_i, $i \in I_S$, that the query string x joins is determined only by T. If one of them has $|A_i \setminus A'_i| \geq n^{2/3} \log n$ then the process would always fail; otherwise, we have that $\mathbf{L}$ can grow by at most two and this happens with probability (over the randomness of $\mathbf{H}$ but conditioning on $(T, \mathbf{H}, s)$ reaching u) at most

$$p = O\left(\frac{n^{2/3} \log n}{n}\right) = O\left(\frac{\log n}{n^{1/3}}\right)$$

because $|A_i \cap \overline{M}| = \Omega(n)$ ($i \in I_S$), the special variable of $\boldsymbol{h}_i$ is uniform over $A_i \cap \overline{M}$ by Claim 2.3.24, and for i to be added to $\mathbf{L}$ (P'_i becomes inconsistent), the special variable of $\boldsymbol{h}_i$ must lie in $A_i \setminus A'_i$ (of size at most $n^{2/3} \log n$).

In summary, after each query the random process either fails, or if it does not fail, $\mathbf{L}$ can grow by at most two with probability at most p. Therefore, the probability that we succeed is at most

$$\mathop{\mathbf{P}r}_{\mathbf{m} \sim \mathrm{Bin}(q,p)} \left[2\mathbf{m} \geq \frac{n^{1/3}}{2 \log n} \right] = o(1),$$

where $\mathbf{m} \sim \mathrm{Bin}(q, p)$ is the Binomial distribution with q trials: for each trial $\mathbf{m}$ increases by 1 independently with probability p. The probability above is $o(1)$ since $q = n^{2/3}/\log^3 n$ and $p = O(\log n / n^{1/3})$.

This finishes the proof that $(\mathbf{T}, \mathbf{H}, \mathbf{s})$ passes through an edge in E_2 with probability $o(1)$.

2.4 An $\tilde{\Omega}(n)$ lower bound for non-adaptive testing of unateness

In this section we prove Theorem 1.2.5: an $\tilde{\Omega}(n)$ lower bound on the query complexity of testing unateness for *non-adaptive* algorithms with *one-sided* errors. Our argument is an adaptation of Theorem 19 of [Fis+02] to the setting of unateness, with one additional observation that allows us to obtain a higher lower bound. For the rest of the section, we fix $q = n/\log^2 n$.

Let's first describe a distribution $\mathcal{D}_{no}$ supported on Boolean functions f over $n + 2$ variables. We then show that every $\mathbf{f} \sim \mathcal{D}_{no}$ is $\Omega(1)$-far from unate. A function $\mathbf{f} \sim \mathcal{D}_{no}$ is drawn by first drawing an index $\boldsymbol{i} \sim [n]$ uniformly at random, and then letting $\mathbf{f} = f_{\boldsymbol{i}}$,

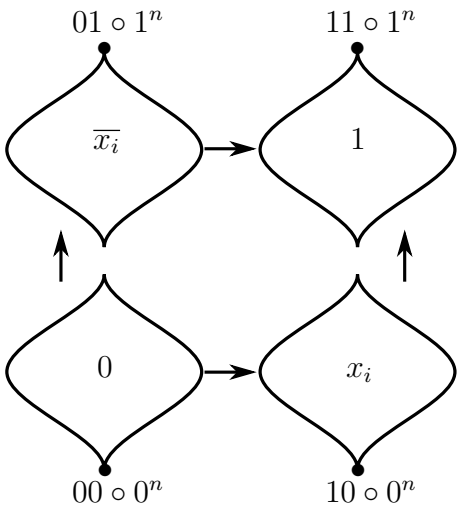

Figure 2.3: An illustration of $f_i \colon \{0,1\}^{n+2} \to \{0,1\}$. The first two bits index the subcubes.

where for each $x \in \{0,1\}^n$:

$$f_{\boldsymbol{i}}(0,0,x) = 0,$$
$$f_{\boldsymbol{i}}(0,1,x) = \overline{x_{\boldsymbol{i}}},$$
$$f_{\boldsymbol{i}}(1,0,x) = x_{\boldsymbol{i}},$$
$$f_{\boldsymbol{i}}(1,1,x) = 1.$$

In order to simplify the notation, given $a, b \in \{0,1\}$ and $i \in [n]$, we write $f_{i,ab} \colon \{0,1\}^n \to \{0,1\}$ to denote the function $f_{i,ab}(x) = f_i(a,b,x)$ that agrees with f_i when a and b are the first two bits of the input.

Figure 2.3 gives a simple visual representation of f_i. We show that f_i is the $\Omega(1)$-far from unate.

Lemma 2.4.1. *For all $i \in [n]$, f_i is $\Omega(1)$-far from unate.*

Proof. This is immediate from Lemma 2.1.2, because there are $\Omega(2^n)$ monotone edges in the $(i+2)$th direction of f_i, as well as $\Omega(2^n)$ anti-monotone edges in the $(i+2)$th direction of f_i. □

We consider *non-adaptive*, deterministic q-query algorithm B with *one-sided* errors that, given oracle access to a Boolean function f, tries to distinguish whether f is unate or far from being unate. Note that such an algorithm B simply consists of a pre-selected set of q query strings $x^1, \ldots, x^q$[9], as well as a decision procedure which outputs "accept" or "reject" given $f(x^k)$ for each $k \in [q]$. Furthermore, since B only has one-sided errors, B outputs "reject" only if it observes a *violation* to unateness (which is equivalent to what we described in the introduction), defined as follows:

Definition 2.4.2. *A* violation *to unateness for a function* $f\colon \{0,1\}^n \to \{0,1\}$ *is a function* $v\colon \{0,1\}^n \to (\{0,1\}^n)^2$*, such that for each* $r \in \{0,1\}^n$*:* $v(r) = (x, y)$ *where* $x, y \in \{0,1\}^n$ *and*

$$x \oplus r \prec y \oplus r \quad \textit{and} \quad f(x) = 1, f(y) = 0.$$

Intuitively, a violation to unateness consists of a violation to monotonicity, for every possibly orientation $r \in \{0,1\}^n$. We refer to $f^r\colon \{0,1\}^n \to \{0,1\}$ as the function $f^r(x) = f(x \oplus r)$, for any $r \in \{0,1\}^n$. So a violation to unateness for f consists of a violation to monotonicity for each f^r, $r \in \{0,1\}^n$.

Thus, we may equivalently view the algorithm B with oracle access to $f\colon \{0,1\}^n \to \{0,1\}$ works in the following way:

1. Query the oracle with queries $Q = \{x^1, \ldots, x^q\} \subset \{0,1\}^n$.
2. If there exists a violation to unateness of f, $v\colon \{0,1\}^n \to (\{0,1\}^n)^2$ where the image of v, $\{v(r)\colon r \in \{0,1\}^n\}$, is a subset of $Q \times Q$, then output "reject"; otherwise, output "accept".

Note that if B does not find a violation (in which case it always outputs "accept"), then there exists some unate function $f'\colon \{0,1\}^n \to \{0,1\}$ which is consistent with Q (i.e., $f'(x^k) = f(x^k)$ for all $k \in [q]$). In order to say that B does not find a violation, it suffices

[9]Here we use x^i to denote different strings, rather than powers of x.

to exhibit some $r \in \{0,1\}^n$ such that B does not find a violation to monotonicity of f^r. Given Lemma 2.4.1, Theorem 1.2.5 follows from the following lemma:

Lemma 2.4.3. *For any q-query non-adaptive algorithm B, there exists some $r \in \{0,1\}^{n+2}$ such that with probability $1 - o(1)$ over $\boldsymbol{i} \sim [n]$, B does not observe any violations to monotonicity of $f^r_{\boldsymbol{i}}$.*

Proof of Theorem 1.2.5 assuming Lemma 2.4.3 and Lemma 2.4.1. Lemma 2.4.3 implies that with probability $1 - o(1)$ over the draw of $\mathbf{f} \sim \mathcal{D}_{no}$, B does not observe any violation to unateness, since there is some $r \in \{0,1\}^{n+2}$ where B does not observe any violation for monotonicity of $\mathbf{f}^r$. Thus, any q-query algorithm B does not output "reject" on inputs drawn from $\mathcal{D}_{no}$ with probability at least $\frac{2}{3}$. By combining Lemma 2.4.1 this finishes the proof of Theorem 1.2.5. □

We now proceed to prove Lemma 2.4.3. For two strings $y, z \in \{0,1\}^n$, recall we write the Hamming distance between y and z as $d(y,z) = |\{k \in [n] \colon y_k \neq z_k\}|$. We first show the following lemma:

Lemma 2.4.4. *For any q strings $x^1, \ldots, x^q \in \{0,1\}^n$, there exists an $r \in \{0,1\}^n$ such that for any $j, k \in [q]$, if $x^j \oplus r \prec x^k \oplus r$, then $d(x^j, x^k) \leq 2 \log n$.*

Proof. Consider a random n-bit $\mathbf{r} \sim \{0,1\}^n$. Suppose x^j and x^k have $d(x^j, x^k) > 2 \log n$. Then:

$$\Pr_{\mathbf{r} \sim \{0,1\}^n} \left[x^j \oplus \mathbf{r} \prec x^k \oplus \mathbf{r} \right] < 2^{-2 \log n} = n^{-2},$$

since if x^j and x^k differ at index $i \in [n]$, $\mathbf{r}_i$ can only take one of two possible values to make $x^k_i \leq x^j_i$. Thus we can union bound over all possible pairs of queries with distance at least $2 \log n$ to obtain

$$\Pr_{\mathbf{r} \sim \{0,1\}^n} \left[\exists j, k \in [q], d(x^j, x^k) > 2 \log n \text{ and } x^j \oplus \mathbf{r} \prec x^k \oplus \mathbf{r} \right] < n^2 / n^2 = 1.$$

Therefore, there exists an r such that for all $j, k \in [q]$, $x^j \oplus r \prec x^k \oplus r$ implies $d(x^j, x^k) > 2 \log n$. □

Then we are ready to prove Lemma 2.4.3:

Proof of Lemma 2.4.3. Consider a non-adaptive, deterministic algorithm B making q queries $y^1, \ldots, y^q \in \{0,1\}^{n+2}$, and let $x^1, \ldots, x^q$ be the last n bits of these strings (there may be duplicates among these strings but we keep them labeled differently). We will focus on $x^1, \ldots, x^q$ and refer to the sub-functions that they query. For example x^k will query the sub-function f_{ab} corresponding to $a = y_1^k$ and $b = y_2^k$. We may partition the set of queries $Q = \{x^1, \ldots, x^q\}$, according to the sub-function queried:

$$Q_{00} = \{x^k \in Q \colon y_1^k = y_2^k = 0\}$$
$$Q_{01} = \{x^k \in Q \colon y_1^k = 0, y_2^k = 1\}$$
$$Q_{10} = \{x^k \in Q \colon y_1^k = 1, y_2^k = 0\}$$
$$Q_{11} = \{x^k \in Q \colon y_1^k = y_2^k = 1\}.$$

Let $r \in \{0,1\}^n$ be the string such that all comparable pairs among $x^1 \oplus r, \ldots, x^q \oplus r$ have distance at most $2 \log n$, which is guaranteed to exist by Lemma 2.4.4. We will show that when $r' = (0, 0, r) \in \{0,1\}^{n+2}$, with probability $1 - o(1)$ over the draw of $\boldsymbol{i} \sim [n]$, B does not observe any violation to monotonicity of $f_{\boldsymbol{i}}^{r'}$.

Consider any $i \in [n]$ and one possible violation to monotonicity, given by the pair (x^k, x^j) where

$$y^k \oplus r' \prec y^j \oplus r' \quad \text{and} \quad f_i^{r'}(y^k) = 1, f_i^{r'}(y^j) = 0$$

Then $x^k \notin Q_{00}$ and $x^j \notin Q_{11}$ since $f_{i,00}^r$ and $f_{i,11}^r$ are the constant 0 and 1 functions, respectively. Additionally, if $x^j \in Q_{00}$, then $x^k \in Q_{00}$ since $r'_1 = r'_2 = 0$, but this contradicts the fact that $f_i^{r'}(y^k) = 1$, so $x^j \notin Q_{00}$. Similarly, $x^k \notin Q_{11}$.

Additionally, if $x^k \in Q_{01}$ (or Q_{10}) and $x^j \in Q_{10}$ (or Q_{01}), y^k and y^j are incomparable, so $y^k \oplus r'$ and $y^j \oplus r'$ are incomparable. Also, for any $i \in [n]$, either $f^r_{i,01}$ or $f^r_{i,10}$ is monotone, so it suffices to consider pairs (x^k, x^j) where either both $x^k, x^j \in Q_{01}$, or both $x^k, x^j \in Q_{10}$. Consider the case $f^r_{i,10}$ is monotone (and as a result, both x^k and x^j are in Q_{01}), since the other case is symmetric. Therefore, it suffices to show that with probability $1 - o(1)$ over the choice of $\boldsymbol{i} \sim [n]$, B does not observe any violations to monotonicity for $f^r_{\boldsymbol{i},01}$ from queries in Q_{01}.

Similarly to [Fis+02], consider the graph of the queries where two queries $x^{j'}$ and $x^{k'}$ are connected if $x^{j'} \oplus r$ and $x^{k'} \oplus r$ are comparable. Additionally, consider a spanning forest T over this graph. For any $i \in [n]$, if $f^r_{i,01}(x^{j'}) \neq f^r_{i,01}(x^{k'})$ when $x^{j'}$ and $x^{k'}$ are connected in T, then there exists an edge in T, (z, w), where $f^r_{i,01}(z) \neq f^r_{i,01}(w)$. Thus, it suffices to upperbound the probability that some edge (z, w) in T has $f^r_{\boldsymbol{i},01}(z) \neq f^r_{\boldsymbol{i},01}(w)$ over the draw of $\boldsymbol{i} \sim [n]$, and this only happens when $z \oplus r$ and $w \oplus r$ differ at index $\boldsymbol{i}$.

We have:

$$\Pr_{\boldsymbol{i} \sim [n]} \left[\exists (z, w) \in T : f^r_{\boldsymbol{i},01}(z) \neq f^r_{\boldsymbol{i},01}(w) \right] \leq \frac{q \cdot 2 \log n}{n}$$

since the two end points of each edge have Hamming distance at most $2 \log n$ (recall our choice for r). We union bound over at most q edges in T to conclude that with probability at least $1 - 2q \log n/n$ over the draw of $\boldsymbol{i} \sim [n]$, B does not observes a violation to monotonicity for $f^r_{\boldsymbol{i},01}$ in Q_{01}. When $q = n/\log^2 n$, this probability is at least $1 - o(1)$. □

2.5 Disscussion

In this section, we provide some intuition why the analyses of [BB16] and this thesis for lower bound of monotonicity testing are tight. In particular, we sketch algorithms to find violating pairs in the far-from-monotone functions from the distributions considered. While these arguments imply that a new construction (of hard distributions) is needed for an improved lower bound of monotonicity testing, we haven't reached similar conclusions

regarding testing of unateness (potentially we can prove an improved lower bound with the same construction in Section 2.3 and a new analysis). Therefore we will focus on monotonicity testing in this section, and we will maintain this discussion at a high level.

2.5.1 An $O(n^{1/4})$-query algorithm for distributions of [BB16]

Belovs and Blais define a pair of distributions ${\mathcal{D}_{yes}}^*$ and ${\mathcal{D}_{no}}^*$ over n-variable Boolean functions. To describe ${\mathcal{D}_{yes}}^*$ and ${\mathcal{D}_{no}}^*$, recall Talagrand's random DNF [Tal96] (letting $N = 2^{\sqrt{n}}$): A function $\mathbf{f}$ drawn from $\mathbf{Tal}$ is the disjunction of N terms $\mathbf{T}_i$, $i \in [N]$, where each $\mathbf{T}_i$ is the conjunction of $\sqrt{n}$ variables sampled independently and uniformly from $[n]$.

Next we use $\mathbf{Tal}$ to define $\mathbf{Tal}_{\pm}$. To draw a function $\mathbf{g}$ from $\mathbf{Tal}_{\pm}$, one samples an $\mathbf{f}$ from $\mathbf{Tal}$ and a random $\sqrt{n}$-subset $\boldsymbol{S}$ of $[n]$.[10] Then $\mathbf{g}(x) = \mathbf{f}(x^{(\boldsymbol{S})})$, where $x^{(\boldsymbol{S})}$ is the string obtained from x by flipping each variable in $\boldsymbol{S}$. Equivalently variables in $\mathbf{T}_i \cap \boldsymbol{S}$ appear negated in the conjunction of $\mathbf{T}_i$. The ${\mathcal{D}_{yes}}^*$ distribution is then the truncation of $\mathbf{Tal}$, and the ${\mathcal{D}_{no}}^*$ distribution is the truncation of $\mathbf{Tal}_{\pm}$. Every $\mathbf{f} \sim {\mathcal{D}_{yes}}^*$ is monotone by definition; [BB16] shows that $\mathbf{g} \sim {\mathcal{D}_{no}}^*$ is far from monotone using the extremal noise sensitivity property of Talagrand functions [MO03].

We now sketch an $O(n^{1/4})$-query algorithm that rejects $\mathbf{g} \sim {\mathcal{D}_{no}}^*$ with high probability. Note that the description below is not a formal analysis; the goal is to discuss the main idea behind the algorithm. Let g be a function in the support of ${\mathcal{D}_{no}}^*$ defined by T_i and S with $T_i' = T_i \setminus S$. Then the algorithm starts by sampling a random string $\mathbf{x} \in \{0,1\}^n$ in the middle layers with $g(\mathbf{x}) = 1$. It is likely ($\Omega(1)$ probability by a simple calculation) that:

1. $\mathbf{x}$ satisfies a unique term T_k' among all T_i''s.
2. $T_k \cap S$ contains a unique $\ell \in [n]$.

[10] Formally, $\boldsymbol{S}$ is sampled by including each element of $[n]$ independently with probability $1/\sqrt{n}$.

3. $T_k = T'_k \cup \{\ell\}$ and $\mathbf{x}$ has $\mathbf{x}_\ell = 0$ (since $g(\mathbf{x}) = 1$).

Fix $\mathbf{x} = x$ that satisfies these properties, and let A_0 and A_1 denote the set of 0-indices and 1-indices of x, respectively. Then $T'_k \subseteq A_1$ and $\ell \in A_0$.

The first stage of the algorithm goes as follows:

> **Stage 1.** Repeat the following for $n^{1/4}$ times: Pick a random subset $\mathbf{R} \subset A_1$ of size $\sqrt{n}$ and query $g(x^{(\mathbf{R})})$. By 1) and 2) above, $g(x^{(\mathbf{R})}) = 1$ if and only if $\mathbf{R} \cap T'_k = \emptyset$, which happens with $\Omega(1)$ probability. Let $\mathbf{A}'_1$ denote A_1 after removing those indices of $\mathbf{R}$ with $g(x^{(\mathbf{R})}) = 1$ encountered. Then we have $T'_k \subset \mathbf{A}'_1$ and most likely, $C = A_1 \setminus \mathbf{A}'_1$ has size $\Theta(n^{3/4})$.

After the first stage, with high probability the algorithm has shrunk A_1 by $\Theta(n^{3/4})$ while still making sure that variables of T'_k lie in the new (smaller) A_1. Again, assume such event happens and fix such smaller A_1 (as well as the set of variables C removed from A_1). In the second stage, the algorithm takes advantage of the smaller A_1 to search for ℓ in A_0, with each query essentially covering $\Theta(n^{3/4})$ indices of A_0:

> **Stage 2.** Randomly partition A_0 into $O(n^{1/4})$ many disjoint parts $\mathbf{A}_{0,1}, \mathbf{A}_{0,2}, \ldots$, each of size $|C| = \Theta(n^{3/4})$. For each $\mathbf{A}_{0,j}$, query $g(x^{(\mathbf{A}_{0,j} \cup C)})$. For each $\mathbf{A}_{0,j}$ with $\ell \notin \mathbf{A}_{0,j}$, g must return 1; for the $\mathbf{A}_{0,h}$ with $\ell \in \mathbf{A}_{0,h}$, g returns 0 with $\Omega(1)$ probability[11] and when this happens, the algorithm has found an $O(n^{3/4})$-size subset $A_{0,h}$ of A_0 containing ℓ. Let $y = x^{(A_{0,h} \cup C)}$, and we have $g(y) = 0$.

Note that the algorithm cannot directly query $g(x^{(A_{0,h})})$ since the new string will be outside of the middle layers (unless $|A_{0,h}| = O(\sqrt{n})$, in which case one needs $\Omega(\sqrt{n})$ queries to cover A_0). This is only achieved by flipping $A_{0,h}$ and C at the same time but in opposite directions (and that's why we need the first stage to shrink A_1 and get C). In the last stage, the algorithm will find a violation to monotonicity of g, by providing $z \prec y$ with $g(z) = 1$.

[11] Informally speaking, this is because the values of $g(x)$ and $g(y)$ essentially become independent when x and y are far from each other.

Stage 3. Randomly partition $A_{0,h}$ into $O(n^{1/4})$ many disjoint parts $\mathbf{\Delta}_1, \mathbf{\Delta}_2, \ldots$, each of size $O(\sqrt{n})$. For each $\mathbf{\Delta}_i$, query $g(y^{(\mathbf{\Delta}_i)})$. There always exists some $\mathbf{\Delta}_i \ni \ell$, and we have $g(y^{(\mathbf{\Delta}_i)}) = 1$ with probability $\Omega(1)$. When this happens for $\mathbf{\Delta}_i = \Delta_i$, we have $z = y^{(\Delta_i)} \prec y$ and $g(z) = 1$, as desired.

2.5.2 An $O(n^{1/3})$-query algorithm for our distributions

The idea sketched above can be applied to our far from monotone distribution $\mathcal{D}_{no}$ from Section 2.2. It is slightly more complicated, since now the algorithm must attack two levels of Talagrand functions, which will incur the query cost of $\tilde{O}(n^{1/3})$ rather than $O(n^{1/4})$. Similarly to Section 2.5.1 above, we will give a high level description rather than a formal analysis. The goal is to show the main obstacle one faces in improving the lower bound.

Assume g is in the support of $\mathcal{D}_{no}$. The algorithm works in stages and follows a similar pattern to the one described in Section 2.5.1 above. We may assume the algorithm has a string $x \in \{0,1\}^n$ where x satisfies a unique term T_i, and falsifies no clauses, so $g(x) = 1$ (this happens with $\Omega(1)$ probability for a random string $\mathbf{x}$).

Stage 1. Repeat the following for $n^{1/3}$ times: pick a random subset $\mathbf{R} \subset A_1$ of size $\sqrt{n}$ and query $g(x^{(\mathbf{R})})$. Let $\mathbf{A}_1'$ denote A_1 after removing those indices of $\mathbf{R}$ with $g(x^{(\mathbf{R})})) = 1$ encountered. Then we have $T_i \subset \mathbf{A}_1'$ and most likely, $\mathbf{C}_1 = A_1 \setminus \mathbf{A}_1'$ has size $\Theta(n^{5/6})$. Suppose this is the case and let's fix the new (smaller) set $\mathbf{A}_1 = A_1$ (as well as C_1).

The following stage will be repeated for $n^{1/6}$ many times, and each makes $n^{1/6}$ many queries.

Stage 2. Pick a random subset $\mathbf{C}_0 \subset A_0$ of size $n^{5/6}$. Let $\mathbf{y} = x^{(C_1 \cup \mathbf{C}_0)}$ and query $g(\mathbf{y})$. With probability $\Omega(1)$, $g(\mathbf{y})$ satisfies the unique term T_i (as did x), falsifies a unique clause $C_{i,j}$, and $g(\mathbf{y}) = h_{i,j}(\mathbf{y}) = 0$. Additionally, with probability $\Omega(n^{-1/6})$, $h_{i,j}(\mathbf{y}) = \overline{\mathbf{y}_{\boldsymbol{\ell}}}$, where $\boldsymbol{\ell} \in \mathbf{C}_0$.

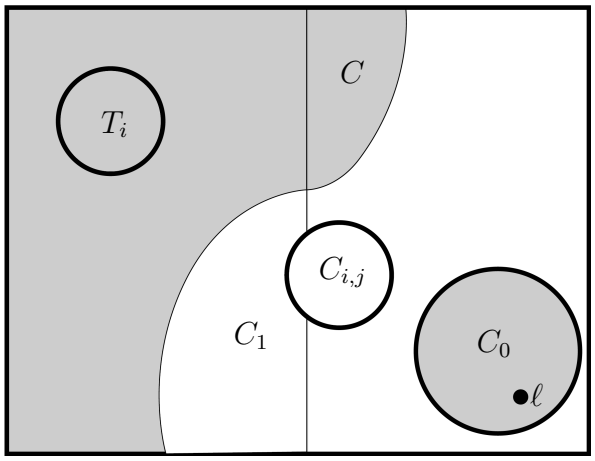

Figure 2.4: A visual representation of the algorithm for finding violations in the two-level Talagrand construction. The whole rectangle represents the set $[n]$, which is shaded for indices which are set to 1, and clear for indices which are set to 0 (at the beginning of Stage 1, indices on the left of the centerline are set to 1's, and indices on the right are set to 0's). T_i is the unique term satisfied and $C_{i,j}$ is the unique clause falsified. The functions $h_{i,j}$ is an anti-dictator of index ℓ. The sets illustrated represent the current knowledge at the end of Stage 3 of the algorithm. Note that $|C_1| = \Theta(n^{5/6})$, $|C| = \Theta(n^{2/3})$, $|C_0| = n^{5/6}$, $|T_i| = |C_{i,j}| = \Theta(\sqrt{n})$.

Overall the event $\boldsymbol{\ell} \in \mathbf{C}_0$ happens with probability $\Omega(1)$ since we repeat Stage 2 for $n^{1/6}$ times. Fixing C_0, y and ℓ such that the above event happens, we will likely find a violation with help of them as follows:

Stage 3. Repeat the following for $n^{1/6}$ times: pick a random subset $\mathbf{R} \subset A_0 \setminus C_0$ of size $\sqrt{n}$ and query $g(y^{(\mathbf{R})})$. Let $\mathbf{A}'_0$ denote $A_0 \setminus C_0$ after removing those indices of $\mathbf{R}$ with $g(y^{(\mathbf{R})}) = 0$. Let $\mathbf{C} = (A_0 \setminus C_0) \setminus \mathbf{A}'_0$, where very likely $|\mathbf{C}| = \Theta(n^{2/3})$. Fix $\mathbf{C} = C$ such that it's of size $\Theta(n^{2/3})$. Our sets satisfy the following three conditions: 1) $T_i \subset A_1$, 2) $C_{i,j} \subset A'_0 \cup C_1 \setminus C_0$, and 3) $\ell \in C_0$. See Figure 2.4 for a visual representation of these sets.

Stage 4. Partition C_0 into $O(n^{1/6})$ many disjoint parts $C_{0,1}, C_{0,2}, \ldots$, each of size $\Theta(n^{2/3})$ and query $g(y^{(C_{0,j} \cup C)})$. For each $C_{0,j}$ with $\ell \notin C_{0,j}$ and no new terms are satisfied, g must return 0. If for some sets $C_{0,j}$, g returns 1, then either $\ell \in C_{0,j}$ and

no new terms are satisfied, or new terms are satisfied; however, we can easily distinguish these cases with a statistical test.

The final stage is very similar to the final stage of Section 2.5.1. After Stage 4, we assume we have found a set $C_{0,j}$ containing ℓ. We further partition $C_{0,j}$ (when $g(y^{(C_{0,j} \cup C)}) = 1$) into $O(n^{1/6})$ parts of size $\sqrt{n}$ to find a violation. One can easily generalize the above algorithm sketch to $O(1)$-many levels of Talagrand. This suggests that the simple extension of our construction to $O(1)$ many levels (which still gives a far-from-monotone function) cannot achieve lower bounds better than $\tilde{\Omega}(n^{1/3})$.

This page intentionally left blank.

Chapter 3

Distribution-free Testing of k-juntas

In this chapter we will start to discuss about the more general problem of distribution-free testing. As the first example, we will study testing of k-juntas under this setting. We will present an adaptive distribution-free tester for k-juntas with query complexity $\tilde{O}(k^2/\epsilon)$ (independent of n), and also show that any non-adaptive distribution-free testers for k-juntas must make at least $\tilde{\Omega}(2^{k/3})$ queries (for some constant distance parameter ϵ). Combining these two results together we know adaptivity provides an exponential improvement (in terms of query complexity) for this problem, which stands in sharp contrast to the standard uniform distribution testing of k-juntas. More formally, we will start with some preparation in Section 3.1. Then we will present our algorithm and prove Theorem 1.3.2 in Section 3.2, and present the proof of our lower bound, stated as Theorem 1.3.3, in Section 3.3.

3.1 Preparation

In this section we give some formal definitions and notation that will be useful.

We study the distribution-free testing of k-juntas in this chapter. Recall that a Boolean function f is a k-junta if it depends on at most k variables. More precisely, f is a k-junta if there exists a subset $J = \{i_1, \ldots, i_k\} \subset [n]$ of size k and a Boolean function $g : \{0,1\}^k \to \{0,1\}$ over k variables such that $f(x_1, \ldots, x_n) = g(x_{i_1}, \ldots, x_{i_k})$ for all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$.

We say that a Boolean function f is a *literal* if f depends on exactly one variable, i.e.

Procedure BinarySearch(f, x, y)
Input: Black-box oracle access to $f\colon \{0,1\}^n \to \{0,1\}$ and two strings $x, y \in \{0,1\}^n$ with $f(x) \neq f(y)$.
Output: Two strings $x', y' \in \{0,1\}^n$ with $f(x') \neq f(y')$ and $x' = y'^{(i)}$ for some $i \in \mathsf{diff}(x, y)$.

1. Let $B = \mathsf{diff}(x, y) \subseteq [n]$.
2. If $|B| = 1$ return x and y.
3. Partition (arbitrarily) B into B_1 and B_2 of size $\lfloor |B|/2 \rfloor$ and $\lceil |B|/2 \rceil$, respectively.
4. Query $f(x^{(B_1)})$.
5. If $f(x) \neq f(x^{(B_1)})$, return **BinarySearch**$(f, x, x^{(B_1)})$.
6. Otherwise, return **BinarySearch**$(f, x^{(B_1)}, y)$.

Figure 3.1: Description of the standard binary search procedure.

for some $i \in [n]$, we have that either $f(x) = x_i$ for all x or $f(x) = \overline{x_i}$ for all x. Note that the two constant (all-1 and all-0) functions are one-juntas but are *not* literals.

We will often work with restrictions of Boolean functions. Given $f : \{0,1\}^n \to \{0,1\}$, $B \subseteq [n]$ and a string $z \in \{0,1\}^B$, the *restriction of f over B by z*, denoted by $f\restriction_z$, is the Boolean function $g : \{0,1\}^{\overline{B}} \to \{0,1\}$ defined by $g(x) = f(x \circ z)$ for all $x \in \{0,1\}^{\overline{B}}$. We will also use the term "*block*" to refer to a *nonempty* subset of $[n]$, which should be interpreted as a nonempty subset of the n input variables of a Boolean function $f : \{0,1\}^n \to \{0,1\}$. The following definition of distinguishing pairs and relevant blocks will be heavily used in our algorithms.

Definition 3.1.1 (Distinguishing pairs and relevant blocks). *Given $x, y \in \{0,1\}^n$, a Boolean function f, and a block $B \subseteq [n]$, we say that (x, y) is a* distinguishing pair of f for B *if $x_{\overline{B}} = y_{\overline{B}}$ (and therefore $\mathsf{diff}(x, y) \subset B$) and $f(x) \neq f(y)$. We say B is a* relevant *block of f if such a distinguishing pair exists for B (or equivalently, the influence of B in f is positive). When $B = \{i\}$ is a relevant block we simply say that the ith variable is a relevant variable of f.*

Clearly an empty set B can never has a distinguish pair, and from now on whenever

we say there is a set with a distinguish pair, it can be assumed to be a (nonempty) block automatically.

An important ingredient of our algorithms is the following binary search procedure (see Figure 3.1): it takes as input two strings $x, y \in \{0,1\}^n$ with $f(x) \neq f(y)$, makes $O(\log n)$ queries on f, and returns a pair of strings $x', y' \in \{0,1\}^n$ with $f(x') \neq f(y')$ and $x' = y'^{(i)}$ for some $i \in \mathsf{diff}(x, y)$, i.e., a distinguishing pair for the ith variable for some $i \in \mathsf{diff}(x, y)$.

3.2 An $\tilde{O}(k^2/\epsilon)$-query distribution-free testers of k-juntas

In this section, we present an adaptive distribution-free tester of k-juntas with query complexity $\tilde{O}(k^2/\epsilon)$ and prove Theorem 1.3.2. We start with some high level ideas in Section 3.2.1. Then as a warm-up we present a simple algorithm called **SimpleDJunta** (with query complexity that depends on n) for our task in Section 3.2.2. Based on ideas from **SimpleDJunta** and the uniform distribution tester of [Bla09] we give more intuition behind our main algorithm **MainDJunta** with query complexity $\tilde{O}(k^2/\epsilon)$ in Section 3.2.3. In the end we present our main algorithm **MainDJunta** and proof of its correctness in Sections 3.2.4, 3.2.5 and 3.2.6.

3.2.1 High level ideas

We give some high level ideas behind our algorithms in this section.

It will become clear later that all our algorithms reject a function f only when they have found $k+1$ distinct relevant variables (or disjoint relevant blocks). When this happens, it means that f cannot be a k-junta. Therefore our algorithms only has one-sided errors, and here for intuition we will focus on showing that they reject the input pair $(f, \mathcal{D})$ with high probability when f is ϵ-far from k-juntas with respect to $\mathcal{D}$.

As a first step toward our $\tilde{O}(k^2/\epsilon)$-query main algorithm, in Section 3.2.2 we first present a simple adaptive algorithm called **SimpleDJunta**, that distribution-free tests k-juntas using $O(k/\epsilon + k \log n)$ queries. **SimpleDJunta** uses binary search and is an adaptation to the distribution-free setting from the uniform-distribution algorithm of [Bla09] with some simplification (we conduct standard bit-wise binary search rather than block-wise binary search that is much harder to analyze and will be discussed later). The algorithm maintains a set I of *relevant* variables: a string $x \in \{0,1\}^n$ has been found for each $i \in I$ such that $f(x) \neq f(x^{(i)})$, and the algorithm rejects only when $|I|$ becomes larger than k. In each round, the algorithm samples a string $\mathbf{x} \sim \mathcal{D}$ and a random subset $\mathbf{R} \subset \overline{I}$ uniformly at random (by including each variable in $\overline{I}$ into $\mathbf{R}$ with probability 1/2 independently). We will show in Lemma 3.2.3 that, if f is far from k-juntas with respect to $\mathcal{D}$, then $f(\mathbf{x}) \neq f(\mathbf{x}^{(\mathbf{R})})$ with high probability as long as $|I| \leq k$. With such a pair $(\mathbf{x}, \mathbf{x}^{(\mathbf{R})})$ in hand, it is straightforward to find a new relevant variable using binary search over indices in $\mathbf{R}$ (see Figure 3.1), with at most $\log n$ additional queries. Then we can show with high probability the size of I will continue to grow until $|I| > k$ and the algorithm will reject.

As discussed in the introduction the more interesting case for testing k-juntas is when $k << n$, and we would prefer to have algorithms with query complexity that is independent of n. In order to achieve this, clearly one must employ a more efficient approach than binary search over $\Omega(n)$ indices (since most likely the set $\mathbf{R}$ has size $\Omega(n)$ for the range of k we are interested in). In the uniform-distribution setting this is accomplished in [Bla09] by first randomly partitioning the variable space $[n]$ into $s = \text{poly}(k/\epsilon)$ disjoint blocks $B_1, \ldots, B_s$ of variables and carrying out binary search over blocks (see Figure 3.2) rather than over individual indices; this reduces the cost of each binary search to $\log(k/\epsilon)$. The algorithm maintains a set of *relevant blocks*: two strings $x, y \in \{0,1\}^n$ have been found for each such block B which satisfy $f(x) \neq f(y)$ and $\mathsf{diff}(x,y) \subseteq B$, and the algorithm rejects when more than k relevant blocks have been found. In each round the algorithm samples

two strings $\mathbf{x}, \mathbf{y}$ uniformly at random conditioned on their agreeing with each other on the relevant blocks that have already been found in previous rounds; if $f(\mathbf{x}) \neq f(\mathbf{y})$, then the binary search over blocks is performed to find a new relevant block. To establish the correctness of this approach [Bla09] employs a detailed and technical analytic argument based on the influence of indices and the Efron-Stein orthogonal decomposition of functions over product spaces. This machinery is well suited for dealing with product distributions, and indeed the analysis of [Bla09] goes through for any product distribution over $\{0,1\}^n$ (and even for more general finite domains and ranges). However, it is far from clear how to extend this machinery to work for the completely unstructured distributions $\mathcal{D}$ that must be handled in the distribution-free model.

Procedure BlockBinarySearch$(f, x, y; B_1, \ldots, B_r)$
Input: Black-box oracle access to $f\colon \{0,1\}^n \to \{0,1\}$, two strings $x, y \in \{0,1\}^n$ with $f(x) \neq f(y)$, and a sequence of pairwise disjoint blocks $B_1, \ldots, B_r$ for some $r \geq 1$ with $\mathsf{diff}(x, y) \subseteq B_1 \cup \cdots \cup B_r$.
Output: Two strings $x', y' \in \{0,1\}^n$ with $f(x') \neq f(y')$ and $\mathsf{diff}(x, y) \subseteq B_i$ for some $i \in [r]$.

1. If $r = 1$ return x and y.
2. Let $t = \lfloor r/2 \rfloor$ and B be the intersection of $\mathsf{diff}(x, y)$ and $B_1 \cup \cdots \cup B_t$.
3. Query $f(x^{(B)})$.
4. If $f(x) \neq f(x^{(B)})$, return **BlockBinarySearch**$(f, x, x^{(B)}; B_1, \ldots, B_t)$.
5. Otherwise, return **BlockBinarySearch**$(f, x^{(B)}, y; B_{t+1}, \ldots, B_r)$.

Figure 3.2: Description of the blockwise version of the binary search procedure.

The main result from [Bla09] is stated as the following theorem (and we will use it later):

Theorem 3.2.1. *There exists an $O((k/\epsilon) + k \log k)$-query algorithm **UniformJunta**(f, k, ϵ) with one-sided errors that, upon black-box oracle access to a Boolean function f, rejects with probability at least 2/3 when f is ϵ-far from k-juntas under the uniform distribution. Moreover, it rejects only when it has found $k+1$ pairwise disjoint blocks and a distinguishing pair of f for each of them.*

Our main distribution-free junta testing algorithm, denoted as **MainDJunta**, draws ideas from both **SimpleDJunta** and the uniform-distribution tester of [Bla09]. To avoid the $\log n$ cost, the algorithm carries out binary search over blocks rather than over individual indices, and maintains a set of disjoint relevant blocks $B_1, \ldots, B_\ell$, i.e., for each B_j a pair of strings x^j and y^j have been found such that they agree with each other over variables in $\overline{B_j}$ and satisfy $f(x^j) \neq f(y^j)$. Let w^j be the projection of x^j (and y^j) over $\overline{B_j}$ and let g_j be the Boolean function over $\{0,1\}^{B_j}$ obtained from f by setting variables in $\overline{B_j}$ to w^j. For clarity we assume further that every function g_j is very close to a *literal* (i.e. there exists some $i_j \in B_j$ and $\tau \in \{x_{i_j}, \overline{x_{i_j}}\}$ such that $g_j(x) = \tau$ for all $x \in \{0,1\}^{B_j}$) under the *uniform* distribution. (To justify this assumption we note that if g_j is far from every literal under the uniform distribution, then it is easy to split B_j further into two relevant blocks using the uniform-distribution algorithm of [Bla09].) Let $I = \{i_j : j \in [\ell]\}$. Even though the algorithm does not know I, there is indeed a way to draw uniformly random subsets $\mathbf{R}$ of $\overline{I}$. First we draw a partition of B_j into $\mathbf{P}_j$ and $\mathbf{Q}_j$ uniformly at random, for each j. Since g_j is close to a literal, it is not difficult to figure out whether $\mathbf{P}_j$ or $\mathbf{Q}_j$ contains the hidden variable (with index) i_j. Let's assume $i_j \in \mathbf{P}_j$ for every j. Then the union of all $\mathbf{Q}_j$'s together with a uniformly random subset of $\overline{B_1 \cup \cdots \cup B_\ell}$, denoted by $\mathbf{R}$, turns out to be a uniformly random subset of $\overline{I}$. With $\mathbf{R}$ in hand, Lemma 3.2.3 for **SimpleDJunta** implies that $f(\mathbf{x}) \neq f(\mathbf{x}^{(\mathbf{R})})$ with high probability when $\mathbf{x} \sim \mathcal{D}$, and when this happens, the algorithm can carry out binary search over blocks to increase the number of relevant blocks by one. We can then show with high probability the number of relevant blocks we found will continue to grow until there are more than k such blocks (corresponding to $|I| > k$), in which case the algorithm will reject. In Section 3.2.3 we will explain the intuition behind the main algorithm in more detail.

3.2.2 Warmup: an $O(k/\epsilon + k \log n)$-query tester

As a warmup, we present in this section a simple distribution-free algorithm **SimpleDJunta** for testing k-juntas. It uses $O(k/\epsilon + k \log n)$ queries and only has one-sided errors. The idea behind **SimpleDJunta** and its analysis (Lemma 3.2.3) will be useful in the next few sections where we present our main algorithm to remove the dependency of query complexity on n.

The algorithm **SimpleDJunta** maintains a set $I \subset [n]$ such that a distinguishing pair has been found for each $i \in I$ (i.e., I is a set of relevant variables of f discovered so far). The algorithm sets $I = \emptyset$ at the beginning and rejects only when $|I|$ reaches $k+1$, which implies immediately that f cannot be a k-junta. Therefore the algorithm only has one-sided errors. **SimpleDJunta** proceeds round by round: in each round it draws a pair of random strings $\mathbf{x}$ and $\mathbf{y}$ with $\mathbf{x}_I = \mathbf{y}_I$, and if $f(\mathbf{x}) \neq f(\mathbf{y})$, the standard binary search procedure is used on $\mathbf{x}$ and $\mathbf{y}$ to find a distinguishing pair for a new relevant variable (with index) $i \in \overline{I}$, which is then added to I.

We present the description of this algorithm in Figure 3.3.

Algorithm SimpleDJunta$(f, \mathcal{D}, k, \epsilon)$
Input: Black-box oracle access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, sampling access to a probability distribution $\mathcal{D}$ over $\{0,1\}^n$, a positive integer k, and a distance parameter $\epsilon > 0$.
Output: Either "accept" or "reject."

1. Set $I = \emptyset$.
2. Repeat $8(k+1)/\epsilon$ times:
   - 2.1 Sample $\mathbf{x} \sim \mathcal{D}$ and a subset $\mathbf{R}$ of $\overline{I}$ uniformly at random. Set $\mathbf{y} = \mathbf{x}^{(\mathbf{R})}$.
   - 2.2 If $f(\mathbf{x}) \neq f(\mathbf{y})$, then run the standard binary search on $\mathbf{x}, \mathbf{y}$ to find a distinguishing pair for a new relevant variable $i \in \mathbf{R} \subseteq \overline{I}$. Set $I = I \cup \{i\}$.
   - 2.3 If $|I| > k$, then halt and output "reject."
3. Halt and output "accept."

Figure 3.3: Description of the distribution-free testing algorithm **SimpleDJunta** for k-juntas.

The following theorem establishes the correctness of this algorithm.

Theorem 3.2.2. *(i) The algorithm* **SimpleDJunta** *makes $O(k/\epsilon + k\log n)$ queries and always accepts when f is a k-junta. (ii) It rejects with probability at least $2/3$ if f is ϵ-far from k-juntas with respect to $\mathcal{D}$.*

Proof. For part (i), note that the algorithm has k/ϵ rounds but only runs binary search (and costs $O(\log n)$ queries) at most $k+1$ times, which implies it makes at most $O(k/\epsilon + k\log n)$ queries in total. The rest of part (i) is immediate from the description of the algorithm.

For part (ii), it suffices to show that when $|I| \leq k$ at the beginning of a round, a new relevant variable is discovered in this round with high probability. For this purpose we use the following simple but crucial lemma and note the fact that $\mathbf{x}$ and $\mathbf{y}$ in step 2.1 of **SimpleDJunta** can be equivalently drawn by first sampling $\mathbf{x} \sim \mathcal{D}$ and $\mathbf{w} \sim \{0,1\}^n$ and then setting $\mathbf{y} = \mathbf{x}_I \circ \mathbf{w}_{\overline{I}}$ (the way we draw $\mathbf{x}$ and $\mathbf{y}$ in Figure 3.3 via $\mathbf{R} \sim \overline{I}$ makes it easier to connect with the main algorithm in the next section).

Lemma 3.2.3. *If f is ϵ-far from k-juntas with respect to $\mathcal{D}$, then for any $I \subset [n]$ of size at most k, we have*

$$\Pr_{\mathbf{x}\sim\mathcal{D},\mathbf{w}\sim\{0,1\}^n}\left[f(\mathbf{x}) \neq f(\mathbf{x}_I \circ \mathbf{w}_{\overline{I}})\right] \geq \epsilon/2. \tag{3.1}$$

Before proving Lemma 3.2.3, we use it to finish the proof of part (ii). Assuming Lemma 3.2.3 and that f is ϵ-far from k-juntas with respect to $\mathcal{D}$, for each round with $|I| \leq k$ the algorithm finds a new relevant variable with probability at least $\epsilon/2$. Using a coupling argument, the probability that the algorithm rejects f (i.e., $|I|$ reaches $k+1$ during the $8(k+1)/\epsilon$ rounds) is at least the probability that

$$\sum_{i=1}^{8(k+1)/\epsilon} \mathbf{Z}_i \geq k+1,$$

where $\mathbf{Z}_i$'s are i.i.d. $\{0,1\}$-variables that are 1 with probability $\epsilon/2$. It follows from the Chernoff bound that the latter probability is at least $2/3$. This finishes the proof of the

theorem. □

Proof of Lemma 3.2.3. Let I be a subset of $[n]$ of size at most k. To prove (3.1) for I, we use I and f to define the following Boolean function $h : \{0,1\}^n \to \{0,1\}$: for each $x \in \{0,1\}^n$ we set:

$$h(x) := \arg\max_{b \in \{0,1\}} \left\{ \Pr_{\mathbf{w} \sim \{0,1\}^n} \left[f(x_I \circ \mathbf{w}_{\bar{I}}) = b \right] \right\},$$

where we break ties arbitrarily. Then for any $x \in \{0,1\}^n$, we have:

$$\Pr_{\mathbf{w} \sim \{0,1\}^n} \left[f(x_I \circ \mathbf{w}_{\bar{I}}) = h(x) \right] \geq 1/2. \tag{3.2}$$

Based on this we then know that:

$$\begin{aligned}
&\Pr_{\mathbf{x} \sim \mathcal{D}, \mathbf{w} \sim \{0,1\}^n} \left[f(\mathbf{x}) \neq f(\mathbf{x}_I \circ \mathbf{w}_{\bar{I}}) \right] \\
=& \sum_{z \in \{0,1\}^n} \Pr_{\mathbf{x} \sim \mathcal{D}} \left[\mathbf{x} = z \right] \cdot \Pr_{\mathbf{w} \sim \{0,1\}^n} \left[f(z) \neq f(z_I \circ \mathbf{w}_{\bar{I}}) \right] \\
\geq& \sum_{z \in \{0,1\}^n} \Pr_{\mathbf{x} \sim \mathcal{D}} \left[\mathbf{x} = z \right] \cdot \Big((1/2) \cdot \mathbf{1}\big[f(z) \neq h(z) \big] \Big) \\
=& (1/2) \cdot \Pr_{\mathbf{x} \sim \mathcal{D}} \left[f(\mathbf{x}) \neq h(\mathbf{x}) \right] \geq \epsilon/2,
\end{aligned}$$

where the first inequality follows from (3.2) and the second inequality follows from the assumption that f is ϵ-far from k-juntas with respect to $\mathcal{D}$ and the fact that h is a k-junta (since it only depends on variables in I and $|I| \leq k$). This finishes the proof of this lemma. □

3.2.3 Intuition and preparation for an $\tilde{O}(k^2/\epsilon)$-query tester

Based on ideas of **SimpleDJunta** from the previous section, we give some more intuition behind our main algorithm in this section. Again our main algorithm only has one-sided

errors, and in this section we only focus on showing the algorithm rejects the input pair $(f, \mathcal{D})$ with high probability when f is ϵ-far from k-juntas with respect to $\mathcal{D}$.

Recall that the factor of $\log n$ in the query complexity of **SimpleDJunta** is due to the use of the standard binary search procedure. To avoid it, one could choose to terminate each call to binary search early (just like the algorithm of [Bla09]) but this ends up giving us relevant *blocks* of variables instead of exact relevant variables. To highlight the challenge, imagine that the algorithm has found so far $\ell \leq k$ many pairwise disjoint relevant blocks B_j, $j \in [\ell]$, i.e., it has found a distinguishing pair for each block B_j. By definition, each B_j must contain at least one relevant variable $i_j \in B_j$. However, we do not know exactly which variable in B_j is i_j, and thus it is not clear how to draw a set $\mathbf{R}$ from $\overline{I}$ uniformly at random, where $I = \{i_j : j \in [\ell]\}$ (as what we did in **SimpleDJunta**), in order to apply Lemma 3.2.3 to discover a new relevant block. It seems that we are facing a dilemma when trying to improve **SimpleDJunta** and remove the $\log n$ factor: on one hand, unless we pin down a set of relevant variables (the set I), it is not clear how to draw a random set from its complement; on the other hand, pinning down a single relevant variable using the standard binary search procedure would already cost $\log n$ queries.

To explain the main idea behind our $\tilde{O}(k^2/\epsilon)$-query algorithm, let's assume again that $\ell \leq k$ many disjoint relevant blocks $B_1, \ldots, B_\ell$ have been found so far, and we have a distinguishing pair (x^j, y^j) for each B_j, $j \in [\ell]$ (satisfying that $\mathsf{diff}(x^j, y^j) \subseteq B_j$ and $f(x^j) \neq f(y^j)$ by definition). Let

$$w^j = \left(x^j\right)_{\overline{B_j}} = \left(y^j\right)_{\overline{B_j}} \in \{0,1\}^{\overline{B_j}}.$$

Next let us assume further that for each $j \in [\ell]$ the function $g_j := f|_{w^j}$ is a *literal*, i.e. either $g_j(z) = z_{i_j}$ for all $z \in \{0,1\}^{B_j}$ or $g_j(z) = \overline{z_{i_j}}$ for all $z \in \{0,1\}^{B_j}$, for some unknown variable $i_j \in B_j$. (While this may seems very implausible, we make this assumption for now and explain below why it is not too far from real situations after excluding some

corner cases.)

To make progress, we draw a random two-way partition of each B_j into $\mathbf{P}_j$ and $\mathbf{Q}_j$, i.e., each $i \in B_j$ is added to $\mathbf{P}_j$ or $\mathbf{Q}_j$ with probability $1/2$ (so $\mathbf{P}_j$ and $\mathbf{Q}_j$ are disjoint and $B_j = \mathbf{P}_j \cup \mathbf{Q}_j$). $\mathbf{P}_j$ or $\mathbf{Q}_j$ can be empty, but it will become clear later that this won't affect our algorithm and analysis since we will only be dealing with sets with distinguish pairs, which are guaranteed to be non-empty. We make three simple but crucial observations to increase the number of disjoint relevant blocks by one.

1. Since g_j is assumed to be a literal on the i_j-th variable (and by the definition of g_j we have query access to g_j), it is easy to tell whether $i_j \in \mathbf{P}_j$ or $i_j \in \mathbf{Q}_j$, simply by picking an arbitrary string $x \in \{0,1\}^{B_j}$ and comparing $g_j(x)$ with $g_j(x^{(\mathbf{P}_j)})$. The following subroutine **WhereIsTheLiteral** will be used to distinguish the two cases in our main algorithm:

Subroutine WhereIsTheLiteral(g, P, Q)
Input: Black-box oracle access to a Boolean function g over $\{0,1\}^B$ with P, Q being a partition of B.
Output: Either a distinguishing pair of g for P, a distinguishing pair for Q, or "fail."

1. Draw $\mathbf{x} \sim \{0,1\}^B$ uniformly at random.
2. If $g(\mathbf{x}) \neq g(\mathbf{x}^{(P)})$, return $(\mathbf{x}, \mathbf{x}^{(P)})$ as a distinguishing pair for P.
3. Draw $\mathbf{y} \sim \{0,1\}^B$ uniformly at random.
4. If $g(\mathbf{y}) \neq g(\mathbf{y}^{(Q)})$, return $(\mathbf{y}, \mathbf{y}^{(Q)})$ as a distinguishing pair for Q.
5. Otherwise return "fail."

Figure 3.4: Description of the subroutine **WhereIsTheLiteral**.

Below we assume that the algorithm uses **WhereIsTheLiteral** and correctly determines whether i_j is in $\mathbf{P}_j$ or $\mathbf{Q}_j$ for all $j \in [\ell]$. We let $\mathbf{S}_j$ denote the set (block) among $\mathbf{P}_j$ and $\mathbf{Q}_j$ that contains i_j, and let $\mathbf{T}_j$ denote the other one. We also assume below that the algorithm has obtained a distinguishing pair of g_j for each block $\mathbf{S}_j$.

2. Next we include each element of $\overline{B_1 \cup \cdots \cup B_\ell}$ into a random set $\mathbf{T}$ independently with probability $1/2$. Crucially, the way that $\mathbf{P}_j$ and $\mathbf{Q}_j$ were drawn, and the above assumption that $\mathbf{S}_j$ contains i_j, implies that

$$\mathbf{R} := \mathbf{T} \cup \mathbf{T}_1 \cup \cdots \cup \mathbf{T}_\ell$$

is indeed a uniform random subset of $\overline{I}$ (recall that $I = \{i_j : j \in [\ell]\}$) since other than those in I, each variable is included in $\mathbf{R}$ independently with probability $1/2$. If we draw a random string $\mathbf{x} \sim \mathcal{D}$, then Lemma 3.2.3 implies that $f(\mathbf{x}) \neq f(\mathbf{y})$, where $\mathbf{y} = \mathbf{x}^{(\mathbf{R})}$, with probability at least $\epsilon/2$.

3. Finally, assuming that $f(\mathbf{x}) \neq f(\mathbf{y})$ (with $\mathsf{diff}(\mathbf{x}, \mathbf{y}) = \mathbf{R}$), running the blockwise binary search on $\mathbf{x}, \mathbf{y}$ and blocks $\mathbf{T}, \mathbf{T}_1, \ldots, \mathbf{T}_\ell$ will lead to a distinguishing pair for one of these blocks and will only require $O(\log \ell) \leq O(\log k)$ queries. If it is a distinguishing pair for $\mathbf{T}$, then we can add $\mathbf{T}$ to the list of relevant blocks $B_1, \ldots, B_\ell$ and they remain pairwise disjoint. If it is a distinguish pair for $\mathbf{T}_j$ for some $j \in [\ell]$, then we can replace B_j in the list by $\mathbf{S}_j$ and $\mathbf{T}_j$, each with a new distinguishing pair we found (recall that a distinguishing pair has already been found for each $\mathbf{S}_j$ in the first step). In either case the number of pairwise disjoint relevant blocks grows by one.

Coming back to the assumption we made earlier, although g_j is very unlikely to be a literal, it must fall into one of the following three cases: (1) close to a literal; (2) close to a (all-0 or all-1) constant function; or (3) far from 1-juntas. Here in all cases "close" and "far" are measured with respect to the *uniform distribution* over $\{0,1\}^{B_j}$. As we discuss in more detail in the rest of this section, with some more careful probability analysis the above arguments generalize to the case in which every g_j is only close to (rather than exactly equal to) a literal. On the other hand, if one of the blocks B_j is in case (2) or (3), then by using the fact that we have a distinguishing pair for B_j it is easy to split B_j

Subroutine Literal(g)
Input: Black-box oracle access to a Boolean function g over $\{0,1\}^C$, where C has a distinguishing pair of g.
Output: "True" (indicating g is close to a literal) or disjoint nonempty subsets C', C^* of C and a distinguishing pair of g for each.

1. Repeat $\log k + 6$ times:
   - If **UniformJunta**$(g, 1, \gamma := 1/(8k))$ rejects, then return the two disjoint blocks and distinguishing pairs associated with them found by **UniformJunta**.
2. Let (x, y) be the distinguishing pair for C.
3. Repeat $\log k + 3$ times:
   - Draw a random partition $\mathbf{C}', \mathbf{C}^*$ of C and query $g(x^{(\mathbf{C}')}), g(x^{(\mathbf{C}^*)}), g(y^{(\mathbf{C}')}), g(y^{(\mathbf{C}^*)})$.
   - If $g(x^{(\mathbf{C}')}) = g(x^{(\mathbf{C}^*)}) \neq g(x)$, then return $\mathbf{C}', \mathbf{C}^*$ and $(x, x^{(\mathbf{C}')})$ and $(x, x^{(\mathbf{C}^*)})$ as their distinguishing pairs.
   - If $g(y^{(\mathbf{C}')}) = g(y^{(\mathbf{C}^*)}) \neq g(y)$, then return $\mathbf{C}', \mathbf{C}^*$ and $(y, y^{(\mathbf{C}')})$ and $(y, y^{(\mathbf{C}^*)})$ as their distinguishing pairs.
4. Return "True."

Figure 3.5: Description of the subroutine **Literal**.

into two blocks and find a distinguishing pair for each of them (For example, for case (3) this can be done by running Blais's uniform-distribution junta testing algorithm [Bla09].). More precisely, the subroutine **Literal** in Figure 3.5 will be used to handle these three cases (determine which case we are in and handle it correspondingly).

Here the algorithm **UniformJunta**(g, k, ϵ) comes from Theorem 3.2.1, and we only use the special case when $k = 1$.

Following ideas above we can then show that, for each round, our algorithm can make progress with high probability by increasing the number of pairwise disjoint relevant blocks by one. The algorithm basically keeps repeating these steps until the number of such blocks reaches $k + 1$, in which case the algorithm will reject.

3.2.4 Description of the main algorithm and the proof of correctness

Our algorithm **MainDJunta**$(f, \mathcal{D}, k, \epsilon)$ is described in Figure 3.6.

It maintains two collections of blocks $V = \{B_1, \ldots, B_v\}$ and $U = \{C_1, \ldots, C_u\}$ for some nonnegative integers v and u. They are set to be $\emptyset$ at the beginning, and we will prove the following conditions for V and U are always satisfied as the algorithm runs:

(A). $B_1, \ldots, B_v, C_1, \ldots, C_u \subseteq [n]$ are pairwise disjoint (nonempty) blocks of variables;

(B). A distinguishing pair has been found for each of these blocks. For notational convenience we use (x^j, y^j) to denote the distinguishing pair for each B_j and (x^C, y^C) to denote the distinguishing pair for each block $C \in U$. We also use the following notation:

$$w^j := \left(x^j\right)_{\overline{B_j}} = \left(y^j\right)_{\overline{B_j}} \in \{0,1\}^{\overline{B_j}} \quad \text{and} \quad w^C := \left(x^C\right)_{\overline{C}} = \left(y^C\right)_{\overline{C}} \in \{0,1\}^{\overline{C}},$$

and we let $g_j := f|_{w^j}$ and $g_C := f|_{w^C}$ be Boolean functions over $\{0,1\}^{B_j}$ and $\{0,1\}^C$, respectively.

Throughout the algorithm and its analysis, we set a key parameter $\gamma := 1/(8k)$. Blocks in V are intended to be those that have been "verified" to satisfy the condition that g_j is γ-close to a literal (x_{i_j} or $\overline{x_{i_j}}$ for some unknown variable $i_j \in B_j$) under the uniform distribution, while blocks in U have not been verified yet so they may not satisfy this condition. More formally, at any point in the execution of the algorithm we say that the algorithm is in *good condition* if its current collections V and U satisfy conditions **(A)**, **(B)** and the following condition:

(C). Every g_j, $j \in [v]$, is γ-close to a literal under the uniform distribution over $\{0,1\}^{B_j}$.

Obviously our algorithm starts in good condition, and we will show that it remains in good condition with high probability as the algorithm runs.

The algorithm **MainDJunta**$(f, \mathcal{D}, k, \epsilon)$ starts with $V = U = \emptyset$ and proceeds round by round. For each round, we consider two different situations that this round may get into: we define it is a type-1 round if $u = |U| = 0$ (corresponding to step 2.1 of **MainDJunta**), and it is a type-2 round if $u > 0$ (corresponding to step 2.2). For a type-1 round (with $u = 0$), as described in Figure 3.6 we will draw partitions $\mathbf{P}_j$ and $\mathbf{Q}_j$ from each $B_j \in V$ and then run **WhereIsTheLiteral** on g_j (the function defined for B_j in condition (B) above), $\mathbf{P}_j$ and $\mathbf{Q}_j$ to determine if there is a relevant variable in $\mathbf{P}_j$ or $\mathbf{Q}_j$. Then depending on the results we will follow the idea sketched in Section 3.2.3 and run blockwise binary search on blocks we carefully selected, hoping to increase the number of relevant blocks we can find (note that the whole process and argument still work even when $V = \emptyset$.). We will prove the following lemma for this case in Section 3.2.5:

Lemma 3.2.4. *Assume that f is ϵ-far from k-juntas with respect to $\mathcal{D}$ and* **MainDJunta**$(f, \mathcal{D}, k, \epsilon)$ *is in good condition at the beginning of a type-1 round with $u = 0$ and $v \leq k$. Then it* ***always*** *remain in good condition at the end of this round. Moreover, letting V' and U' be the two corresponding collections of blocks at the end of this round, we have either $|V'| = v$ and $|U'| = 1$, or $|V'| = v - 1$ and $|U'| = 2$ with probability at least $\epsilon/4$.*

For the case when the algorithm is in a type-2 round (with $u \geq 1$), we pick an arbitrary block C from U and check whether g_C is close to a literal under the uniform distribution by using the subroutine **Literal**. We will prove the following lemma for this case in Section 3.2.6:

Lemma 3.2.5. *Assume that f is ϵ-far from k-juntas with respect to $\mathcal{D}$ and* **MainDJunta**$(f, \mathcal{D}, k, \epsilon)$ *is in good condition at the beginning of a type-2 round with $u > 0$ and $v + u \leq k$. Then with probability at least $1 - 1/(64k)$, one of the following two events*

Algorithm MainDJunta$(f, \mathcal{D}, k, \epsilon)$ with the same input / output as **SimpleDJunta** in Figure 3.3.

1. Initialization: Set $V = U = \emptyset$, $r_1 = 64k/\epsilon$ and $r_2 = 3(k+1)$.
2. Repeat the following until $r_1 = 0$ or $r_2 = 0$:
   Let $V = \{B_1, \ldots, B_v\}$ and $U = \{C_1, \ldots, C_u\}$. Let $x^j, y^j, w^j, g_j, x^C, y^C, w^C, g_C$ be the corresponding strings and functions for each $B_j \in V$ and $C \in U$ as described above in condition (B).

   2.1 If $u = 0$, then:

   2.1.1 Set r_1 to be $r_1 - 1$.

   2.1.2 For each $j \in [v]$:

   * Draw a random partition $\mathbf{P}_j, \mathbf{Q}_j$ of B_j and run **WhereIsTheLiteral**$(g_j, \mathbf{P}_j, \mathbf{Q}_j)$.
   * If a distinguishing pair of g_j for $\mathbf{P}_j$ is returned, set $\mathbf{S}_j = \mathbf{P}_j$ and $\mathbf{T}_j = \mathbf{Q}_j$;
   * Else if a distinguishing pair of g_j for $\mathbf{Q}_j$ is returned, set $\mathbf{S}_j = \mathbf{Q}_j$ and $\mathbf{T}_j = \mathbf{P}_j$;
   * Otherwise "fail" is returned. Then skip this round and go back to the beginning of step 2.

   2.1.3 Draw $\mathbf{x} \sim \mathcal{D}$ and a subset $\mathbf{T}$ of $\overline{B_1 \cup \cdots \cup B_v}$ uniformly at random. Let $\mathbf{y} = \mathbf{x}^{(\mathbf{R})}$ with $\mathbf{R} = \mathbf{T} \cup \mathbf{T}_1 \cup \cdots \cup \mathbf{T}_v$. Skip this round and go back to the beginning of step 2 if $f(\mathbf{x}) = f(\mathbf{y})$; otherwise run the blockwise binary search on $\mathbf{x}$ and $\mathbf{y}$ with blocks $\mathbf{T}, \mathbf{T}_1, \ldots, \mathbf{T}_v$:

   * If a distinguishing pair of f for $\mathbf{T}$ is returned, then add $\mathbf{T}$ to U.
   * Otherwise a distinguishing pair (x^*, y^*) of f for $\mathbf{T}_{j^*}$ is returned for some $j^* \in [v]$. Then Remove B_{j^*} from V and add both $\mathbf{S}_{j^*}$ and $\mathbf{T}_{j^*}$ to U.

   2.2 Otherwise $u > 0$, then:

   2.2.1 Set r_2 to be $r_2 - 1$.

   2.2.2 Pick a block $C \in U$ arbitrarily; let (x, y) be its distinguishing pair, $w = x_{\overline{C}}$ and $g = f\restriction_w$. Run **Literal**(g):

   * If **Literal**(g) returns "true," remove C from U and add it to V.
   * Otherwise **Literal**(g) returns disjoint subsets C', C^* of C, each with a distinguish pair of f. Then remove C from U and add both C' and C^* to U.

   2.3 If $|V| + |U| \geq k + 1$, then halt and output "reject."
3. Halt and output "accept."

Figure 3.6: Description of the distribution-free testing algorithm **MainDJunta** for k-juntas.

happens at the end of this round (letting V' and U' be the two corresponding collections of blocks at the end of this round):

1. *The algorithm remains in good condition with $|V'| = v + 1$ and $|U'| = u - 1$;*
2. *The algorithm remains in good condition with $|V'| = v$ and $|U'| = u + 1$.*

Assuming Lemma 3.2.4 and Lemma 3.2.5 for now, we are ready to prove the correctness of **MainDJunta**.

Theorem 3.2.6. *(i) The algorithm* **MainDJunta** *makes $\tilde{O}(k^2/\epsilon)$ queries and always accepts f when it is a k-junta. (ii) It rejects with probability at least $2/3$ when f is ϵ-far from k-juntas with respect to $\mathcal{D}$.*

Proof of Theorem 3.2.6 Assuming Lemmas 3.2.4 and Lemma 3.2.5. **MainDJunta** has one-sided errors since it rejects f only when it has found $k + 1$ pairwise disjoint relevant blocks (in either U or V) of f.

The number of queries it makes for each type-1 round (corresponding to step 2.1 of **MainDJunta**) is $O(k) + O(\log k) = O(k)$, and for each type-2 round it's $O(k \log k + \log k) = O(k \log k)$. Since the number of type-1 rounds is at most $r_1 = 64k/\epsilon$ and the number of type-2 rounds is at most $r_2 = O(k)$, we know the query complexity of **MainDJunta** is $O(k^2/\epsilon + k^2 \log k) = \tilde{O}(k^2/\epsilon)$. This finishes the proof for (i).

In the rest of the proof we assume f is ϵ-far from k-juntas with respect to $\mathcal{D}$ and it's enough to show that our algorithm rejects f with probability at least $2/3$.

For this purpose we introduce a simple potential function F to measure the progress:

$$F(V, U) := 3|V| + 2|U|.$$

Clearly each round of the algorithm is either of type-1 (when $|U| = 0$) or of type-2 (when $|U| > 0$). By Lemma 3.2.4, if the algorithm is in good condition at the beginning of a type-1 round, then the algorithm **always** ends this round in good condition and with probability

at least $\epsilon/4$ the potential function F goes up by at least one (in which case we say that the algorithm succeeds in this type-1 round). By Lemma 3.2.5, if the algorithm is in good condition at the beginning of a type-2 round, then with probability at least $1 - 1/(64k)$ the algorithm ends this round in good condition and F goes up by at least one (in which case we say it succeeds in this type-1 round).

Note that F is 0 at the beginning ($V = U = \emptyset$) and that we must have $|U|+|V| \geq k+1$ (and thus, the algorithm rejects) when the potential function F reaches $3(k+1)$ or above. As a result, a necessary condition for the algorithm to accept is that one of the following two events happens:

E_1: At least one of the type-2 rounds fails.

E_2: E_1 does not happen (so the algorithm succeeds in every type-2 round and remains in good condition all the time). In order to keep F below $3(k+1)$ in the end there are at most $3k+2$ many type-2 rounds and exactly $64k/\epsilon$ many type-1 rounds (so that the algorithm can finish), while the algorithm succeeds in at most $3k+2$ many type-1 rounds out of them.

By a union bound, the probability that E_1 happens is at most:

$$3(k+1) \cdot 1/(64k) \leq 6k \cdot 1/(64k) < 1/8.$$

With a coupling argument we can also show the probability that E_2 happens is at most the probability that

$$\sum_{i=1}^{64k/\epsilon} \mathbf{Z}_i \leq 3k+2,$$

where $\mathbf{Z}_i$'s are i.i.d. $\{0,1\}$-valued random variables that take 1 with probability $\epsilon/4$. The expectation of the sum from LHS is $16k$, and it follows from the Chernoff bound this probability is at most: (using $3k+2 \leq 5k$)

$$\exp\left(-\left(\frac{11}{16}\right)^2 \cdot \frac{16k}{2}\right) = \exp\left(-\frac{121k}{32}\right) < \exp(-3) < 1/8.$$

Finally it follows from a union bound that the algorithm rejects with probability at least $2/3$ when f is ϵ-far from k-juntas with respect to $\mathcal{D}$. This finishes the proof. □

3.2.5 Proof of Lemma 3.2.4

We prove Lemma 3.2.4 in this section. Let's start with a lemma for the subroutine **WhereIsTheLiteral** (Figure 3.4):

Lemma 3.2.7. *Assume that $g : \{0,1\}^B \to \{0,1\}$ is γ-close (with respect to the uniform distribution) to a literal x_i or $\overline{x_i}$ for some $i \in B$. If $i \in P$, then* **WhereIsTheLiteral**(g, P, Q) *returns a distinguishing pair of g for P with probability at least $1 - 4\gamma$; If $i \in Q$, then it returns a distinguishing pair of g for Q with probability at least $1 - 4\gamma$.*

Proof. Let K be the set of strings $x \in \{0,1\}^B$ such that $g(x)$ disagrees with the literal which it is γ-close to (so $|K| \leq \gamma \cdot 2^{|B|}$). We work on the case when $i \in Q$; the proof when $i \in P$ is similar.

Following the description of **WhereIsTheLiteral**, it returns a distinguishing pair for Q if

$$g(\mathbf{x}) = g(\mathbf{x}^{(P)}) \quad \text{and} \quad g(\mathbf{y}) \neq g(\mathbf{y}^{(Q)}).$$

Note that this holds if all four strings fall outside of K (in which case g agrees with the literal x_i or $\overline{x_i}$ with $i \in Q$) and thus, the probability that it does not hold is at most the probability that at least one of these four strings falls inside K. The latter by a union bound is at most 4γ since each of these four strings is uniformly over $\{0,1\}^B$ when $\mathbf{x}, \mathbf{y}$ are drawn uniformly at random from $\{0,1\}^B$. This finishes the proof of the lemma. □

We are now ready to prove Lemma 3.2.4.

Proof of Lemma 3.2.4. First, it is easy to verify that if the algorithm **MainDJunta** starts a type-1 round in good condition, then it ends it in good condition. This is because we never add blocks to V in type-1 rounds, and whenever a block is added to U, it is disjoint from

other blocks and we have found a distinguishing pair for it (note that for the last case in step 2.1.3 of **MainDJunta** we have found a distinguish pair for S_{j^*} in step 2.1.2).

By definition of good condition we know each block $B_i \in V$ is verified, i.e., it is γ-close to some literal x_{i_j} or $\overline{x_{i_j}}$. It then follows from Lemma 3.2.7 and a union bound that, for any sequence of partitions P_j and Q_j of B_j picked at the beginning of step 2.1.2 of **MainDJunta**, the probability that for each $j \in [v]$ the set $\mathbf{S}_j$ chosen according to output of **WhereIsTheLiteral** contains the variable i_j is at least (recall that $\gamma = 1/(8k)$)

$$1 - 4\gamma \cdot v \geq 1 - 4\gamma \cdot k = 1/2.$$

Conditioning on such event happens, and with $\mathbf{T}$ uniformly drawn from $\overline{B_1 \cup \cdots \cup B_v}$ and random partitions $\mathbf{P}_j, \mathbf{Q}_j$ drawn for each B_j in step 2.1.2 and 2.1.3 of **MainDJunta**, the random set $\mathbf{R} = \mathbf{T} \bigcup \mathbf{T}_1 \ldots \bigcup \mathbf{T}_v$ is uniform over all subsets of $\overline{I}$, where $I = \{i_j : j \in [v]\}$. Following Lemma 3.2.3 and the fact that f is ϵ-far from k-juntas with respect to $\mathcal{D}$ and $v \leq k$, we know that with $\mathbf{x}$ drawn according to $\mathcal{D}$ in step 2.1.3 the probability that $f(\mathbf{x}) \neq f(\mathbf{x}^{(\mathbf{R})})$ is at least $\epsilon/2$. Overall we know the algorithm has $f(\mathbf{x}) \neq f(\mathbf{y})$ in step 2.1.3 with probability at least $\epsilon/4$. Given this, the lemma is immediate by inspection of rest of our algorithm. □

3.2.6 Proof of Lemma 3.2.5

We prove Lemma 3.2.5 in this section, which also finishes the proof of correctness of our algorithm.

Proof of Lemma 3.2.5. First it follows from the description of the subroutine **Literal**(g) (Figure 3.5) that it either returns "true" or a pair of nonempty disjoint subsets C', C^* of C and a distinguishing pair of g for each of them (see Theorem 3.2.1 for the algorithm **UniformJunta**). Now let $C \in U$ be the block picked in step 2.2.2 of **MainDJunta**.

If g is γ-close to a literal, then it is easy to verify that one of the two events described in Lemma 3.2.5 will happen no matter what **Literal**(g) outputs (the second event happens if two disjoint non-empty subsets of C are returned, each with a distinguishing pair of g; we are also safe when **Literal** outputs "true" and the algorithm moves C from U to V, which makes the first event happens, since in this case g is indeed γ-close to a literal). So we focus on the other two cases in the rest of the proof: g is γ-far from 1-juntas or g is γ-close to a (all-1 or all-0) constant function. In both cases we show below that the second event described in Lemma 3.2.5 happens with high probability: the algorithm remains in good condition at the end of this round with $|V'| = |V|$ and $|U'| = |U| + 1$. Let's call this event E.

When g is γ-far from 1-juntas under the uniform distribution, we know one of the $\log k + 6$ calls to **UniformJunta** in **Literal**(g) rejects with probability at least

$$1 - (1/3)^{\log k+6} > 1 - 1/(64k).$$

It's easy to verify that event E will happen when this happens.

When g is γ-close to a constant function (say the all-1 function), we have g disagrees with this all-1 function on either string x or y in the distinguishing pair for C (say it's $g(x) = 0$). Let K be the set of strings in $\{0,1\}^C$ that disagree with the all-1 function, and then we know $|K| \leq \gamma \cdot 2^{|C|}$. Step 3 of **Literal**$(g)$ has $g(x^{(\mathbf{C}')}) = g(x)$ or $g(x^{(\mathbf{C}^*)}) = g(x)$ only when one of $x^{(\mathbf{C}')}$ or $x^{(\mathbf{C}^*)}$ lies in K. As both strings are distributed uniformly over $\{0,1\}^C$ when $\mathbf{C}'$ and $\mathbf{C}^*$ are random partitions of C, this happens with probability at most 2γ by a union bound. Therefore among the $\log k + 3$ runs of step 3 of **Literal**, a random partition $\mathbf{C}', \mathbf{C}^*$ of C and a distinguishing pair of g for each of them are returned with probability at least

$$1 - (2\gamma)^{\log k+3} = 1 - (1/(4k))^{\log k+3} > 1 - (1/4)^{\log k+3} = 1 - 1/(64k^2).$$

Again, it's easy to verify that event E will happen when this happens.

This finishes the proof of Lemma 3.2.5. □

3.3 An $\Omega(2^{k/3})$ lower bound for non-adaptive distribution-free testing of k-juntas

In this section we prove an $\Omega(2^{k/3})$ lower bound for the non-adaptive distribution-free testing of k-juntas that was stated as Theorem 1.3.3. We start with some notation. Given a sequence $Y = (y^i : i \in [q])$ of q strings in $\{0,1\}^n$ and a Boolean function $\phi : \{0,1\}^n \to \{0,1\}$, we write $\phi(Y)$ to denote the q-bit string α with $\alpha_i = \phi(y^i)$ for each $i \in [q]$. We also write $\mathbf{Y} = (\mathbf{y}^i : i \in [q]) \sim \mathcal{D}^q$ to denote a sequence of q independent draws from the same probability distribution $\mathcal{D}$.

Let $q = 2^{k/3}$, and let k and n be two positive integers that satisfy $k \leq n/200$. We may further assume that k is at least some absolute constant C (to be specified later) since otherwise, the claimed $\Omega(2^{k/3})$ lower bound on query complexity holds trivially. Our goal is then to show that there exists no q-query non-adaptive distribution-free tester of k-juntas even when the distance parameter ϵ is $1/3$.

Our proof will follow similar ideas as Yao's mini-max principle. We will define in Section 3.3.1 a pair of probability distributions $\mathcal{YES}$ and $\mathcal{NO}$ over pairs $(\phi, \mathcal{D})$, where ϕ is a Boolean function and $\mathcal{D}$ is a distribution over $\{0,1\}^n$. For clarity we use $(f, \mathcal{D})$ to denote pairs drawn from $\mathcal{YES}$ and $(g, \mathcal{D})$ to denote pairs drawn from $\mathcal{NO}$. We show that (1) Every $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ satisfies that f is a k-junta (Lemma 3.3.2); (2) With probability $1 - o_k(1)$, $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ satisfies that $\mathbf{g}$ is $1/3$-far from k-juntas with respect to $\boldsymbol{\mathcal{D}}$ (Lemma 3.3.3).

Unlike other lower bound proofs in the previous chapter that follow from Yao's principle, to prove Theorem 1.3.3 we will be dealing with a special class of algorithms which we call two-phase algorithms, due to our definition of non-adaptive distribution-free testers.

More formally, we define two-phase algorithms as follows: an algorithm A is a q-query two-phase algorithm if it consists of two deterministic maps A_1 and A_2 and decides to either accept or reject each input pair $(\phi, \mathcal{D})$ (where again $\phi : \{0,1\}^n \to \{0,1\}$ and $\mathcal{D}$ is a probability distribution over $\{0,1\}^n$) as follows: upon an input pair $(\phi, \mathcal{D})$, the algorithm receives in the first phase a sequence $Y = (y^i : i \in [q])$ of q strings (which should be thought of as random samples drawn from $\mathcal{D}$) and a binary string $\alpha = \phi(Y)$ of length q. In the second phase, the algorithm A uses the first map A_1 to obtain a sequence of q strings $Z = (z^i : i \in [q]) = A_1(Y, \alpha)$, and feeds them to the black-box oracle. Once the query results $\beta = \phi(Z)$ are back, $A_2(Y, \alpha, \beta)$ returns either 0 or 1 (notice that we do not need to include Z as an input of A_2 since it is determined by Y and α) in which cases the algorithm A either rejects or accepts, respectively. It can be easily seen that a non-adaptively distribution-free tester T works in the same way as above, except that the two maps T_1 and T_2 it contains can be randomized.

Given the description above, unlike typical deterministic algorithms, whether a two-phase algorithm A accepts or not depends on not only $(\phi, \mathcal{D})$ but also the sample strings $\mathbf{Y} \sim \mathcal{D}^q$ it draws. Formally we have

$$\Pr\left[A \text{ accepts } (\phi, \mathcal{D})\right] = \Pr_{\mathbf{Y} \sim \mathcal{D}^q}\left[A_2\Big(\mathbf{Y}, \phi(\mathbf{Y}), \phi\big(A_1(\mathbf{Y}, \phi(\mathbf{Y}))\big)\Big) = 1\right].$$

Then we can show that, to prove Theorem 1.3.3, it suffices to prove the following main technical lemma, which informally says that any q-query two-phase algorithm must behave similarly when it gets input $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ versus $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$:

Lemma 3.3.1. *Any q-query two-phase algorithm A satisfies:*

$$\left| \Pr_{(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}}\left[A \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}})\right] - \Pr_{(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}}\left[A \text{ accepts } (\mathbf{g}, \boldsymbol{\mathcal{D}})\right] \right| \leq 1/4. \tag{3.3}$$

Proof of Theorem 1.3.3 Assuming Lemma 3.3.1, Lemma 3.3.2 and Lemma 3.3.3. Assume for a contradiction that there exists a q-query non-adaptive distribution-free tester T that

ϵ-tests k-juntas when $\epsilon = 1/3$. Then it follows from Lemma 3.3.2 and Lemma 3.3.3 that

$$\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}}\big[T \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\big] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}}\big[T \text{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}})\big] \geq 1/3 - o_k(1),$$

since the first probability is at least $2/3$ and the second is at most

$$1/3\big(1 - o_k(1)\big) + o_k(1) \leq 1/3 + o_k(1).$$

Following our definition of two-phase algorithms, T can be simulated by a random mixture of q-query two-phase algorithms, and there must exist a q-query two-phase algorithm A that satisfies

$$\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}}\big[A \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\big] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}}\big[A \text{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}})\big] \geq 1/3 - o_k(1),$$

a contradiction with Lemma 3.3.1 when k is sufficiently large. □

Here is the high level ideas behind the construction of our distributions $\mathcal{YES}$ and $\mathcal{NO}$ and how we prove Lemma 3.3.1. In making a draw either from $\mathcal{YES}$ or from $\mathcal{NO}$, first $m = \Theta(2^k \log n)$ strings are selected uniformly at random from $\{0,1\}^n$ to form a set $\mathbf{S}$, and the distribution $\boldsymbol{\mathcal{D}}$ in both $\mathcal{YES}$ and $\mathcal{NO}$ is set to be the uniform distribution over $\mathbf{S}$. Also in both $\mathcal{YES}$ and $\mathcal{NO}$, a "background" k-junta $\boldsymbol{h}$ is selected uniformly at random by first picking a set $\mathbf{J}$ of k variables at random and then a random truth table for $\boldsymbol{h}$ over the variables in $\mathbf{J}$. We view the variables in $\mathbf{J}$ as partitioning $\{0,1\}^n$ into 2^k disjoint sections depending on how they are set.

In the case of a draw from $\mathcal{YES}$, the Boolean function $\mathbf{f}$ that goes with the above-described $\boldsymbol{\mathcal{D}}$ is simply the background junta $\mathbf{f} = \boldsymbol{h}$. In the case of a draw from $\mathcal{NO}$, the function $\mathbf{g}$ that goes with $\boldsymbol{\mathcal{D}}$ is formed by modifying the background junta $\boldsymbol{h}$ in the following way (roughly speaking; see Section 3.3.1 for precise details): for each $z \in \mathbf{S}$, we toss a fair coin $\mathbf{b}(z)$ and set the value of $\mathbf{g}(z')$ for each string z' that belongs to the

same section as z and lies within Hamming distance $0.4n$ from z (including z itself) to $\mathbf{b}(z)$ (see Figure 3.7). Note that the value of $\mathbf{g}$ at each string in $\mathbf{S}$ is a fair coin toss, which is completely independent of the background junta $\boldsymbol{h}$. Using the choice of m it can be argued that with high probability $\mathbf{g}$ is $1/3$-far from k-juntas with respect to $\boldsymbol{\mathcal{D}}$ when $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$.

The rough idea of why a pair $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ is difficult for a q-query two-phase algorithm A to distinguish from a pair $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ is as follows. Intuitively, in order for A to distinguish the no-case from the yes-case, it must obtain two strings x^1, x^2 that belong to the same section but are labeled differently. Since there are 2^k sections but q is only $2^{k/3}$, by the birthday paradox it is very unlikely that A obtains two such strings among the first q samples $\mathbf{Y} = (\mathbf{y}^1, \ldots, \mathbf{y}^q)$ drawn from the distribution $\boldsymbol{\mathcal{D}}$. In fact, in both the yes-case and no-case, writing $(\boldsymbol{\phi}, \boldsymbol{\mathcal{D}})$ to denote the (function, distribution) pair, the distribution of the q pairs $(\mathbf{y}^1, \boldsymbol{\phi}(\mathbf{y}^1)), \ldots, (\mathbf{y}^q, \boldsymbol{\phi}(\mathbf{y}^q))$ will be statistically very close to $(\mathbf{x}^1, \mathbf{b}_1), \ldots, (\mathbf{x}^q, \mathbf{b}_q)$ where each pair $(\mathbf{x}^j, \mathbf{b}_j)$ is independently drawn uniformly from $\{0,1\}^n \times \{0,1\}$. Roughly speaking, this implies that the sample pairs $(\mathbf{y}^i, \boldsymbol{\phi}(\mathbf{y}^i))$ from the sampling oracle have no useful information about the set $\mathbf{J}$ of variables that the background junta depends on, and we cannot distinguish whether we are in the yes-case or no-case solely based on them.

What about the q strings $\mathbf{z}^1, \ldots, \mathbf{z}^q$ that A feeds to the black-box oracle? It is also unlikely that any two strings from $\mathbf{y}^1, \ldots, \mathbf{y}^q, \mathbf{z}^1, \ldots, \mathbf{z}^q$ belong to the same section but are labeled differently. Fix an $i \in [q]$, and we give some intuition why it is very unlikely that there is any j such that $\mathbf{z}^i$ lies in the same section as $\mathbf{y}^j$ and has $f(\mathbf{z}^i) \neq f(\mathbf{y}^j)$ (via a union bound, the same intuition handles all $i \in [q]$). Intuitively, since the random samples from the sampling oracle provide no useful information about the set $\mathbf{J}$ defining the background junta, the only thing that A can do in selecting $\mathbf{z}^i$ is to choose how far it lies, in terms of Hamming distance, from the strings in $\mathbf{y}^1, \ldots, \mathbf{y}^q$ (which, recall, are uniform random). Fix $j \in [q]$: if $\mathbf{z}^i$ is within Hamming distance $0.4n$ from $\mathbf{y}^j$, then if $\mathbf{z}^i$ lies in the same section as $\mathbf{y}^j$ it will be labeled the same way as $\mathbf{y}^j$ whether we are in the yes- case or the no-

case. On the other hand, if $\mathbf{z}^i$ is farther than $0.4n$ in Hamming distance from $\mathbf{y}^j$, then it is overwhelmingly likely that $\mathbf{z}^i$ will lie in a different section from $\mathbf{y}^j$ (since it is very unlikely that all $0.4n$ of the flipped variables avoid the set $\mathbf{J}$ of size k). We will formally define distributions $\mathcal{YES}$ and $\mathcal{NO}$ in the next section and prove Lemma 3.3.1 in Section 3.3.2.

3.3.1 The $\mathcal{YES}$ and $\mathcal{NO}$ distributions

Given $J \subseteq [n]$, we partition $\{0,1\}^n$ into *sections* (with respect to J) where the z-section, $z \in \{0,1\}^J$, consists of those $x \in \{0,1\}^n$ which have $x_J = z$. We write $\mathcal{JUNTA}_J$ to denote the uniform distribution over all juntas over J. More precisely, a Boolean function $\boldsymbol{h} : \{0,1\}^n \to \{0,1\}$ drawn from $\mathcal{JUNTA}_J$ is generated as follows: for each $z \in \{0,1\}^J$, a bit $\mathbf{b}(z)$ is chosen independently and uniformly at random, and for each $x \in \{0,1\}^n$ the value of $\boldsymbol{h}(x)$ is set to $\mathbf{b}(x_J)$.

Let $m := 36 \cdot 2^k \ln n$. We start with $\mathcal{YES}$. A pair $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ drawn from $\mathcal{YES}$ is generated as follows:

1. First we draw independently a subset $\mathbf{J}$ of $[n]$ of size k uniformly at random and a subset of strings $\mathbf{S} \subset \{0,1\}^n$ of size m uniformly at random.
2. Next we draw $\mathbf{f} \sim \mathcal{JUNTA}_{\mathbf{J}}$ and set $\boldsymbol{\mathcal{D}}$ to be the uniform distribution over $\mathbf{S}$.

For technical reasons that will become clear in Section 3.3.2 we use $\mathcal{YES}^*$ to denote the probability distribution supported over triples $(f, \mathcal{D}, J)$, with $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{YES}^*$ being generated by the same two steps above (so the only difference is that we include $\mathbf{J}$ in elements of $\mathcal{YES}^*$).

The following observation is straight-forward from the definition of $\mathcal{YES}$.

Lemma 3.3.2. *The function f is a k-junta for every pair $(f, \mathcal{D})$ in the support of $\mathcal{YES}$.*

We now describe $\mathcal{NO}$. A pair $(\mathbf{g}, \boldsymbol{\mathcal{D}})$ drawn from $\mathcal{NO}$ is generated as follows:

1. We draw $\mathbf{J}$ and $\mathbf{S}$ in the same way as the first step of $\mathcal{YES}$.
2. Next we draw $\boldsymbol{h} \sim \mathcal{JUNTA}_{\mathbf{J}}$ and a map $\boldsymbol{\gamma} : \mathbf{S} \to \{0,1\}$ uniformly at random by choosing a bit independently and uniformly at random for each string in $\mathbf{S}$. We usually refer to $\boldsymbol{h}$ as the "*background junta*."
3. The distribution $\boldsymbol{\mathcal{D}}$ is set to be the uniform distribution over $\mathbf{S}$, which is the same as $\mathcal{YES}$. The function $\mathbf{g} : \{0,1\}^n \to \{0,1\}$ is defined using $\boldsymbol{h}, \mathbf{S}$ and $\boldsymbol{\gamma}$ as follows:
   (a) For each string $y \in \mathbf{S}$, set $\mathbf{g}(y) = \boldsymbol{\gamma}(y)$;
   (b) For each string $x \notin \mathbf{S}$, if there exists no $y \in \mathbf{S}$ with $y_{\mathbf{J}} = x_{\mathbf{J}}$ and $d(x,y) \leq 0.4n$, set $\mathbf{g}(x) = \boldsymbol{h}(x)$; otherwise we set $\mathbf{g}(x) = 1$ if there exists such a $y \in \mathbf{S}$ with $\boldsymbol{\gamma}(y) = 1$, and set $\mathbf{g}(x) = 0$ if every such $y \in \mathbf{S}$ has $\boldsymbol{\gamma}(y) = 0$. (The choice of the tie-breaking rule here is not important; we just pick one to make sure that $\mathbf{g}$ is well defined in all cases.)

Similarly we let $\mathcal{NO}^*$ denote the distribution supported on triples $(g, \mathcal{D}, J)$ that are generated above.

See Figure 3.7 for an illustration of a function drawn from $\mathcal{NO}$. To gain some intuition, we first note that about half of the strings $z \in S$ have $g(z)$ disagree with the value of the background junta on the section it lies in. With such a string z in hand (from one of the samples received in the first round), an algorithm may attempt to find a string w that lies in the same section as z but satisfies $g(z) \neq g(w)$. If such a string is found, the algorithm knows for sure that $(g, \mathcal{D})$ is from the $\mathcal{NO}$ distribution and can safely reject the input. However, finding such a w is not easy because one must flip more than $0.4n$ bits of z, but without knowing the variables in J it is hard to keep w in the same section as z after flipping this many bits.

Next we prove that with high probability, $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ satisfies that $\mathbf{g}$ is $1/3$-far from k-juntas with respect to $\boldsymbol{\mathcal{D}}$:

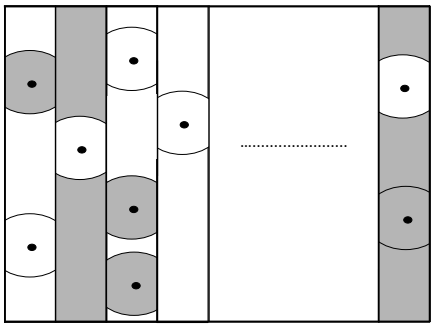

Figure 3.7: A schematic depiction of how $\{0,1\}^n$ is labeled by a function g from $\mathcal{NO}$. The domain $\{0,1\}^n$ is partitioned into 2^k *sections* corresponding to different settings of the variables in J; each section is a vertical strip in the figure. Shaded regions correspond to strings where g evaluates to 1 and unshaded regions to strings where g evaluates to 0. Each string in S is a black dot and the value of g on each such string is chosen uniformly at random. Since in this figure the truncated circles are disjoint, the tie-breaking rule does not come into effect, and for each $z \in S$ all strings in its section within distance at most $0.4n$ (the strings in the truncated circle around z) have the same value as z. The value of g on other strings is determined by the background junta h which assigns a uniform random bit to each section.

Lemma 3.3.3. *With probability at least* $1 - o_k(1)$, $(\mathbf{g}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ *is such that* $\mathbf{g}$ *is* $1/3$*-far from* k*-juntas with respect to the distribution* $\boldsymbol{\mathcal{D}}$ *.*

Proof. Fix a k-junta h, i.e. any set $I \subset [n]$ with $|I| = k$ and any 2^k-bit truth table over variables in I. We have that $dist_{\boldsymbol{\mathcal{D}}}(\mathbf{g}, h)$ is precisely the fraction of strings $z \in \mathbf{S}$ such that $\boldsymbol{\gamma}(z) \neq h(z)$. Since each bit $\boldsymbol{\gamma}(z)$ is drawn independently and uniformly at random, we have that

$$\Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}}\big[\, dist_{\boldsymbol{\mathcal{D}}}(\mathbf{g}, h) \leq 1/3 \,\big] = \Pr_{\mathbf{j}\sim \mathrm{Bin}(m,1/2)}\big[\mathbf{j} \leq m/3\big],$$

which, recalling that $m = 36 \cdot 2^k \ln n$, by a standard Chernoff bound is at most $e^{-m/36} = n^{-2^k}$. The result follows by a union bound over all (at most)

$$\binom{n}{k} \cdot 2^{2^k} \leq n^k \cdot 2^{2^k} = o_k(n^{2^k})$$

possible k-juntas h over n variables. This finishes the proof of the lemma. □

Given Lemma 3.3.2 and Lemma 3.3.3, to prove Theorem 1.3.3 it remains only to prove Lemma 3.3.1.

3.3.2 Proof of Lemma 3.3.1

The following definitions will be useful. Let $Y = (y^i : i \in [q])$ be a sequence of q strings in $\{0,1\}^n$, α be a q-bit string, and $J \subset [n]$ be a set of variables of size k. We say that (Y, α, J) is *consistent* if

$$\alpha_i = \alpha_j \quad \text{for all } i, j \in [q] \text{ with } \quad y^i_J = y^j_J. \tag{3.4}$$

Given a consistent triple (Y, α, J), we write $\mathcal{JUNTA}_{Y,\alpha,J}$ to denote the uniform distribution over all juntas h over J that are consistent with (Y, α). More precisely, a draw of $\boldsymbol{h} \sim \mathcal{JUNTA}_{Y,\alpha,J}$ is generated as follows: for each $z \in \{0,1\}^J$, if there exists a y^i such that $y^i_J = z$, then $\boldsymbol{h}(x)$ is set to α_i for all $x \in \{0,1\}^n$ with $x_J = z$; if no such y^i exists, then a uniform random bit $\mathbf{b}(z)$ is chosen independently and $\boldsymbol{h}(x)$ is set to $\mathbf{b}(z)$ for all x with $x_J = z$.

To prove Lemma 3.3.1, let's fix a q-query two-phase algorithm A and let A_1 and A_2 be the two deterministic algorithms that A consists of. We first derive from A a new *randomized* algorithm A' that works on triples $(\phi, \mathcal{D}, J)$ from the support of either $\mathcal{YES}^*$ or $\mathcal{NO}^*$. Again for clarity we use ϕ to denote a function from the support of $\mathcal{YES}/\mathcal{YES}^*$ or $\mathcal{NO}/\mathcal{NO}^*$, f to denote a function from $\mathcal{YES}/\mathcal{YES}^*$ and g to denote a function from $\mathcal{NO}/\mathcal{NO}^*$.

In addition to being randomized, A' differs from A in two important ways:

1. Like A, A' receives samples $\mathbf{Y} \sim \mathcal{D}^q$ and $\phi(\mathbf{Y})$; but unlike A, A' also receives J for free.
2. Unlike A, A' does not make any black-box queries but simply runs on the triple $(\mathbf{Y}, \phi(\mathbf{Y}), J)$ it receives at the beginning. So formally A' is a randomized algorithm

that runs on triples (Y, α, J), where $Y = (y^i : i \in [q])$ is a sequence of q strings, α is a q-bit string, and $J \subset [n]$ is a set of variables of size k, and outputs "accept" or "reject."

A detailed description of the randomized algorithm A' running on (Y, α, J) is as follows:

1. First, if (Y, α, J) is not consistent, A' immediately halts and rejects (simply because this can never happen if (Y, α, J) is obtained from a triple $(f, \mathcal{D}, J)$ in the support of $\mathcal{YES}^*$). Otherwise A' applies A_1 on (Y, α) to obtain a sequence $Z = (Z^i : i \in [q])$ of q strings.
2. Next, A' draws $\boldsymbol{h}' \sim \mathcal{JUNTA}_{Y,\alpha,J}$. (This is the only part of A' that is randomized.)
3. Finally, A' runs $A_2(Y, \alpha, \boldsymbol{h}'(Z))$ and accepts the input if A_2 outputs 1; otherwise it rejects.

From the description of A' above, whether it accepts a triple $(\phi, \mathcal{D}, J)$ or not depends on both the randomness of $\mathbf{Y}$ and $\boldsymbol{h}'$. Formally we have

$$\Pr\left[A' \text{ accepts } (\phi, \mathcal{D}, J)\right] = \Pr_{\mathbf{Y}, \boldsymbol{h}'}\Big[(\mathbf{Y}, \boldsymbol{\alpha}, J) \text{ is consistent and } A_2\big(\mathbf{Y}, \boldsymbol{\alpha}, \boldsymbol{h}'(\mathbf{Z})\big) = 1\Big].$$

Lemma 3.3.1 follows immediately from the following three lemmas (note that the marginal distribution of $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ in $\mathcal{YES}^*$ (or $(\mathbf{g}, \boldsymbol{\mathcal{D}})$ in $\mathcal{NO}^*$) is the same as $\mathcal{YES}$ (or $\mathcal{NO}$)). In all three lemmas we assume that A is a q-query two-phase algorithm while A' is the randomized algorithm derived from A as described above.

Lemma 3.3.4 (A' behaves similarly on $\mathcal{YES}^*$ and $\mathcal{NO}^*$). *We have*

$$\left| \Pr_{(\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{YES}^*}\left[A' \textit{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J})\right] - \Pr_{(\mathbf{g}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{NO}^*}\left[A' \textit{ accepts } (\mathbf{g}, \boldsymbol{\mathcal{D}}, \mathbf{J})\right] \right| \leq 1/8.$$

Lemma 3.3.5 (A and A' behave identically on $\mathcal{YES}$ and $\mathcal{YES}^*$, respectively). *We have*

$$\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\mathbf{J})\sim\mathcal{YES}^*}\left[A \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\right] = \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\mathbf{J})\sim\mathcal{YES}^*}\left[A' \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}},\mathbf{J})\right]. \tag{3.5}$$

Lemma 3.3.6 (A and A' behave similarly on $\mathcal{NO}$ and $\mathcal{NO}^*$, respectively). *We have*

$$\left|\Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}},\mathbf{J})\sim\mathcal{NO}^*}\left[A \text{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}})\right] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}},\mathbf{J})\sim\mathcal{NO}^*}\left[A' \text{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}},\mathbf{J})\right]\right| \leq 1/8.$$

We start with the proof of Lemma 3.3.4, which says that a limited algorithm such as A' (that doesn't make black-box queries) cannot effectively distinguish between a draw from $\mathcal{YES}^*$ versus $\mathcal{NO}^*$:

Proof of Lemma 3.3.4. Since A' runs on (Y,α,J), it suffices to show that the distributions of $(\mathbf{Y},\boldsymbol{\alpha},\mathbf{J})$ induced from $\mathcal{YES}^*$ and $\mathcal{NO}^*$ have small total variation distance. For this purpose we first note that the distributions of $(\mathbf{Y},\mathbf{J})$ induced from $\mathcal{YES}^*$ and $\mathcal{NO}^*$ are identical: in both cases, $\mathbf{Y}$ and $\mathbf{J}$ are independent; $\mathbf{J}$ is a uniform random subset of $[n]$ of size k; $\mathbf{Y}$ is obtained by first uniformly sampling a subset $\mathbf{S}$ of $\{0,1\}^n$ of size m and then uniformly drawing a sequence of q strings from $\mathbf{S}$ with replacement.

Fix a pair (Y,J) in the support of $(\mathbf{Y},\mathbf{J})$. We say $Y=(y^i : i\in[q])$ is *scattered* by J if $y^i_J \neq y^j_J$ for all $i\neq j\in[q]$. In particular this implies that no string appears more than once in Y. The following claim, whose proof we defer, shows that $\mathbf{Y}$ is scattered by $\mathbf{J}$ with high probability.

Claim 3.3.7. *We have that* $\mathbf{Y}$ *is scattered by* $\mathbf{J}$ *with probability at least* $1-O(2^{-k/3})$.

Fix any (Y,J) in the support of $(\mathbf{Y},\mathbf{J})$ such that Y is scattered by J. We claim that the distributions of $\boldsymbol{\alpha}$ conditioning on $(\mathbf{Y},\mathbf{J})=(Y,J)$ in the $\mathcal{YES}^*$ case and the $\mathcal{NO}^*$ case are identical, from which it follows that the total variation distance between the distributions of $(\mathbf{Y},\boldsymbol{\alpha},\mathbf{J})$ in the two cases is at most $O(2^{-k/3})\leq 1/8$ when k is sufficiently large. Indeed $\boldsymbol{\alpha}$ is uniform over strings of length q in both cases. This is trivial for $\mathcal{NO}^*$. For $\mathcal{YES}^*$ note

that $\boldsymbol{\alpha}$ is determined by the random k-junta $\mathbf{f} \sim \mathcal{JUNTA}_J$; the claim follows from the assumption that Y is scattered by J. □

Proof of Claim 3.3.7. We fix J and show that $\mathbf{Y}$ is scattered by J with high probability. As strings of $\mathbf{Y}$ are drawn one by one, the probability of $\mathbf{y}^i$ colliding with one of the previous samples is at most $(i-1)/m \leq q/m$. By a union bound, all strings in $\mathbf{Y}$ are distinct with probability at least

$$1 - q \cdot (q/m) = 1 - q^2/m = 1 - O(2^{-k/3}).$$

Conditioning on this event, $\mathbf{Y} = (\mathbf{y}^i)$ is distributed precisely as a uniform random sequence from $\{0,1\}^n$ with no repetition and thus each pair $(\mathbf{y}^i, \mathbf{y}^j)$ is distributed uniformly over pairs of distinct strings in $\{0,1\}^n$. As a result, we have

$$\Pr\left[\mathbf{y}^i_J = \mathbf{y}^j_J\right] = \frac{2^{n-k}-1}{2^n-1} \leq \frac{1}{2^k}.$$

By a union bound over $\binom{q}{2}$ pairs we have that the probability of $\mathbf{Y}$ being scattered by J is at least

$$\left(1 - O(2^{-k/3})\right) \cdot \left(1 - \binom{q}{2} \cdot 2^{-k}\right) \geq 1 - O(2^{-k/3}).$$

This finishes the proof of the claim. □

Next we prove Lemma 3.3.5.

Proof of Lemma 3.3.5. The first probability in (3.5) is equal to the probability that

$$A_2\Big(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{f}\big(A_1(\mathbf{Y}, \boldsymbol{\alpha})\big)\Big) = 1,$$

where $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{YES}^*$, $\mathbf{Y} \sim \boldsymbol{\mathcal{D}}^q$ and $\boldsymbol{\alpha} = \mathbf{f}(\mathbf{Y})$. For the second probability, since the

triple $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{YES}^*$ is always consistent, we can rewrite it as the probability that

$$A_2\Big(\mathbf{Y}, \boldsymbol{\alpha}, \boldsymbol{h}'\big(A_1(\mathbf{Y}, \boldsymbol{\alpha})\big)\Big) = 1,$$

where $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{YES}^*$, $\mathbf{Y} \sim \boldsymbol{\mathcal{D}}^q$, $\boldsymbol{\alpha} = \mathbf{f}(\mathbf{Y})$ and $\boldsymbol{h}' \sim \mathcal{JUNTA}_{\mathbf{Y},\boldsymbol{\alpha},\mathbf{J}}$.

To show that these two probabilities are equal, we first note that the distributions of $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J})$ are identical. Fixing any triple (Y, α, J) in the support of $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J})$, which must be consistent, we claim that the distribution of $\mathbf{f}$ conditioning on $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}) = (Y, \alpha, J)$ is exactly $\mathcal{JUNTA}_{Y,\alpha,J}$. This is because, for each $z \in \{0,1\}^J$, if $y^i_J = z$ for some y^i in Y, then we have $\mathbf{f}(x) = \alpha_i$ for all strings x with $x_J = z$; otherwise, we have $\mathbf{f}(x) = \mathbf{b}(z)$ for all x with $x_J = z$, where $\mathbf{b}(z)$ is an independent and uniform bit. This is the same as how $\boldsymbol{h}' \sim \mathcal{JUNTA}_{Y,\alpha,J}$ is generated. It follows directly from this claim that the two probabilities are the same. This finishes the proof of this lemma. □

Finally we prove Lemma 3.3.6, the most difficult among the three lemmas:

Proof of Lemma 3.3.6. Similar to the proof of Lemma 3.3.5, the first probability is the probability that

$$A_2\Big(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{g}\big(A_1(\mathbf{Y}, \boldsymbol{\alpha})\big)\Big) = 1,$$

where $(\mathbf{g}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{NO}^*$ and $\boldsymbol{\alpha} = \mathbf{g}(\mathbf{Y})$, while the second probability is the probability that

$$(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}) \text{ is consistent and } A_2\Big(\mathbf{Y}, \boldsymbol{\alpha}, \boldsymbol{h}'\big(A_1(\mathbf{Y}, \boldsymbol{\alpha})\big)\Big) = 1,$$

where $(\mathbf{g}, \boldsymbol{\mathcal{D}}, \mathbf{J}) \sim \mathcal{NO}^*$, $\mathbf{Y} \sim \boldsymbol{\mathcal{D}}^q$, $\boldsymbol{\alpha} = \mathbf{g}(\mathbf{Y})$, and $\boldsymbol{h}' \sim \mathcal{JUNTA}_{\mathbf{Y},\boldsymbol{\alpha},\mathbf{J}}$. We note that the distributions of $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}, \boldsymbol{\mathcal{D}})$ in the two cases are identical.

The following definition is crucial: we say a tuple $(Y, \alpha, J, \mathcal{D})$ in the support of $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}, \boldsymbol{\mathcal{D}})$ is *good* if it satisfies the following three conditions (S below is the support of $\mathcal{D}$):

E_0: Y is scattered by J.

E_1: Let $Z = A_1(Y, \alpha)$. Then every z in Z and every x in $S \setminus Y$ have $d(x, z) > 0.4n$. (In $S \setminus Y$ we abuse the notation and use Y as a set that contains all strings in the sequence Y.)

E_2: If a string z in Z satisfies $z_J = y_J$ for some y in Y, then we must have $d(y, z) \leq 0.4n$.

We delay the proof of the following claim to the end.

Claim 3.3.8. *We have that* $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}, \boldsymbol{\mathcal{D}})$ *is good with probability at least* $7/8$.

Fix any good $(Y, \alpha, J, \mathcal{D})$ in the support and let $Z = A_1(Y, \alpha)$. We finish the proof by showing that the distribution of $\mathbf{g}(Z)$, a binary string of length q, conditioning on $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}, \boldsymbol{\mathcal{D}}) = (Y, \alpha, J, \mathcal{D})$ is the same as that of $\boldsymbol{h}'(Z)$ with $\boldsymbol{h}' \sim \mathcal{JUNTA}_{Y,\alpha,J}$. This combined with Claim 3.3.8 implies that the difference of the two probabilities has absolute value at most $1/8$.

To see this is the case we partition strings of Z into Z_w, where each Z_w is a nonempty set that contains all z in Z with $z_J = w \in \{0, 1\}^J$. For each Z_w, we consider the following two cases:

1. If there exists no string y in Y with $y_J = w$, then by E_1 strings in Z_w are all far (with Hamming distance more than $0.4n$) from strings of S in this section and thus, $\mathbf{g}(z) = \mathbf{b}(w)$ for some independent and uniform random bit $\mathbf{b}(w)$, for all strings $z \in Z_w$ (z can be close to some string in Y from other section, but that won't affect the value of $\mathbf{g}(z)$ according to our definition).
2. If there exists a y in Y with $y_J = w$ (which must be unique by E_0), say y^i, then by E_1 and E_2 strings in Z_w are all close to y and far from other strings of S in this section. As a result, we have $\mathbf{g}(z) = \alpha_i$ for all strings $z \in Z_w$.

So the conditional distribution of $\mathbf{g}(Z)$ is identical to that of $\boldsymbol{h}'(Z)$ with $\boldsymbol{h}' \sim \mathcal{JUNTA}_{Y,\alpha,J}$. This finishes the proof of the lemma. □

Proof of Claim 3.3.8. We bound the probabilities of $(\mathbf{Y}, \boldsymbol{\alpha}, \mathbf{J}, \boldsymbol{\mathcal{D}})$ violating each of the three conditions E_0, E_1 and E_2 and apply a union bound. By Claim 3.3.7, E_0 is violated with probability $O(2^{-k/3})$.

For E_1, we fix a pair (Y, α) in the support and let $\ell \leq q$ be the number of distinct strings in Y and let $Z = A_1(Y, \alpha)$. Conditioning on $\mathbf{Y} = Y$, $\mathbf{S} \setminus \mathbf{Y}$ is a uniformly random subset of $\{0,1\}^n \setminus Y$ of size $m - \ell$. Instead of working with $\mathbf{S} \setminus \mathbf{Y}$, we let $\mathbf{T}$ denote a set obtained by including distinct elements from $m - \ell$ many independent and uniform random draws from $\{0,1\}^n$ (with replacements).

On the one hand, the total variation distance between $\mathbf{S} \setminus \mathbf{Y}$ and $\mathbf{T}$ is exactly the probability that either (1) $\mathbf{T} \cap Y$ is nonempty or (2) $|\mathbf{T}| < m - \ell$. By two union bounds, (1) happens with probability at most $(m - \ell) \cdot (\ell / 2^n) \leq mq/2^n$ and (2) happens with probability at most $(m/2^n) \cdot m$. As a result, the total variation distance is at most $(mq + m^2)/2^n$. On the other hand, the probability that one of the strings of $\mathbf{T}$ has Hamming distance at most $0.4n$ with one of the strings of Z is at most $mq \cdot \exp(-n/100)$ by a Chernoff bound followed by a union bound. Thus, the probability of violating E_1 is at most (using the assumption that $k \leq n/200$)

$$(mq + m^2)/2^n + mq \cdot \exp(-n/100) = O\big(2^{-n/300} \ln^2 n\big).$$

For E_2, we fix a pair (Y, α) in the support and let $Z = A_2(Y, \alpha)$. Because $\mathbf{J}$ is independent from $(\mathbf{Y}, \boldsymbol{\alpha})$, it remains a subset of $[n]$ of size k drawn uniformly at random. For each pair (y, z) with y from Y and z from Z that satisfy $d(y, z) > 0.4n$, the probability of $y_{\mathbf{J}} = z_{\mathbf{J}}$ over the uniform random draw of $\mathbf{J}$ is at most

$$\frac{\binom{0.6n}{k}}{\binom{n}{k}} \leq (0.6)^k.$$

Since there are at most q^2 many such pairs, it follows from a union bound that the probability of violating E_2 is at most $q^2 \cdot (0.6)^k \leq e^{-0.04k}$.

Finally the lemma follows from a union bound when k (and thus, n) is sufficiently large. $\square$

Chapter 4

Distribution-free Testing of Monotone Conjunctions

In this chapter we will continue to discuss about distribution-free testing, with respect to another class called monotone conjunctions. We will start with some preparation in Section 4.1. Then we will show that any distribution-free tester of monotone conjunctions need to make at least $\tilde{\Omega}(n^{1/3})$ queries (for some constant distance parameter ϵ) in Section 4.2, and also provide an $\tilde{O}(n^{1/3}/\epsilon^5)$-query adaptive distribution-free tester for the same problem in Section 4.3, which pins down the optimal query complexity of this problem at $\tilde{\Theta}(n^{1/3})$ for some constant ϵ, if we ignore the poly-logarithmic factors. In the end we will also extend our proofs (mostly lower bound proofs) to other classes of Boolean functions such as general conjunctions, decision lists and linear threshold functions in Section 4.4.

4.1 Preparation

Let's first introduce some formal definitions and useful notation in this section.

We focus on the distribution-free testing of MCONJ, the class of all monotone conjunctions (or monotone monomials as in [DR11]): $f : \{0,1\}^n \to \{0,1\}$ is in MCONJ if there exists a subset $S \subseteq [n]$ with

$$f(z_1, \ldots, z_n) = \bigwedge_{i \in S} z_i.$$

Note that f is the all-1 function when S is empty. In addition to monotone conjunctions we will also talk about the distribution-free testing of general conjunctions, decision lists, and linear threshhold functions:

- We say $f : \{0,1\}^n \to \{0,1\}$ is a general conjunction if there exist two sets

$S, S' \subseteq [n]$ with

$$f(z_1, \ldots, z_n) = \left(\bigwedge_{i \in S} z_i\right) \bigwedge \left(\bigwedge_{i \in S'} \overline{z_i}\right).$$

.

- A decision list $f : \{0,1\}^n \to \{0,1\}$ of length k over Boolean variables $z_1, \ldots, z_n$ is defined by a sequence of k pairs $(\ell_1, \beta_1), \ldots, (\ell_k, \beta_k)$ and a bit β_{k+1}, where $\beta_i \in \{0,1\}$ for all $i \in [k+1]$ and each ℓ_i is a literal in $\{z_1, \ldots, z_n, \overline{z_1}, \ldots, \overline{z_n}\}$. Given any $z \in \{0,1\}^n$, $f(z)$ is determined in the following way: $f(z) = \beta_i$ if $i \in [k]$ is the smallest index such that ℓ_i is made true by z; if no ℓ_i is true then $f(z) = \beta_{k+1}$.
- We say $f : \{0,1\}^n \to \{0,1\}$ is a linear threshold function if there exist $w_1, w_2, \ldots, w_n, \theta \in \mathbb{R}$ such that $f(z) = 1$ if $w_1 z_1 + \cdots + w_n z_n \geq \theta$ and $f(z) = 0$ if $w_1 z_1 + \cdots + w_n z_n < \theta$.

We use Conj, Dlist, Ltf to denote the class of general conjunctions, decision lists, and linear threshold functions, respectively.

Given a distribution $\mathcal{D}$ over $\{0,1\}^n$ we use $\mathcal{D}(z)$ to denote the probability of a string z in $\{0,1\}^n$ and $\mathcal{D}(C)$ to denote the total probability of strings in $C \subseteq \{0,1\}^n$. We also call $\mathcal{D}(z)$ as the weight of z under the distribution $\mathcal{D}$.

For each string $x \in \{0,1\}^n$, we call x a 0-string (with respect to a Boolean function f) if $f(x) = 0$, and write $f^{-1}(0)$ to denote the set of all 0-strings; we call x a 1-string if $f(x) = 1$, and write $f^{-1}(1)$ to denote the set of 1-strings. We also call $f(x)$ as the label of x by the function f.

For both our lower and upper bound proofs, it is easier to use the language of sets. Given $z \in \{0,1\}^n$:

$$\textsc{Zero}(z) = \big\{i \in [n] : z_i = 0\big\}.$$

For convenience we abuse the notation and allow f to take as input a subset of $[n]$: $f(E)$ is defined as $f(z)$ with $z \in \{0,1\}^n$ and $E = \textsc{Zero}(z)$. This should be clear from the context, since we use lowercase letters for strings and uppercase letters for sets. We call A a 0-set

Procedure MC-Search (f, x)
Input: Black-box oracle access to $f\colon \{0,1\}^n \to \{0,1\}$ and one string $x \in f^{-1}(0)$.
Output: Either an index $i \in \textsc{Zero}(x)$ such that $f(\{i\}) = 0$, or nil.

1. Let $Z = \textsc{Zero}(x)$. If $Z = \emptyset$, return nil; if $|Z| = 1$, output the only index in Z.
2. While $|Z| \geq 2$ do the following:
   - Let Z_0 be the subset of Z that contains the smallest $\lceil |Z|/2 \rceil$ indices in Z, and $Z_1 = Z \setminus Z_0$.
   - Query both $f(Z_0)$ and $f(Z_1)$.
   - If $f(Z_0) = 0$, set $Z = Z_0$; if $f(Z_0) = 1$ but $f(Z_1) = 0$, set $Z = Z_1$; otherwise, return nil.
3. Return the only element that remains in Z.

Figure 4.1: **MC-Search** procedure from Dolev and Ron [DR11].

if $f(A) = 0$, and B a 1-set if $f(B) = 1$.

As shown in the figure above we will use a deterministic procedure called **MC-Search** as an important ingredient of our algorithm, and it is also needed in the argument of our lower bound proof. This procedure was introduced by Dolev and Ron in [DR11] (we name it differently to distinguish it from other binary search procedures in this thesis). Given a string $x \in f^{-1}(0)$, it uses binary search and tries to find a single index $i \in \textsc{Zero}(x)$ such that $f(\{i\}) = 0$. We can easily note that such an index i always exists if f is in MCONJ.

We record the following property of **MC-Search** :

Property 4.1.1. ***MC-Search*** *uses $O(\log n)$ many queries. Given as an input $x \in f^{-1}(0)$, it returns either* nil *or an index $i \in \textsc{Zero}(x)$ such that $f(\{i\}) = 0$. The former never happens if $f \in$ MCONJ.*

Given $x \in f^{-1}(0)$, we write $h(x) \in [n] \cup \{\text{nil}\}$ to denote the output of **MC-Search** running on (f, x) ($h(\cdot)$ is well-defined since the procedure is deterministic). We follow [DR11] and call $x \in f^{-1}(0)$ an *empty* string (with respect to f) if $h(x) = \text{nil}$. We say **MC-Search** fails on (f, x) when $h(x) = \text{nil}$, and call $h(x) \in [n]$ the *representative index* of x (with respect to f) when it doesn't fail.

4.2 Lower bound for distribution-free testing of monotone conjunctions

In this section we present an $\tilde{\Omega}(n^{1/3})$ lower bound for distribution-free testing of monotone conjunctions (and some constant distance parameter ϵ) and prove Theorem 1.4.3. We start with some high level ideas in Section 4.2.1. Then following Yao's mini-max principle, just like before we define two distributions $\mathcal{YES}$ and $\mathcal{NO}$ in Section 4.2.2, and show it's hard to distinguish the two distributions in Sections 4.2.3 to complete the proof.

4.2.1 High level ideas

We define two distributions $\mathcal{YES}$ and $\mathcal{NO}$ over pairs $(f, \mathcal{D})$, where f is a Boolean function mapping from $\{0,1\}^n \to \{0,1\}$ and $\mathcal{D}$ is a distribution over $\{0,1\}^n$. The two distributions have the following properties:

1. Each draw $(f, \mathcal{D}_f)$ from $\mathcal{YES}$ satisfies that $f \in$ MCONJ;
2. Each draw $(g, \mathcal{D}_g)$ from $\mathcal{NO}$ satisfies that g is $1/3$ far from MCONJ with respect to $\mathcal{D}_g$.

Then following Yao's mini-max principle and definition of distribution-free testers, to prove Theorem 1.4.3 it's enough to prove the following lemma:

Lemma 4.2.1. *For $q = n^{1/3}/\log^3 n$ and any deterministic algorithm T that, upon each input pair $(f, \mathcal{D})$, makes at most q queries to the sampling oracle of $\mathcal{D}$ and at most q queries to the black-box oracle of f, we have:*

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})\sim\mathcal{YES}}[T \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}_\mathbf{f})] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})\sim\mathcal{NO}}[T \text{ accepts } (\mathbf{g}, \boldsymbol{\mathcal{D}}_\mathbf{g})] \right| \leq 1/4.$$

The construction of distributions $\mathcal{YES}$ and $\mathcal{NO}$ as well as the proof of Lemma 4.2.1 are based on ideas from the $\tilde{\Omega}(n^{1/5})$ lower bound proof by Glasner and Servedio [GS09].

Let's briefly review their proof first, and then explain how we improve this lower bound.

For both distributions they sample $m = n^{2/5}$ pairwise disjoint sets $\mathbf{C}_i \subset [n]$ of size $\ell = n^{2/5}$ uniformly at random, and let $\mathbf{R}$ be the union of all $\mathbf{C}_i$'s, $i \in [m]$. For each $i \in [m]$ they further randomly partition $\mathbf{C}_i$ into $\mathbf{C}_i = \mathbf{A}_i \bigcup \mathbf{B}_i$ with $|\mathbf{A}_i| = |\mathbf{B}_i|$, and sample a special index $\boldsymbol{\alpha}_i \in \mathbf{A}_i$ uniformly at random. Let $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ be the strings with $\mathbf{A}_i = \textsc{Zero}(\mathbf{a}^i), \mathbf{B}_i = \textsc{Zero}(\mathbf{b}^i), \mathbf{C}_i = \textsc{Zero}(\mathbf{c}^i)$[1], for $i \in [m]$. Then for a draw $(\mathbf{f}, \boldsymbol{\mathcal{D}}_{\mathbf{f}})$ from distribution $\mathcal{YES}$, $\mathbf{f}(x)$ is defined as the conjunction of x_j for all $j \notin \mathbf{R}$, as well as all $x_{\boldsymbol{\alpha}_i}$ for $i \in [m]$, and the distribution $\boldsymbol{\mathcal{D}}_{\mathbf{f}}$ puts $2/(3m)$ and $1/(3m)$ weights on $\mathbf{b}^i$ and $\mathbf{c}^i$ respectively for each $i \in [m]$; for a draw $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}})$ from distribution $\mathcal{NO}$, the distribution $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$ is defined as being uniform over the $3m$ strings $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ for $i \in [m]$, and $\mathbf{g}$ is defined the same as $\mathbf{f}$ drawn from $\mathcal{YES}$, except for the following special case $\mathbf{g}$ is simply defined as the conjunction of x_j for all $j \notin \mathbf{R}$: for each $i \in [m]$ such that $x_{\boldsymbol{\alpha}_i} = 0$, x sets at least $s = \log n$ variables (with indices in) in $\mathbf{A}_i$ to 0 and all variables in $\mathbf{B}_i$ to 1.

Then for each draw $(\mathbf{f}, \boldsymbol{\mathcal{D}}_{\mathbf{f}}) \sim \mathcal{YES}$, $\mathbf{f}$ sets $\mathbf{f}(\mathbf{a}^i) = \mathbf{f}(\mathbf{c}^i) = 0, \mathbf{f}(\mathbf{b}^i) = 1$ for $i \in [m]$, and $\mathbf{f}$ is obviously a monotone conjunction as promised. For each draw $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}}) \sim \mathcal{NO}$, it's easy to verify that $\mathbf{g}(\mathbf{a}^i) = \mathbf{g}(\mathbf{b}^i) = 1$ and $\mathbf{g}(\mathbf{c}^i) = 0$ for $i \in [m]$. Each tuple $(\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i)$ violates the property of monotone conjunctions since $\mathbf{a}^i \wedge \mathbf{b}^i = \mathbf{c}^i$, and at least one of the values of $\mathbf{g}(\mathbf{a}^i), \mathbf{g}(\mathbf{b}^i), \mathbf{g}(\mathbf{c}^i)$ must be changed in order to make $\mathbf{g}$ a monotone conjunction. Then since the distribution $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$ is uniform over the $3m$ strings $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ for $i \in [m]$, we know $\mathbf{g}$ is $1/3$-far from $\textsc{Mconj}$ with respect to $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$.

Following Yao's mini-max principle it's then enough to show that any deterministic and adaptive algorithm that, upon each input pair $(f, \mathcal{D})$, makes at most $q = \tilde{\Omega}(n^{1/5}) << \sqrt{\ell/s}$ queries to the sampling oracle of $\mathcal{D}$ and at most q queries to the black-box oracle of f, cannot distinguish whether $(f, \mathcal{D})$ is drawn from distribution $\mathcal{YES}$ or $\mathcal{NO}$ with high probability.

Let's fix an input pair $(f, \mathcal{D})$ drawn from either distribution $\mathcal{YES}$ or $\mathcal{NO}$. As discussed

[1]Recall the definition of $\textsc{Zero}(\cdot)$ as the set of indices set to 0 in the input string.

before, one can always assume the algorithm makes all queries to the sampling oracle before making queries to the black-box oracle, and each query to the sampling oracle gives us a random string $\mathbf{x}$ drawn from $\mathcal{D}$ as well as its label $f(\mathbf{x})$ by the function. We will simply call responses of these queries from the sampling oracle as samples. Then the proof can be completed with the following three arguments:

- Following birthday paradox and the fact that $q << \sqrt{\ell} = \sqrt{m}$, we can assume the algorithm gets at most one sample from each triple (a^i, b^i, c^i). Then clearly the q samples give no information about whether the input is drawn from distribution $\mathcal{YES}$ or $\mathcal{NO}$: they are just some random strings either with ℓ 0's and a label of 0 by f, or with $\ell/2$ 0's and a label of 1 by f, and there is no overlap between the variables those strings set to 0. We call the samples with ℓ 0's as long samples, and the samples with $\ell/2$ 0's as short samples.
- After receiving the q samples, the algorithm can make q queries to the black-box oracle. Let Z be the set of indices i that there exists a string x in the samples with $x_i = 0$. Then the algorithm can not risk putting 0's on indices outside Z in a future black-box query. This is because for both distributions putting a 0 on any index outside R will immediately let the function evaluate to 0 and gives us no information. Recall when we draw $(f, \mathcal{D})$ from either distribution $\mathcal{YES}$ or $\mathcal{NO}$, R is uniformly drawn from $[n]$, and the algorithm has a probability of roughly $1-|R|/n >> 1-1/q$ to put a 0 outside R if it puts a 0 outside Z[2]. With at most q future queries to make, this risk is too high to take. Now we just assume the algorithm never puts 0's on indices outside Z.
- Finally we can further sort the possibilities that the algorithm can distinguish the distributions $\mathcal{YES}$ and $\mathcal{NO}$ into the following two rare events (note that for each i, at most one of a^i, b^i or c^i is in the samples):

[2]In actual arguments we may consider the distributions conditioning on receiving the random samples, but here for intuition we just keep it simple

– With a long sample $(c^i, 0)$ for some $i \in [m]$ in hand, the algorithm makes a query that sets at least s variables including the one with index α_i[3] in C_i to 0 (and all other variables to 1), and none of them actually lie in B_i. This event enables us to distinguish the distributions $\mathcal{YES}$ and $\mathcal{NO}$ because setting α_i to 0 should always let the function f evaluate to 0 if the input $(f, \mathcal{D})$ is from distribution $\mathcal{YES}$, while for the case when $(f, \mathcal{D})$ is from distribution $\mathcal{NO}$, the new query meets the special condition described in definition of $\mathcal{NO}$ and will let f evaluate to 1. However, since we don't know how C_i is split into A_i and B_i, we can show with a union bound over q choices for i and q choices for future queries that such kind of events happen with probability at most $q^2/2^s << 1$.
– With a short sample $(z, 1)$ in hand (either a^i or b^i for some $i \in [m]$), the algorithm makes a query that sets less than s variables in $\textsc{Zero}(z)$ to 0 (and all other variables to 1), and gets a label 0 for this query. This event enables us to distinguish the distributions $\mathcal{YES}$ and $\mathcal{NO}$ because it only happens when $(f, \mathcal{D})$ is drawn from $\mathcal{NO}$, $z = a^i$ for some $i \in [m]$, and the α_ith variable is one of the less than s variables that are set to 0 in this new query. For such event the algorithm doesn't know where α_i is, and the best it can do to make this event happen is to put at most s 0's in each $\textsc{Zero}(z)$ for at most q short samples z it gets, trying to hit α_i for some i. With q future queries, such kind of events happen with probability at most $2q^2 \cdot s/\ell << 1$.

Note that to make the argument above to work, we need $m >> q^2, l >> q^2$ and $|R| \cdot q \geq n$ where $|R| = m \cdot l$. This essentially gives $q = \tilde{\Omega}(n^{1/5})$. So Glasner and Servedio had the optimal parameters for this argument.

We are able to further improve this lower bound of $\tilde{\Omega}(n^{1/5})$ to $\tilde{\Omega}(n^{1/3})$, mainly by

[3]We can always identify α_i from c^i following **MC-Search** described in the preparation section.

making two changes to the construction above: 1. the sets C_i don't have to be pairwise disjoint. By allowing C_i to slightly overlap with each other, we have $|R| << m \cdot l$ and get room for more and larger sets C_i. 2. we allow the algorithm to set a few variables outside Z to 0 so that we get R as large as $\Omega(n)$, again making more room for C_i's. These two changes enable us to increase q to $\tilde{\Omega}(n^{1/3})$ in the argument above and achieve a better lower bound. As one can expect, these changes make the proof more complicated as well, and we manage to handle the proof by bringing better mini-structure into the construction of distributions.

4.2.2 The distributions $\mathcal{YES}$ and $\mathcal{NO}$

In this section we present construction for the two distributions $\mathcal{YES}$ and $\mathcal{NO}$. We define the following useful parameters:

$$h := \frac{n^{2/3}}{2\log^2 n}, \quad r := n^{1/3}\log^2 n, \quad \ell := n^{2/3} + 2, \quad m := n^{2/3}, \quad \text{and } s := \log^2 n.$$

Then a draw $(\mathbf{f}, \boldsymbol{\mathcal{D}}_{\mathbf{f}})$ from distribution $\mathcal{YES}$ is obtained from the following procedure:

1. Select a set $\mathbf{R} \subset [n]$ of size $hr + 2m = n/2 + 2n^{2/3}$ uniformly at random.
2. Select $2m$ distinct indices $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m)$ from R uniformly at random. We will refer $(\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m)$ as special indices.
3. Partition $\mathbf{R}' = \mathbf{R} \backslash \{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m, \boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_m\}$ into r sets of size h uniformly at random. We call each set as a block.
4. For each $i \in [m]$, select distinct $2\log^2 n$ blocks uniformly at random (choices for different i are independent). Let $\mathbf{C}'_i$ be the union of them, and let $\mathbf{C}_i = \mathbf{C}'_i \bigcup \{\boldsymbol{\alpha}_i, \boldsymbol{\beta}_i\}$. Thus $|\mathbf{C}_i| = \ell$.
5. For each $i \in [m]$, select distinct $\log^2 n$ blocks from $\mathbf{C}'_i$ uniformly at random, and let $\mathbf{A}_i$ be the union of them together with $\boldsymbol{\alpha}_i$. Let $\mathbf{B}_i = \mathbf{C}_i \backslash \mathbf{A}_i$. Then $(\mathbf{A}_i, \mathbf{B}_i)$ is a partition of $\mathbf{C}_i$, and $|\mathbf{A}_i| = |\mathbf{B}_i| = \ell/2$.

6. Define $\mathbf{f}_1, \mathbf{f}_2 : \{0,1\}^n \to \{0,1\}$ as follows:

$$\mathbf{f}_1(x_1, \ldots, x_n) = \bigwedge_{j \notin \mathbf{R}} x_j \quad \text{and } \mathbf{f}_2(x_1, \ldots, x_n) = \bigwedge_{i \in [m]} x_{\boldsymbol{\alpha}_i}.$$

7. Define $\mathbf{f} : \{0,1\}^n \to \{0,1\}$ as $\mathbf{f}(x) = \mathbf{f}_1(x) \wedge \mathbf{f}_2(x)$ for each $x \in \{0,1\}^n$.
8. Finally we define $\boldsymbol{\mathcal{D}}_{\mathbf{f}}$ as: for each $i \in [m]$, let $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ be the strings such that $\mathbf{A}_i = \textsc{Zero}(\mathbf{a}^i), \mathbf{B}_i = \textsc{Zero}(\mathbf{b}^i)$ and $\mathbf{C}_i = \textsc{Zero}(\mathbf{c}^i)$. We put weights of $2/(3m)$ on $\mathbf{b}^i$ and $1/(3m)$ on $\mathbf{c}^i$ under distribution $\boldsymbol{\mathcal{D}}_{\mathbf{f}}$. All other strings in $\{0,1\}^n$ have zero weights in $\boldsymbol{\mathcal{D}}_{\mathbf{f}}$.

Then clearly for any draw $(\mathbf{f}, \boldsymbol{\mathcal{D}}_{\mathbf{f}}) \sim \mathcal{YES}$ we have have $\mathbf{f}(\mathbf{a}^i) = 0, \mathbf{f}(\mathbf{b}^i) = 1, \mathbf{f}(\mathbf{c}^i) = 0$ for each $i \in [m]$, and $\mathbf{f}$ is always a monotone conjunction as promised.

A draw $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}})$ from distribution $\mathcal{NO}$ is obtained from the following procedure:

1. Follow the first six steps in procedure define above for distribution $\mathcal{YES}$ to sample and get $\mathbf{R}, \mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \mathbf{f}_1, \mathbf{f}_2$ for $i \in [m]$.
2. Define a string $x \in \{0,1\}^n$ is i-special for some $i \in [m]$, if the following two statements hold:
   a) at least $3\log^2 n/4$ blocks in $\mathbf{A}_i$ contain (strictly) more than s indices j with $x_j = 0$;
   b) at least $3\log^2 n/4$ blocks in $\mathbf{B}_i$ contain at most s indices j with $x_j = 0$.
3. Define $\mathbf{g}' : \{0,1\}^n \to \{0,1\}$ as $\mathbf{g}'(x) = 1$ if $\mathbf{f}_2(x) = 0$ but x is i-special for every i satisfying $x_{\boldsymbol{\alpha}_i} = 0$; $\mathbf{g}'(x) = \mathbf{f}_2(x)$ otherwise.
4. Define $\mathbf{g} : \{0,1\}^n \to \{0,1\}$ as $\mathbf{g}(x) = \mathbf{f}_1(x) \wedge \mathbf{g}'(x)$ for each $x \in \{0,1\}^n$.
5. Finally we define $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$ as: for each $i \in [m]$, we put weights of $1/(3m)$ on each of $\mathbf{a}^i$, $\mathbf{b}^i$, and $\mathbf{c}^i$ under distribution $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$. All other strings in $\{0,1\}^n$ have zero weights in $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$.

Same as the above discussion for $\tilde{\Omega}(n^{1/5})$ lower bound from [GS09], for each draw $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}}) \sim \mathcal{NO}$ we have $\mathbf{g}(\mathbf{a}^i) = 1$, $\mathbf{g}(\mathbf{b}^i) = 1$, and $\mathbf{g}(\mathbf{c}^i) = 0$ for $i \in [m]$, and $\mathbf{g}$ is always $1/3$-far from MCONJ with respect to $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$.

Now it's enough to prove Lemma 4.2.1 for the rest of this proof.

4.2.3 Proof of Lemma 4.2.1

The proof for Lemma 4.2.1 is mostly based on ideas from [GS09], but with some adaption to our new construction of distributions $\mathcal{YES}$ and $\mathcal{NO}$. Given an algorithm T specified in Lemma 4.2.1, we will derive a new deterministic algorithm T' from T that has *no access* to the black-box oracle. We will then show that such an algorithm T' cannot distinguish the two distributions $\mathcal{YES}$ and $\mathcal{NO}$ (Lemma 4.2.2) but T' agrees with T most of the time (Lemma 4.2.3 and Lemma 4.2.10), from which Lemma 4.2.1 follows.

Let $q = n^{1/3}/\log^3 n$, and for now let's fix an input pair $(f, \mathcal{D})$ drawn from either distribution $\mathcal{YES}$ or $\mathcal{NO}$.

First we define a strong sampling oracle based on the original sampling oracle of $\mathcal{D}$. Upon each query, the sampling oracle of $\mathcal{D}$ returns a string $\mathbf{x}$ drawn from $\mathcal{D}$ as well as its label $f(\mathbf{x})$. According to the definition of distributions $\mathcal{YES}$ and $\mathcal{NO}$, when the input pair $(f, \mathcal{D})$ is drawn from one of the two distributions, the label $f(\mathbf{x})$ for this sample is essentially redundant: $f(\mathbf{x}) = 0$ if $|\textsc{Zero}(\mathbf{x})| = \ell$, $f(\mathbf{x}) = 1$ if $|\textsc{Zero}(\mathbf{x})| = \ell/2$, and we always have $|\textsc{Zero}(\mathbf{x})| \in \{\ell, \ell/2\}$. Also, for the case when $|\textsc{Zero}(\mathbf{x})| = \ell$ and no matter the input is drawn from which distribution, we know $\mathbf{x} = c^i$ for some $i \in [m]$ and we can always run **MC-Search** from the last section on $\mathbf{x}$ to get a special index $\alpha_i \in C_i$. Given these intuitions, we define a strong sampling oracle based on the original sampling oracle of $\mathcal{D}$ as follows: upon each query, the new oracle samples $\mathbf{x}$ from $\mathcal{D}$, and returns $(\textsc{Zero}(\mathbf{x}), \alpha_i)$[4] for the unique special index $\alpha_i \in \textsc{Zero}(\mathbf{x})$, if $|\textsc{Zero}(\mathbf{x})| = \ell$; it returns

[4]We note that for the lower bound proof it's also easier to work with the language of sets, so here we return $\textsc{Zero}(\mathbf{x})$ rather than $\mathbf{x}$ itself.

$(\textsc{Zero}(\mathbf{x}), nil)$ otherwise (so no extra information for this case). Clearly, to prove Lemma 4.2.1 it's enough to prove it for algorithms with access to this strong sampling oracle of $\mathcal{D}$ as well as the black-box oracle of f, since we are getting extra information from this new oracle.

Let T be a deterministic algorithm specified from the statement of Lemma 4.2.1, and without loss of generality we assume T starts by making q queries to the strong sampling oracle of $\mathcal{D}$. Let $\mathbf{Q} = \{(\mathbf{D}_i, \boldsymbol{\gamma}_i) : i \in [q]\}$ be the sequence of responses (again let's call them as samples) T gets from the oracle, where $\mathbf{D}_i$'s are sets $\textsc{Zero}(\mathbf{x})$ corresponding to sample strings $\mathbf{x}$ and $\boldsymbol{\gamma}_i$'s are either empty or special indices returned from the oracle. For each fixed Q, let $\Gamma = \Gamma(Q)$ be the set of special indices T gets from Q, and let $S = S(Q) = \bigcup_{i\in[q]} D_i$ and $I = I(Q) \subset [q]$ be the set of $i \in [q]$ such that $|D_i| = \ell/2$.

Given these definitions, we now define a new deterministic algorithm T' based on T and with no access to the black-box oracle of f. In addition to the sequence of samples $\mathbf{Q}$ for the first q sampling queries, it also receives R used in the procedure constructing f (no matter the input comes from which distribution). Given a fixed Q and R, T' simulates the behavior of T after receiving Q as follows: whenever T makes a query $z \in \{0,1\}^n$, T' doesn't query the black-box oracle but instead using $p(z, R, Q)$ as the response to this query and proceed as T would do upon receiving this response, where $p(z, R, Q)$ is defined as:

$$p(z, R, Q) = \begin{cases} 0, & \text{if } z_i = 0 \text{ for some } i \in [n]\backslash R \text{ or } i \in \Gamma(Q); \\ 1, & \text{otherwise.} \end{cases}$$

We can show the following lemma that the algorithm T' described above (with no access to the black-box oracle) cannot distinguish the two distributions $\mathcal{YES}$ and $\mathcal{NO}$ with high probability:

Lemma 4.2.2. *For any deterministic algorithm T^* that, upon each input pair $(f, \mathcal{D})$, receives*

R and a sequence Q of q samples from the strong sampling oracle of $\mathcal{D}$ but has no access to the black-box oracle of f, we have:

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})\sim\mathcal{YES}}[T^* \textit{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})\sim\mathcal{NO}}[T^* \textit{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})] \right| = o(1).$$

Proof of Lemma 4.2.2. We prove a stronger statement by giving the following extra information to T^* for free, upon each input pair $(f,\mathcal{D})$ drawn from $\mathcal{YES}$ or $\mathcal{NO}$:

$$J = \Big(\big(C_i, \{A_i, B_i\}, \{\alpha_i, \beta_i\}\big) : i \in [m] \Big).$$

Note that $\{A_i, B_i\}$ is given to T^* but they are not labelled (T^* doesn't know which one of the two sets is A_i). The same holds for $\{\alpha_i, \beta_i\}$. Also R is revealed in J as $R = \cup_i C_i$. After receiving J, T^* receives a sequence of q samples Q and now needs to either accept or reject with no other information about $(f,\mathcal{D})$. We show that T^* cannot distinguish $\mathcal{YES}$ and $\mathcal{NO}$.

By definition, the distribution of $\mathbf{J}$ when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ is the same as that when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$, and we use $\mathcal{J}$ to denote such distribution of $\mathbf{J}$. Given a tuple J drawn from $\mathcal{J}$, we use $\mathcal{Q}_J$ to denote the distribution of the sequence of q samples $\mathbf{Q}$ conditioning on $\mathbf{J} = J$ when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$, and use $\mathcal{Q}'_J$ to denote the distribution of $\mathbf{Q}$ conditioning on $\mathbf{J} = J$ when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$. We show that for any fixed J,

$$\left| \Pr_{\mathbf{Q}\sim\mathcal{Q}_J} \left[T^* \text{ accepts } (J,\mathbf{Q})\right] - \Pr_{\mathbf{Q}\sim\mathcal{Q}'_J} \left[T^* \text{ accepts } (J,\mathbf{Q})\right] \right| = o(1).$$

The lemma then follows because procedures for $\mathcal{YES}$ and $\mathcal{NO}$ induce the same distribution $\mathcal{J}$ of J.

It suffices to show that $\mathcal{Q}_J$ and $\mathcal{Q}'_J$ are close to each other for any fixed tuple J. For this purpose, we say a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ has *no collision* with respect to J if no two sets D_i and D_j of Q come from $\{A_k, B_k, C_k\}$ with the same k. On the one hand,

using the birthday paradox and our choices of $q << \sqrt{m}$, $\mathbf{Q} \sim \mathcal{Q}_J$ has a collision with probability $o(1)$. On the other hand, conditioning on $\mathbf{Q}$ has no collision with respect to J, the probability of $\mathbf{Q} = Q$ in $\mathcal{Q}_J$ is exactly the same as that of $\mathbf{Q} = Q$ in $\mathcal{Q}'_J$ (which is a product of probabilities, one for each sample Q_i in Q: the probability of receiving each sample $Q_i = (D_i, \gamma_i)$ is $1/(6m)$ if $|D_i| = \ell$ and $1/(3m)$ if $|D_i| = \ell/2$). This finishes the proof of the lemma. □

Given this lemma, it's then enough to show that T and T' agree with each other most of the time when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ or $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$. We will prove them in Section 4.2.3.1 and Section 4.2.3.2 respectively.

4.2.3.1 T versus T' when $(\mathbf{f}, \mathbf{D}) \sim \mathcal{YES}$

In this section we consider the case when the input pair $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ is drawn from $\mathcal{YES}$ and show that T' agrees with T with high probability in this case. More formally we will prove the following lemma:

Lemma 4.2.3. *For any deterministic algorithm T that, upon each input pair $(f, \mathcal{D})$, makes q queries to the strong sampling oracle of $\mathcal{D}$ and the black-box oracle of f each, let T' be the algorithm defined based on T in Section 4.2.3. Then we have:*

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}}[T \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}})] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}}[T' \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}})] \right| \leq 0.1.$$

Proof of Lemma 4.2.3. We start with some notation.

Given a sequence of q-samples Q that T and T' receive at the beginning, we use T_Q to denote the binary decision tree of T of depth q upon receiving Q. So each internal node of T_Q is labeled a query string $z \in \{0,1\}^n$, and each leaf is labeled either "accept" or "reject". Given Q, T walks down the tree by making queries about $f(z)$ to the black-box oracle. Given R and Q, T' walks down the same decision tree T_Q but does not make any query to the black-box oracle; instead it follows the bit $p(z, R, Q)$ for each query string z in T_Q.

We show that the probability of T' accepting a pair $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}$ is very close to that of T.

Let $\mathcal{YES}_Q$ denote the distribution of $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ drawn from $\mathcal{YES}$ conditioning on receiving $\mathbf{Q} = Q$ at the beginning. We claim that for any Q,

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}_Q} \left[T \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}) \right] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}_Q} \left[T' \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}) \right] \right| \leq 0.1. \tag{4.1}$$

The lemma then follows directly. In the rest of the proof we consider a *fixed* sequence of samples Q.

We use $S = S(Q)$ to denote the union of sets in Q (so $|S| \leq q\ell = O(n/\log^3 n)$), and use $t = |\Gamma(Q)|$ to denote the number of special indices α_i revealed in Q. By the definition of $\mathcal{YES}$ and with $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}_Q$, every special index $\alpha_i \in S$ must appear in Q since $\boldsymbol{\mathcal{D}}$ has no weight on $\mathbf{a}^i$ (so the only possibility of having a special index $\alpha_i \in S$ is because $\mathbf{C}_i$ is in Q, for which case α_i is also given in Q). Thus, there are exactly $m - t$ many special indices $\boldsymbol{\alpha}_i$ in $\mathbf{R} \setminus S$ and we use $\boldsymbol{\Delta}$ to denote the set of these indices. Let $\mathcal{R}_Q$ denote the distribution of the set $\mathbf{R}$ conditioning on $\mathbf{Q} = Q$. Given an R from the support of $\mathcal{R}_Q$, we abuse the notation and use $\mathcal{YES}_{Q,R}$ to denote the distribution of $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \boldsymbol{\Delta})$, conditioning on fixed Q and R.

We make a few simple but very useful observations. First the leaf of T_Q that T' reaches only depends on the set $\mathbf{R} \sim \mathcal{R}_Q$ it receives at the beginning; we use $w'(\mathbf{R})$ to denote the leaf that T' reaches. Second, conditioning on $\mathbf{Q} = Q$ (and $\mathbf{S} = S$), all indices $i \in [n] \setminus S$ are symmetric and are equally likely to be in $\mathbf{R}$. Thus, $\mathbf{R} \setminus S$ is a subset of $[n] \setminus S$ of size $hr + 2m - |S|$ drawn uniformly at random. Finally, conditioning on $\mathbf{Q} = Q$ and $\mathbf{R} \sim \mathcal{R}_Q$, all indices $i \in \mathbf{R} \setminus S$ are symmetric and equally likely to be in $\boldsymbol{\Delta}$ (i.e., chosen as a special index $\boldsymbol{\alpha}_j$). As a result, in $\mathcal{YES}_{Q,R}$, $\boldsymbol{\Delta}$ is a subset of $R \setminus S$ of size $m - k$ drawn uniformly at random.

Now we work on (4.1). Our plan is to show that, when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}_Q$, most likely T

and T' reach the same leaf of T_Q (and then either both accept or reject).

We need one more definition. For each leaf w of T_Q, we define $H_w \subseteq [n] \setminus S$ to be the set of indices $i \in [n] \setminus S$ such that there exists a query string z on the path from the root to w with $z_i = 0$ and w lies in the 1-subtree of the node that making this query. By the definition of H_w and the way T' walks down T_Q using $\mathbf{R} \sim \mathcal{R}_Q$, a necessary condition for T' to reach w is that $H_w \subset \mathbf{R}$. However, all indices $i \in [n] \setminus S$ are symmetric and equally likely to be in $\mathbf{R}$. So intuitively it is unlikely for T' to reach w if H_w is large.

Inspired by discussions above, we say a leaf w of T_Q is *bad* if $|H_w| \geq 0.02 \cdot n^{1/3}$; otherwise w is a *good* leaf (notice that whether w is good or bad only depends on Q, $S = S(Q)$ and T_Q). We show that, when $\mathbf{R}$ is drawn from $\mathcal{R}_Q$, the probability of $w'(\mathbf{R})$ being bad is $o(1)$. To see this, for each bad leaf w of T_Q we have (letting $K = hr + 2m - |S| = (n/2) + 2n^{2/3} - |S|$ be the size of $\mathbf{R} \setminus S$ and plugging in $|S| \leq q\ell = O(n/\log^3 n)$)

$$\begin{aligned}\Pr_{\mathbf{R}\sim\mathcal{R}_Q}\left[w'(\mathbf{R}) = w\right] &\leq \Pr_{\mathbf{R}\sim\mathcal{R}_Q}\left[H_w \subset \mathbf{R}\right] = \frac{\binom{n-|S|-|H_w|}{K-|H_w|}}{\binom{n-|S|}{K}} \\ &= \frac{K - |H_w| + 1}{n - |S| - |H_w| + 1} \times \cdots \times \frac{K}{n - |S|} < 2^{-|H_w|} \leq 2^{-0.02 \cdot n^{1/3}}.\end{aligned}$$

By a union bound on the at most 2^q many bad leaves in T_Q and our choice of $q = O(n^{1/3}/\log^3 n)$ we have the probability of T' reaching a bad leaf is $o(1)$, when $\mathbf{R} \sim \mathcal{R}_Q$. This allows us to focus on good leaves.

Let w be a good leaf in T_Q, and fix R to be a set from $\mathcal{R}_Q$ such that $w'(R) = w$ (and thus, we must have $H_w \subset R \setminus S$). We bound probability of T not reaching w, when $(\mathbf{f}, \boldsymbol{\mathcal{D}}, \boldsymbol{\Delta}) \sim \mathcal{YES}_{Q,R}$. We claim that this happens only when some special index $\boldsymbol{\alpha}_i$ is in H_w (or equivalently, $H_w \cap \boldsymbol{\Delta}$ is not empty).

We now prove this claim. Let z denote the first query string along the path from the root to w such that $f(z) \neq p(z, R, Q)$. By the definition of $\mathcal{YES}$ and $p(z, R, Q)$, $p(z, R, Q) = 0$ implies $f(z) = 0$. As a result, we must have $f(z) = 0$ and $p(z, R, Q) = 1$. By $p(z, R, Q) = 1$, we have $\textsc{Zero}(z) \subseteq R$ and $\textsc{Zero}(z)$ has none of the special indices in

$\Gamma(Q)$. By $f(z) = 0$, $\textsc{Zero}(z)$ must contain a special index $\boldsymbol{\alpha}_i$ outside of S, and thus this $\boldsymbol{\alpha}_i$ is in $\boldsymbol{\Delta}$. Note that $p(z, R, Q) = 1$, we know z is one of the strings considered in the definition of H_w, and this implies $\boldsymbol{\alpha}_i \in H_w \cap \boldsymbol{\Delta}$.

Using this claim, our earlier discussion on the distribution of $\boldsymbol{\Delta}$ in $\mathcal{YES}_{Q,R}$ and $|H_w| < 0.02n^{1/3}$ as w is a good leaf of T_Q, we have (again, letting K be the size of $R \setminus S$)

$$\begin{aligned}
&\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\boldsymbol{\Delta})\sim\mathcal{YES}_{Q,R}}\Big[T \text{ does not reach } w\Big] \\
\leq &\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\boldsymbol{\Delta})\sim\mathcal{YES}_{Q,R}}\Big[|H_w \cap \boldsymbol{\Delta}| \neq \emptyset\Big] \\
= &1 - \frac{\binom{K-|H_w|}{m-t}}{\binom{K}{m-t}} \leq 1 - \left(1 - \frac{m}{K - |H_w| + 1}\right)^{|H_w|} \\
\leq &1 - \left(1 - \frac{3m}{n}\right)^{|H_w|} \leq 1 - \left(1 - \frac{3}{n^{1/3}}\right)^{0.02n^{1/3}} \\
\approx &1 - e^{-0.06} < 0.07.
\end{aligned}$$

Combining this and the fact that T' reaches a bad leaf with $o(1)$ probability, we have

$$\begin{aligned}
&\Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}_Q}\Big[T \text{ and } T' \text{ reach different leaves of } T_Q\Big] \\
= &\sum_{w} \sum_{R:\, w'(R)=w} \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\boldsymbol{\Delta})\sim\mathcal{YES}_{Q,R}}\Big[T \text{ does not reach } w\Big] \cdot \Pr_{\mathbf{R}\sim\mathcal{R}_Q}[\mathbf{R} = R] \\
= &o(1) + \sum_{\text{good } w} \sum_{R:\, w'(R)=w} \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}},\boldsymbol{\Delta})\sim\mathcal{YES}_{Q,R}}\Big[T \text{ does not reach } w\Big] \cdot \Pr_{\mathbf{R}\sim\mathcal{R}_Q}[\mathbf{R} = R] \\
< &0.1.
\end{aligned}$$

This finishes the proof of (4.1) and the lemma then follows. □

4.2.3.2 T versus T' when $(\mathbf{f}, \mathbf{D}) \sim \mathcal{NO}$

In this section we consider the case when the input pair $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ is drawn from $\mathcal{NO}$ and show that T' agrees with T with high probability in this case as well.

We start by introducing a condition on a sequence of q samples Q, and show that the sequence Q that T' and T receive at the beginning satisfies this condition with probability $1 - o(1)$.

Definition 4.2.4. *Given a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ of q samples that T' and T receive upon a fixed pair $(f, \mathcal{D})$ drawn from $\mathcal{NO}$, we use H_i to denote the* unique *set C_k for some $k \in [m]$ that contains D_i. Then we say that Q is* separated *with respect to $(f, \mathcal{D})$ (since by Q itself one cannot tell if it satisfies the following condition) if for each $i \in [q]$ the number of $2\log^2 n$ blocks of H_i that do not appear in any other H_j, $j \neq i$, is at least $(15/8)\log^2 n$.*

A simple implication from this condition is that, when Q is separated with respect to $(f, \mathcal{D})$, Q contains at most one set from (A_i, B_i, C_i) for each $i \in [m]$ (what we called as "no collision" in the proof of Lemma 4.2.2).

Here is an observation that inspires (part of) the definition above. Assume that algorithm T, given Q, suspects that D_i in Q is $\mathbf{A}_k$ for some k and wants to find the special index $\boldsymbol{\alpha}_k$. It may note that indices that appear in D_i only, $D_i \setminus \cup_{j \neq i} D_j$, are symmetric and are equally likely to be $\boldsymbol{\alpha}_k$. Q being separated with respect to $(f, \mathcal{D})$ implies that there are many such indices in D, and intuitively makes it harder for T to find special indices. Of course the definition of Q being separated is stronger, and intuition behind it will become clear later in the proof of Lemma 4.2.10.

We show that when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$, the sequence of random samples $\mathbf{Q}$ that T and T' receive at the beginning is separated with respect to $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ with probability $1 - o(1)$.

Lemma 4.2.5. *When $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$, a sequence $\mathbf{Q}$ of q samples from the sampling oracle is separated with respect to $(\mathbf{f}, \boldsymbol{\mathcal{D}})$ with probability $1 - o(1)$.*

Proof. Recall that $\mathbf{R}'$ is the subset of $\mathbf{R}$ with $\boldsymbol{\alpha}_i$'s and $\boldsymbol{\beta}_i$'s removed. For each $R' \subset [n]$ of size hr and a partition of R' into r pairwise disjoint blocks of size h each, we write $J = J(R')$ to denote the tuple consists of all blocks in R'. We write $\mathcal{NO}_J$ to denote the distribution of $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ conditioning on $J(\mathbf{R}') = J$, and for each $i \in [m]$ write $\mathbf{C}'_i$ to

denote the set obtained from $\mathbf{C}_i$ after removing $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$. With $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_J$, each $\mathbf{C}'_i$ is the union of $2\log^2 n$ blocks drawn uniformly at random from the r blocks in J.

Fix an J. Below we show that if each $\mathbf{C}'_i$ is the union of $2\log^2 n$ random blocks and a sequence $\mathbf{j}_1, \ldots, \mathbf{j}_q$ is drawn from $[m]$ uniformly and independently, then with probability $1 - o(1)$ we have for each $i \in [q]$:

$$\text{the number of blocks of } \mathbf{C}'_{\mathbf{j}_i} \text{ that appear in } \cup_{k \neq i} \mathbf{C}'_{\mathbf{j}_k} \text{ is at most } \log^2 n/16.$$

It follows that $\mathbf{Q}$ has the desired properties when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_J$ with probability $1 - o(1)$, and the lemma follows.

We now prove the claim. First of all by the birthday paradox and our choices of q and m, the probability of two indices from $\mathbf{j}_1, \ldots, \mathbf{j}_q$ being the same is $o(1)$. Suppose that no two indices in $\mathbf{j}_1, \ldots, \mathbf{j}_q$ are the same. The distribution of $\mathbf{C}'_{\mathbf{j}_1}, \ldots, \mathbf{C}'_{\mathbf{j}_q}$ is then the same as $\mathbf{L}_1, \ldots, \mathbf{L}_q$, where each $\mathbf{L}_i$ is the union of $2\log^2 n$ blocks in J drawn uniformly and independently at random. For the latter, we show the following claim:

Claim 4.2.6. *With probability $1 - o(1)$, for each $i \in [q]$, the number of blocks in $\mathbf{L}_i$ that appear in $\cup_{k \neq i} \mathbf{L}_k$ is at most $\log^2 n/16$.*

This is not really surprising: on expectation, the number of blocks of $\mathbf{L}_i$ that also appear in $\cup_{k \neq i} \mathbf{L}_k$ is

$$(q-1) \cdot \frac{2\log^2 n \cdot 2\log^2 n}{r} = o(1).$$

A formal proof of Claim 4.2.6 can be found in Section 5.2. □

We write E to denote the event that a sequence $\mathbf{Q}$ of q samples drawn from $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ is separated with respect to $(\mathbf{f}, \boldsymbol{\mathcal{D}})$, and $\mathcal{Q}_E$ to denote the probability distribution of $\mathbf{Q}$ conditioning on E. By definition not every Q is in the support of $\mathcal{Q}_E$; we record the following property of Q in the support of $\mathcal{Q}_E$.

Property 4.2.7. *Given any $Q = ((D_i, \gamma_i) : i \in [q])$ in the support of $\mathcal{Q}_E$, each D_i has at most $\log n^2/8$ many blocks that appear in $\cup_{j \neq i} D_j$.*

Given a Q in the support of $\mathcal{Q}_E$, we write $\mathcal{R}_{Q,E}$ to denote the distribution of $\mathbf{R}$, conditioning on $\mathbf{Q} = Q$ and event E happens. It is clear that $\mathcal{R}_{Q,E}$ is the same as $\mathcal{R}_Q$ with E dropped since all indices in $[n] \setminus S(Q)$ remain symmetric and equally likely to be in $\mathbf{R}$ even given E.

Property 4.2.8. *For $\mathbf{R} \sim \mathcal{R}_{Q,E}$, $\mathbf{R} \setminus S(Q)$ is a set of size $hr + 2m - |S(Q)|$ drawn uniformly from $[n] \setminus S(Q)$.*

For fixed Q and each $i \in I(Q)$ $(= \{i \in [q] : |D_i| = \ell/2\})$, we use F_i to denote the other set of size $\ell/2$ paired with D_i, (so F_i is A_k if D_i is B_k and vice versa). Given $Q = ((D_i, \gamma_i) : i \in [q])$ in the support of $\mathcal{Q}_E$ and R in the support of $\mathcal{R}_{Q,E}$, we use $\mathcal{F}^i_{R,Q,E}$ to denote the distribution of $\mathbf{F}_i$ conditioning on $\mathbf{R} = R, \mathbf{Q} = Q$ and event E happens. Then we have the following property for F_i's:

Property 4.2.9. *Every F_i in the support of $\mathcal{F}^i_{R,Q,E}$ has at least $(7/8)\log^2 n$ blocks in $R \setminus S(Q)$. Moreover, with $\mathbf{F}_i \sim \mathcal{F}^i_{R,Q,E}$ those blocks are drawn uniformly at random from blocks in $R \setminus S(Q)$. (More exactly, the number $\boldsymbol{k}$ of blocks in $\mathbf{F}_i$ in $R \setminus S(Q)$ is drawn from a certain distribution, where $\boldsymbol{k} \geq (7/8)\log^2 n$ with probability 1, and then $\boldsymbol{k}$ blocks are drawn uniformly at random from blocks in $R \setminus S(Q)$ to form $\mathbf{F}_i$.)*

Now we are ready to show that T' agrees with T most of the time when $(\mathbf{f}, \mathbf{D}) \sim \mathcal{NO}$, formally stated as the following lemma:

Lemma 4.2.10. *For any deterministic algorithm T that, upon each input pair $(f, \mathcal{D})$, makes q queries to the strong sampling oracle of $\mathcal{D}$ and the black-box oracle of f each, let T' be the algorithm defined based on T in Section 4.2.3. Then we have:*

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}}[T \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}})] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}}[T' \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}})] \right| \leq 0.1.$$

Proof of Lemma 4.2.10. We consider a fixed Q that is a sequence of q samples in the support of $\mathcal{Q}_E$, and prove that for any such Q:

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}_{Q,E}} \left[T \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\right] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}_{Q,E}} \left[T' \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\right] \right| \leq 0.09. \tag{4.2}$$

The lemma then follows from (4.2) and Lemma 4.2.5.

For convenience, we let $S = S(Q)$, $\Gamma = \Gamma(Q)$ and $I = I(Q)$ (so $|S| = O(n/\log^3 n)$). Given R in the support of $\mathcal{R}_{Q,E}$, we let $w'(R)$ denote the leaf of T_Q that T' reaches given R. We define H_w for each leaf w of T_Q and *good*/*bad* leaves of T_Q similarly as in the proof of Lemma 4.2.3. Using the same argument (as by Property 4.2.8, $R \setminus S$ is also drawn uniformly at random from $[n] \setminus S$) we have the probability of $w'(\mathbf{R})$ being bad is $o(1)$ when $\mathbf{R} \sim \mathcal{R}_{Q,E}$. This again allows us to focus on good leaves in T_Q.

Now we fix a good leaf w of T_Q and a set R from $\mathcal{R}_{Q,E}$ with $w'(R) = w$. We use P_w to denote the path of query strings from the root to w. We drop R and Q in $p(z, R, Q)$ since they are fixed. In the rest of the proof we bound the probability of T not reaching w, when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$ ($(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}$ conditioning on fixed R, Q and event E happens).

We consider all the possibilities of T not reaching w. It happens when there exists some query string z on the path P_w such that $p(z) \neq \mathbf{f}(z)$. By the definition of $\mathcal{NO}$, at least one of the following four events holds. We bound the probability of each event by $o(1)$ when $(\mathbf{f},\boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$, and then apply a union bound to complete the proof. For the four events below, Events E_0, E_1 and E_2 cover the case when $p(z) = 1$ but $\mathbf{f}(z) = 0$ for some z in P_w. Event E_3 covers the case when $p(z) = 0$ but $\mathbf{f}(z) = 1$ for some z in P_w. We present the four events with respect to a fixed pair $(f, \mathcal{D})$ drawn from $\mathcal{NO}_{R,Q,E}$ as follows (recall that $s = \log^2 n$):

Event E_0: There is a string z in P_w such that $p(z) = 1$ (so w is in the 1-subtree of the node making query z) but $z_{\alpha_k} = 0$ for some $\alpha_k \notin S$.

Event E_1: There is a string z in P_w such that $p(z) = 1$ but

1) $z_{\alpha_k} = 0$ for some $\alpha_k \in S$ and $\alpha_k \notin \Gamma$;

2) z is not k-special because there are more than $\log^2 n/4$ many blocks in A_k, each of which has at most s indices j with $z_j = 0$.

Event E_2: There is a z in P_w such that $p(z) = 1$ but

1) $z_{\alpha_k} = 0$ for some $\alpha_k \in S$ and $\alpha_k \notin \Gamma$;

2) z is not k-special because there are more than $\log^2 n/4$ many blocks in B_k, each of which has (strictly) more than s indices j such that $z_j = 0$.

Event E_3: There is a z in P_w such that $z_{\alpha_k} = 0$ for some $\alpha_k \in \Gamma$ but z is k-special, i.e., there are at least $3\log^2 n/4$ blocks in A_k, each of which has (strictly) more than s indices j in it with $z_j = 0$; at the same time, there are at least $3\log^2 n/4$ blocks in B_k, each of which has at most s indices j in it with $z_j = 0$.

The probability that E_0 happens when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$ is less than 0.07 by the same argument in the proof of Lemma 4.2.3.

Next we bound the probability of E_1. Let $D'_i = D_i \setminus (\cup_{j \neq i} D_j)$ for each $i \in [q]$. Note that if there is a special index $\alpha_k \in S$ but $\alpha_k \notin \Gamma$, then $\alpha_k \in D'_i$ for some $i \in I$. Fixing a query string z in P_w and an $i \in I$, we bound the probability that E_1 happens at z and $\alpha_k \in D'_i$, and then apply a union bound on at most q^2 pairs of z and i.

Consider the scenario that D_i is indeed A_k for some k; otherwise E_1 can never happen. When D_i is A_k and $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$, D'_i consists of a random special index $\boldsymbol{\alpha}_k$ and $u \geq 7\log^2 n/8$ random blocks. (Note that u can be determined from the size of $|D'_i|$.) A key observation is that all indices in D'_i are symmetric in this case, and the choice of $\boldsymbol{\alpha}_k$ as well as the partition of the rest of D'_i into u blocks are both done uniformly at random. Let $Z = \textsc{Zero}(z) \cap D'_i$. By the observation above, part 1) of E_1 happens with probability $|Z|/|D'_i| = O(|Z|/\ell)$. So to make part 1) happen, one would like to set Z to be as large as possible. However, we claim that if $|Z| \geq 10\log^4 n$, then with high probability, every

block in D'_i has at least $2s$ indices in $\textsc{Zero}(z)$, from which we know part 2) is violated because by event E the number of blocks in $D_i \setminus D'_i$ is at most $\log^2 n/8$.

The claim above is not surprising, since each block by our discussion earlier is a subset of size h drawn from D'_i uniformly at random. So when $|Z| \geq 10\log^4 n$, the expected number of indices of a block in Z is

$$|Z| \cdot \frac{h}{|D'_i|} \geq (10\log^4 n) \cdot \frac{n^{2/3}}{2\log^2 n} \cdot \frac{1}{n^{2/3}+2} \geq 4\log^2 n = 4s.$$

For a formal proof of the claim, we assume that blocks in D'_i are labelled: D'_i is partitioned into $\boldsymbol{\alpha}_k$ and u blocks uniformly at random and then the blocks are labelled uniformly at random from 1 to u. Let's focus on the block labelled j: it is a set of size h drawn from D'_i uniformly at random and thus, can be also generated as a sequence of indices drawn from D'_i uniformly at random and independently until h distinct indices are sampled. However, even if we draw a sequence of h indices from D'_i uniformly at random and independently (an early-stop version of the previous procedure) the probability of having at least $2s$ samples in Z is already $1 - n^{-\Omega(\log n)}$, e.g., by a folklore extension of Chernoff bound (see Lemma 5.2.1). Thus, the probability of block j having at most $2s$ indices in $\textsc{Zero}(z)$ is bounded by $n^{-\Omega(\log n)}$. By a union bound on all blocks in D'_i, we have that every block in D'_i has at least $2s$ indices in $\textsc{Zero}(z)$ with probability $1 - n^{-\Omega(\log n)}$.

Combining the two cases when Z is small and large, we have that E_1 happens at a fixed z and D_i with probability $O(\log^4 n/\ell)$. Applying a union bound, E_1 happens with probability $O(q^2 \log^4 n/\ell) = o(1)$.

Next we consider E_2. Let $Q = ((D_i, \gamma_i) : i \in [q])$, and F_i be the set paired with D_i for each $i \in I$. For convenience we say a block is dense if it has more than s indices in H_w. We claim that a necessary condition for part 2) of E_2 to happen is that there exists an $i \in I$ such that more than $\log^2 n/8$ dense blocks of F_i are outside of S. To see this is the case, let's consider a $z \in P_w$ and k such that E_2 happens at z and α_k. Then it must be

the case that A_k is in Q and B_k is one of the F_i's. By part 2) of E_2, more than $\log^2 n/4$ blocks of B_k has more than s indices in $\textsc{Zero}(z)$. Also, given E, we know that at least $\log^2 n/8$ many such blocks are outside of S. By $p(z) = 1$ we know z is one of the strings used to define H_w. Thus, all indices of $\textsc{Zero}(z)$ outside of S belong to H_w, and these at least $\log^2 n/8$ blocks are dense blocks. The claim then follows.

We fix an $i \in I$ (and apply a union bound later). Also note that H_w is a fixed set in $R \setminus S$ of size at most $0.02n^{1/3}$ because w is a good leaf of T_Q. Consider any partition of $R \setminus S$ into *disjoint* blocks (and certain number of α_i's and β_i's). Then by the size of H_w, at most $O(n^{1/3}/s)$ many of these blocks are dense (have an intersection of size more than s with H_w), and a block drawn uniformly at random from $R \setminus S$ is a dense block with probability only $O(1/\log^4 n)$. By Property 4.2.9 and the fact that $|\mathbf{F}_i| = \ell/2$, $\mathbf{F}_i \sim \mathcal{F}^i_{R,Q,E}$ draws at most $\log^2 n$ blocks uniformly at random from those in $R \setminus S$. The probability that more than $\log^2 n/8$ of them are dense blocks can be bounded by $n^{-\Omega(\log^4 n)}$ (e.g., by following a similar argument used in Section 5.2 and considering a sequence of $2\log^2 n$ blocks sampled uniformly and independently). By applying a union bound on all $i \in I$ we have that E_2 happens with probability $o(1)$ when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$.

For event E_3, we bound the probability that E_3 happens for some string z in P_w and some special index $\alpha_k \in \Gamma$, and then apply a union bound on at most q^2 many pairs of z in P_w and $\alpha_k \in \Gamma$. Consider an adversary that picks a string z and aims to make E_3 happen on z and α_k with probability as high as possible, conditioning on fixed R, Q and event E happens. Since $\alpha_k \in \Gamma$, C_k is a set in Q (paired with α_k as a sample Q_i). To ease the proof, we reveal β_k and all the blocks in C_k to the adversary for free and let J be the set of these blocks. Next, consider the distribution of $\mathbf{A}_k$ and $\mathbf{B}_k$ conditioning on fixed $\alpha_k, \beta_k, J, R, Q$ and event E happens. A key observation is that all blocks in J are equally likely to be in $\mathbf{A}_k$ and $\mathbf{B}_k$: $\mathbf{A}_k$ is the union of α_k and $\log^2 n$ blocks drawn uniformly at random from J, and $\mathbf{B}_k$ is the union of β_k and the rest of blocks from J. This is because, given E and that C_k is in Q, neither $\mathbf{A}_k$ nor $\mathbf{B}_k$ is in Q. Thus, neither of $\alpha_k, \beta_k, J, R, Q, E$ reveals any

information about how blocks in C_k are partitioned into $\mathbf{A}_k$ and $\mathbf{B}_k$ except for the indices α_k and β_k.

Let M denote the set of blocks in J that have strictly more than s indices in $\textsc{Zero}(z)$. For event E_3 to happen, $\mathbf{A}_k$ draws $\log^2 n$ blocks from J uniformly at random and have to hit $3\log^2 n/4$ blocks in M, while $\mathbf{B}_k$ draw random blocks in the same way and only have at most $\log^2 n/4$ blocks in M, which is highly unlikely. For a formal proof, note that M must have at least $3\log^2 n/4$ blocks; otherwise the event never happens. Also, M certainly has at most $2\log^2 n$ blocks (total number of blocks in C_k). We sample $\mathbf{B}_k$ using the following procedure: include in the first phase each block in $\mathbf{B}_k$ independently with probability $1/2$, and then in the second phase either add or remove random blocks to left $\mathbf{B}_k$ with $\log^2 n$ blocks. By Chernoff bound, we have that with probability $1 - n^{-\Omega(\log n)}$ after the first phase $\mathbf{B}_k$ has at least $(11/32)\log^2 n$ blocks in M and at most $(33/32)\log^2 n$ blocks in total (since the expectation for the first number is at least $3\log^2 n/8$ and the expectation for the second number is equal to $\log^2 n$). When this happens, $\mathbf{B}_k$ sampled in the end must have at least $(5/16)\log^2 n > \log^2 n/4$ blocks in M, since we remove at most $(1/32)\log^2 n$ blocks from $\mathbf{B}_k$ in the second phase in this case .

Applying a union bound on all z in P_w and α_k in Γ, we have that E_3 happens with probability $o(1)$.

Combining these bounds on the probability of events E_i, $i \in \{0, 1, 2, 3\}$, we have the probability of T not reaching w when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}_{R,Q,E}$ is less than 0.08. The lemma then follows. □

4.2.3.3 Putting all pieces together

We now combine all the lemmas to prove Lemma 4.2.1 (and then Theorem 1.4.3 follows).

Proof of Lemma 4.2.1. Let T be a deterministic oracle algorithm that makes at most q queries to each oracle, and T' be the algorithm that simulates T with no access to the

black-box oracle described in Section 4.2.3. By Lemmas 4.2.2, 4.2.3, 4.2.10:

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})\sim\mathcal{YES}} \left[T \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})\right] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})\sim\mathcal{NO}} \left[T \text{ accepts } (\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})\right] \right| \leq o(1) + 0.1 + 0.1 < 1/4.$$

This finishes the proof of Lemma 4.2.1. □

4.3 A distribution-free testing algorithm of monotone conjunctions

In this section we present an $\tilde{O}(n^{1/3}/\epsilon^5)$-query adaptive distribution-free tester **MainDM-conj** of monotone conjunctions and prove Theorem 1.4.2. We first give some high level ideas about how our main algorithm **MainDMconj** works in Section 4.3.1. Then we present the algorithm **MainDMconj** in Section 4.3.2 and conduct the analysis of its correctness in Section 4.3.3.

4.3.1 High level ideas

We describe the high-level ideas behind our algorithm **MainDMconj** in this section. For clarity, here we assume that ϵ is a constant. We first review the $\tilde{O}(n^{1/2})$-query algorithm of Dolev and Ron [DR11]. Recall from the preparation section, an ingredient from [DR11], **MC-Search** (Figure 4.1), is a deterministic binary search procedure that, upon $x \in f^{-1}(0)$, attempts to find an index $i \in \textsc{Zero}(x)$ such that $f(\{i\}) = 0$. If it fails on x, then f is not a monotone conjunction for sure; otherwise, we use $h(x)$ to denote the index found and call it the representative index of x. Roughly speaking, the algorithm of Dolev and Ron draws $n^{1/2}$ samples from $\mathcal{D}$ and uses **MC-Search** to compute the representative index $h(x)$ of each sample x from $f^{-1}(0)$. Then the algorithm rejects if $y_\alpha = 0$ for some sample $y \in f^{-1}(1)$ and some representative index α found. The algorithm has one-sided errors,

and Dolev and Ron showed that $n^{1/2}$ samples are enough to reject with high probability when f is far from monotone conjunctions with respect to $\mathcal{D}$.

Our algorithm was inspired by Dolev and Ron's work, as well as obstacles encountered when we were trying to improve the $\tilde{\Omega}(n^{1/3})$ lower bound from last section. To give some intuition, consider a fixed pair $(g, \mathcal{D}) \sim \mathcal{NO}$ from our lower bound proof, with $m \approx n^{2/3}$ many C_i's being of size $\ell \approx n^{2/3}$, $A_i \bigcup B_i$ being a partition of C_i with equal sizes, and α_i being a random special index in A_i. Similar as before, for $i \in [m]$ let a^i, b^i, c^i be the strings with $A_i = \textsc{Zero}(a^i)$, $B_i = \textsc{Zero}(b^i)$ and $C_i = \textsc{Zero}(c^i)$. Then $\mathcal{D}$ is the uniform distribution over all a^i, b^i, c^i's, and g satisfies $g(a^i) = g(b^i) = 1$ and $g(c^i) = 0$. Now consider the following scenario where an adversary tries to modify entries of g outside of $\{a^i, b^i, c^i\}_{i\in[m]}$, aiming to fool algorithms with a small number of queries and pretend g being a monotone conjunction (while it's clearly far from $\textsc{Mconj}$ with respect to $\mathcal{D}$). Let $t \approx n^{1/3}$, then we note the adversary has difficulty in handling the following two testers:

(While we can not just draw random samples directly from $g^{-1}(1)$ or $g^{-1}(0)$ with respect to $\mathcal{D}$, it will become clear later that the more interesting case is when g is not too biased to all-0 or all-1 functions with respect to $\mathcal{D}$, and as a result it's not too hard to draw samples from $g^{-1}(1)$ or $g^{-1}(0)$. Here for intuition let's just say we can directly do so.)

> **Tester 1**. Draw t samples $\mathbf{y}^1, \ldots, \mathbf{y}^t$ from $g^{-1}(1)$ with respect to $\mathcal{D}$. Let $\mathbf{E}_i = \textsc{Zero}(\mathbf{y}^i)$, $\mathbf{E} = \cup_i \mathbf{E}_i$. Given the definition of $\mathcal{D}$ and that $g(a^i) = g(b^i) = 1$ and $g(c^i) = 0$, each $\mathbf{E}_i$ is either A_k or B_k. Repeat t times: pick a subset $\mathbf{Z}$ of $\mathbf{E}$ of size t uniformly at random and query $\mathbf{z}$ with $\textsc{Zero}(\mathbf{z}) = \mathbf{Z}$. (Note that if g is a monotone conjunction, then $\mathbf{E}$ cannot contain any index of a variable that belongs to the conjunction and hence for every $\mathbf{Z} \subseteq \mathbf{E}$ and z with $\textsc{Zero}(z) = Z$, g must return 1 to query z.)
>
> **Tester 2**. Draw $t - 1$ samples $\mathbf{y}^1, \ldots, \mathbf{y}^{t-1}$ from $g^{-1}(1)$ with respect to $\mathcal{D}$, and one sample $\mathbf{x}$ from $g^{-1}(0)$ (so $\textsc{Zero}(\mathbf{x}) = C_k$ for some k). Define $\mathbf{E}_i$ and $\mathbf{E}$ similarly. Use the binary search procedure to find the representative index $h(\mathbf{x})$ of $\mathbf{x}$; for the sake

of discussion here assume that it finds the special index $\alpha_k \in C_k$ (as in our lower bound proof). Pick a subset $\mathbf{Z}$ of $\mathbf{E}$ of size $t-1$ uniformly at random, and query $\mathbf{z}$ with $\textsc{Zero}(\mathbf{z}) = \mathbf{Z} \cup \{\alpha_k\}$. (Note that if g is a monotone conjunction, then $h(x)$ must be the index of a variable in the conjunction and hence, we have $h(x) \notin \mathbf{E}$ and for every $\mathbf{Z} \subseteq \mathbf{E}$ and $\mathbf{z}$ with $\textsc{Zero}(\mathbf{z}) = \mathbf{Z} \cup \{h(x)\}$, g must return 0 to query $\mathbf{z}$.)

Consider an algorithm that runs both testers with independent samples. Clearly g fails and gets rejected (i.e., a proof that $g \notin \textsc{Mconj}$ is found) if it returns 0 to a query z from Tester 1 or it returns 1 to a query z from Tester 2. It turns out that there is no way to design a g that returns the correct bit to pass both testers most of the time. To see this is the case, fix some E and E_i's defined from the samples we get. Assume for now that about half of the E_i's in Tester 1 are indeed some A_k's, so each of them contains a unique special index α_k; in total there are $\Omega(t)$ many of them in E. Given that $|E| \leq n$, $t \approx n^{1/3}$, and we repeat t times in picking $\mathbf{z}$, each with t 0's, most likely one of the strings z queried has a special index $\alpha_k \in \textsc{Zero}(z)$ and it is also the only index in $\textsc{Zero}(z) \cap E_i^*$, where we let E_i^* denote the indices that are unique to E_i among all E_j's. (For the latter, the intuition is that there simply cannot be too many large E_i^* because they are disjoint and their union is E.)

For such a string z drawn and queried in Tester 1, g has to return 1. However, the distribution of such $\mathbf{z}$ is very similar to the distribution of $\mathbf{z}$ queried in Tester 2, where an $\boldsymbol{\alpha}_k$ is first picked randomly (by drawing a random 0-string $\mathbf{x}$ correpsonding to some $C_k = \textsc{Zero}(\mathbf{x})$ and running the binary search procedure on it to reveal α_k) and then unioned with a set of $t-1$ indices drawn uniformly from E' obtained from $t-1$ samples from $g^{-1}(1)$ (think $E' = E \backslash E_i^*$ as the union of $t-1$ samples after deleting E_i from samples discussed above for Tester 1).

This is essentially how our algorithm **MainDMconj** works. It consists of two stages, each of which implements one of the two testers. The main challenge for us is the analysis to show that it works for any input pair $(f, \mathcal{D})$ that not necessarily looks like those

constructed in $\mathcal{NO}$. At a high level, we show that if f is far from monotone conjunctions with respect to $\mathcal{D}$ and passes Stage 1 with high probability, then it fails Stage 2 and gets rejected with high probability since the two distributions of $\mathbf{z}$ queried in the two stages are very close to each other.

An important part of our analysis is the notion of *violation bipartite graph* G_f for each pair $(f, \mathcal{D})$. Compared to the *violation hypergraph* H_f introduced by Dolev and Ron [DR11], our bipartite graph G_f is easier to work with and its vertex covers also characterize the distance between f and the class of monotone conjunctions. In particular, our analysis of correctness heavily relies on a *highly regular* bipartite subgraph G_f^* of G_f, of which every vertex cover still has total weight $\Omega(\epsilon)$. The regularity of G_f^* plays a critical role in our comparison of the two stages. More specifically, it helps bound the double counting when we lower bound the probability of $(f, \mathcal{D})$ failing Stage 2, assuming that it passes Stage 1 with high probability.

4.3.2 Description of the main algorithm

We present our main algorithm **MainDMconj** in this section. For clarity we always write x to denote a string from $f^{-1}(0)$, y to denote a string from $f^{-1}(1)$, and z to denote a string with $f(z)$ unknown (or we do not care about $f(z)$). Just like before we again assume without loss of generality that, upon input $(f, \mathcal{D}, \epsilon)$, each query to the sampling oracle of $\mathcal{D}$ gives a string $x \sim \mathcal{D}$ as well as its label $f(x)$ by f.

We use the following parameters in the algorithm and its future analysis:

$$d := \frac{\log^2(n/\epsilon)}{\epsilon}, \quad d^* := d^2/\epsilon, \quad r := n^{1/3}, \quad t := d \cdot r \quad \text{and} \quad s := t \log n. \tag{4.3}$$

Given these parameters, we present the algorithm **MainDMconj** in Figure 4.2. It consists of three stages (the first stage is for drawing samples and making simple tests for some special corner cases, and the last two stages are the more crucial parts as described

Procedure MainDMconj $(f, \mathcal{D}, \epsilon)$
Input: Black-box oracle access to $f\colon \{0,1\}^n \to \{0,1\}$, sampling access to a probability distribution $\mathcal{D}$ over $\{0,1\}^n$, and a distance parameter $\epsilon \in (0,1)$.
Output: Either "accept" or "reject".

Stage 0. Query $f(1^n)$ and Reject if $f(1^n) = 0$. Make $3t(d^*+1)/\epsilon$ many queries to the sampling oracle. Let $(\mathbf{z}^{i,j}, f(\mathbf{z}^{i,j}))$ denote the pairs received, for $i \in [d^*+1]$ and $j \in [3t/\epsilon]$. Run **MC-Search** to compute the representative index $h(x)$ for each $x \in f^{-1}(0)$ sampled. Reject if one of them has $h(x) = \text{nil}$.
Stage 1. Accept if the number of $j \in [3t/\epsilon]$ with $\mathbf{z}^{1,j} \in f^{-1}(1)$ is less than t; otherwise, we let $\mathbf{y}^1, \ldots, \mathbf{y}^t$ be the first t (not necessarily distinct) 1-strings in $(\mathbf{z}^{1,j})$. Let $\mathbf{B}_i = \textsc{Zero}(\mathbf{y}^i)$, $\mathbf{B} = \cup_i \mathbf{B}_i$.

1.1. Repeat s times: Draw an index $\mathbf{i}$ from $\mathbf{B}$ uniformly at random. Reject if $f(\{\mathbf{i}\}) = 0$.

1.2. Repeat s times: Draw a subset $\mathbf{Z} \subseteq \mathbf{B}$ of size r uniformly at random. Reject if $f(\mathbf{Z}) = 0$.

Stage 2. Repeat the following steps for d^* iterations. For the ith iteration:

- Accept if the number of $j \in [3t/\epsilon]$ with $\mathbf{z}^{i+1,j} \in f^{-1}(1)$ is less than $t-1$ or no string in $(\mathbf{z}^{i+1,j})$ is from $f^{-1}(0)$; otherwise, let $\mathbf{y}^1, \ldots, \mathbf{y}^{t-1}$ be the first $t-1$ (not necessarily distinct) 1-strings from $(\mathbf{z}^{i+1,j})$, and $\mathbf{x}$ be the first 0-string from $(\mathbf{z}^{i+1,j})$. Let $\mathbf{B}_i = \textsc{Zero}(\mathbf{y}^i)$ for each i, and $\mathbf{B} = \cup_i \mathbf{B}_i$.
  Use the binary search procedure to compute $h(\mathbf{x})$, and Reject if $h(\mathbf{x}) = \text{nil}$. Otherwise:
  - 2.1 Let $\boldsymbol{\alpha} = h(\mathbf{x}) \in \textsc{Zero}(\mathbf{x})$. Reject if $\boldsymbol{\alpha} \in \mathbf{B}$.
  - 2.2. Uniformly draw a $\mathbf{P} \subseteq \mathbf{B}$ of size $r-1$. Reject if $f(\mathbf{P} \cup \{\boldsymbol{\alpha}\}) = 1$.

End of Stage 2. Accept.

Figure 4.2: The distribution-free algorithm for testing monotone conjunctions.

from last section). We have the following simple observations:

Observation 4.3.1. ***MainDMconj*** $(f, \mathcal{D}, \epsilon)$ *makes at most* $O((n^{1/3}/\epsilon^5) \cdot \log^7(n/\epsilon))$ *queries to either the black-box oracle of* f *or sampling oracle of* $\mathcal{D}$.

Observation 4.3.2. *All queries to the sampling oracle of* $\mathcal{D}$ *are made in Stage 0.*

Next we prove that **MainDMconj** only has one-sided errors:

Lemma 4.3.3. *If* $f \in$ *MCONJ, then* ***MainDMconj*** *always accepts* $(f, \mathcal{D})$ *for any distribution* $\mathcal{D}$ *over* $\{0,1\}^n$.

Proof. Since **MainDMconj** always accepts at the end of Stage 2, it suffices to show that it never rejects when f is a monotone conjunction. First note that when f is a monotone conjunction, $f(1^n)$ must be 1, and following Property 4.1.1 $h(x) = \text{nil}$ can never happen in Stage 0 with $x \in f^{-1}(0)$.

This leaves us to check steps 1.1, 1.2, 2.1 and 2.2. Assume that $f \in$ MCONJ:

1. If $\mathbf{B}_1, ..., \mathbf{B}_k \subseteq [n]$ satisfy $f(\mathbf{B}_1) = \cdots = f(\mathbf{B}_k) = 1$, then $\mathbf{B}_1, ..., \mathbf{B}_k$ contains no index of variable in the conjunction of f, and every $\mathbf{Z} \subseteq \mathbf{B} = \cup_i \mathbf{B}_i$ satisfies $f(\mathbf{Z}) = 1$. This implies that **MainDMconj** never rejects in step 1.1 or 1.2.
2. For step 2.2, $\boldsymbol{\alpha} = h(\mathbf{x})$ implies that $f(\{\boldsymbol{\alpha}\}) = 0$ and $\boldsymbol{\alpha}$ is an index of variable in the conjunction of f. Therefore $f(\mathbf{P} \cup \{\boldsymbol{\alpha}\})$ always evaluates to 0 in this case and **MainDMconj** never rejects in step 2.2.
3. Combining the two arguments above we know **MainDMconj** doesn't reject in step 2.1 as well: $\boldsymbol{\alpha}$ is an index of variable in the conjunction of f while $f(\mathbf{B})$ always evaluate to 1. This implies that $\boldsymbol{\alpha} \notin \mathbf{B}$.

This finishes the proof of the lemma. □

Theorem 1.4.2 follows directly from the following lemma combined with Observation 4.3.1 and Lemma 4.3.3 (since **MainDMconj** only has one-sided errors its success probability in Lemma 4.3.4 can be easily amplified to $2/3$).

Lemma 4.3.4. *If f is ϵ-far from MCONJ with respect to $\mathcal{D}$, **MainDMconj** $(f, \mathcal{D}, \epsilon)$ rejects with probability at least* 0.1.

4.3.3 Analysis

We prove Lemma 4.3.4 and finish the analysis of our main algorithm in this section. We will start with some simplification of the proof in Section 4.3.3.1. Then we will introduce an important tool of our proof called *violation bipartite graph* in Section 4.3.3.2. Based on

these preparation, we will finish the proof of Lemma 4.3.4 and complete the whole analysis in Section 4.3.3.3.

4.3.3.1 Reduction to well-supported probability distributions

To ease the proof of Lemma 4.3.4, we show that it suffices to focus on so-called well-supported distributions. We say a probability distribution $\mathcal{D}$ on $\{0,1\}^n$ is *well-supported* with respect to f if every empty string ($x \in f^{-1}(0)$ with $h(x) = \mathsf{nil}$) of f has probability zero in $\mathcal{D}$. Given $f \notin$ MCONJ, intuitively an adversary that wants to break Lemma 4.3.4 by pairing f with a hard probability distribution $\mathcal{D}$ may not want to allocate much probability on empty strings of f, since **MainDMconj** rejects when finding any empty string sampled in Stage 0.

Following the intuition that well-supported probability distributions are probably hard cases of Lemma 4.3.4, we prove Lemma 4.3.5 below concerning such distributions in the rest of the section. Before its proof we show that it indeed implies Lemma 4.3.4.

Lemma 4.3.5. *Assume that f is a Boolean function and $\mathcal{D}^*$ is a well-supported distribution with respect to f. If f is $(\epsilon/2)$-far from* MCONJ *with respect to $\mathcal{D}^*$,* ***MainDMconj*** *$(f, \mathcal{D}^*, \epsilon)$ rejects with probability at least 0.1.*

Proof of Lemma 4.3.4 assuming Lemma 4.3.5. Assume that f is ϵ-far from MCONJ with respect to $\mathcal{D}$. Let $\delta \geq 0$ denote the total probability of $\mathcal{D}$ over empty strings of f. If $\delta = 0$, Lemma 4.3.4 follows directly from Lemma 4.3.5 since $\mathcal{D}$ is well-supported. If $\delta \geq \epsilon/2$, **MainDMconj** rejects whenever it samples an empty string in Stage 0, and this happens with probability $1 - o(1)$. We consider below the remaining case when $0 < \delta < \epsilon/2$.

Let $\mathcal{D}'$ denote the following distribution derived from $\mathcal{D}$. The probability of any empty string of f in $\mathcal{D}'$ is 0. The probability of any other string is set to be its probability in $\mathcal{D}$ multiplied by $1/(1-\delta)$. Clearly $\mathcal{D}'$ is now a well-supported probability distribution with respect to f. We prove the following claim:

Claim 4.3.6. *The probability of **MainDMconj** rejecting $(f, \mathcal{D}, \epsilon)$ is at least as large as that of rejecting $(f, \mathcal{D}', \epsilon)$.*

Proof. **MainDMconj** always rejects $(f, \mathcal{D}, \epsilon)$ if one of the samples in Stage 0 is an empty string. Let E denote the event that no sample in Stage 0 is empty. Then the probability of **MainDMconj** accepting $(f, \mathcal{D}', \epsilon)$ is exactly that of it accepting $(f, \mathcal{D}, \epsilon)$ conditioning on E. This follows from the definition of $\mathcal{D}'$ and our observation 4.3.2: Stages 1 and 2 access the black-box oracle only, which does not involve $\mathcal{D}$ or $\mathcal{D}'$. As a result, we have

$$\begin{aligned}\Pr\left[\textbf{MainDMconj } (f, \mathcal{D}, \epsilon) \text{ accepts}\right] &= \Pr\left[\textbf{MainDMconj } (f, \mathcal{D}, \epsilon) \text{ accepts} \,\middle|\, E\right] \cdot \Pr[E] \\ &\leq \Pr\left[\textbf{MainDMconj } (f, \mathcal{D}', \epsilon) \text{ accepts}\right].\end{aligned}$$

This finishes the proof of the claim. □

Finally we show that f is $(\epsilon/2)$-far from MCONJ with respect to $\mathcal{D}'$. Given this we can then apply Claim 4.3.6 to finish the proof of the lemma. To see this is the case, note that the total variation distance $d_{TV}(\mathcal{D}, \mathcal{D}')$ is δ by the definition of $\mathcal{D}'$. This implies that for any Boolean function g, we have

$$\left|dist_{\mathcal{D}}(f, g) - dist_{\mathcal{D}'}(f, g)\right| \leq d_{TV}(\mathcal{D}, \mathcal{D}') \leq \delta.$$

As a result, $dist_{\mathcal{D}'}(f, \text{MCONJ}) \geq dist_{\mathcal{D}}(f, \text{MCONJ}) - \delta \geq \epsilon/2$. This finishes the proof of the lemma. □

Now it's enough to prove Lemma 4.3.5 in the rest of the discussion. For convenience, we still use $\mathcal{D}$ to denote the unknown distribution, but from now on we always assume without loss of generality that 1) $\mathcal{D}$ is well-supported with respect to f, and 2) f is $(\epsilon/2)$-far from MCONJ with respect to $\mathcal{D}$.

It is worth mentioning that since $\mathcal{D}$ is well-supported, **MainDMconj** can skip Stage 0 (which is the reason why it is named Stage 0), and have both Stage 1 and each iteration of

Stage 2 start by making $3t/\epsilon$ new queries to the sampling oracle. We will follow this view in the rest of analysis.

4.3.3.2 The violation bipartite graph

Now let's introduce the notion of *violation bipartite graph*, which plays a crucial part of our analysis. The main lemma of this section shows that if $dist_{\mathcal{D}}(f) \geq \epsilon/2$, then the *violation bipartite graph* G_f of f has a *highly regular* subgraph G_f^* with vertex covers of weight (the probability of sampling certain strings from $\mathcal{D}$) at least $\Omega(\epsilon)$.

We first review the *violation hypergraph* of a Boolean function f introduced by Dolev and Ron [DR11], which inspires us to define the *violation bipartite graph*:

Definition 4.3.7 (Violation Hypergraph). *Given f, we call $H_f = (V(H_f), E(H_f))$ the* violation hypergraph *of f, where $V(H_f) = \{0,1\}^n$; $E(H_f)$ contains all subsets (hyperedges) $\{x, y^1, \ldots, y^t\} \subseteq \{0,1\}^n$ such that*

- *$f(x) = 0$; $f(y^i) = 1$ for all $i \in [t]$; and $\textsc{Zero}(x) \subseteq \cup_{i=1}^t \textsc{Zero}(y^i)$.*

Note that $\{1^n\} \in E(H_f)$ if $f(1^n) = 0$ (this is the only special case when $t = 0$).

It turns out that $dist_{\mathcal{D}}(f, \textsc{Mconj})$ is characterized by weights of vertex covers of H_f.

Lemma 4.3.8 (Lemmas 3.2 and 3.4 of [DR11]). *A function f is in $\textsc{Mconj}$ if and only if $E(H_f) = \emptyset$. For any Boolean function f, every vertex cover C of H_f has total probability $\mathcal{D}(C) \geq dist_{\mathcal{D}}(f, \textsc{Mconj})$.*

Note that this lemma holds for any (not necessarily well-supported) probability distribution $\mathcal{D}$. Now we define the violation bipartite graph of f.

Definition 4.3.9 (Violation Bipartite Graph). *Given a Boolean function f we call the following graph $G_f = (L \cup R, E)$ the* violation bipartite graph *of f: vertices on the left side are $L = f^{-1}(1)$; vertices on the right side are $R = \{j \in [n] : \exists x \in f^{-1}(0)$ and $h(x) = j\}$; add an edge between $y \in f^{-1}(1)$ and $j \in R$ if $y_j = 0$.*

Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^n$. It defines a nonnegative weight $\mathrm{wt}_{\mathcal{D}}(\cdot)$ for each vertex in G_f as follows. The weight of $y \in f^{-1}(1) = L$ is simply $\mathrm{wt}_{\mathcal{D}}(y) = \mathcal{D}(y)$. The weight of $j \in R$ is

$$\mathrm{wt}_{\mathcal{D}}(j) = \sum_{x \in f^{-1}(0):\, h(x)=j} \mathcal{D}(x).$$

Given a set of vertices $C \subseteq L \cup R$, we let $\mathrm{wt}_{\mathcal{D}}(C)$ denote the total weight of C: $\mathrm{wt}_{\mathcal{D}}(C) = \sum_{u \in C} \mathrm{wt}_{\mathcal{D}}(u)$. Most of the time when $\mathcal{D}$ is clear from the context, we drop the subscript and use simply $\mathrm{wt}(\cdot)$ for the weight.

We get the following corollary:

Corollary 4.3.10. *If $\mathcal{D}$ is well-supported, then every vertex cover C of G_f has $wt(C) \geq dist_{\mathcal{D}}(f, \textsc{Mconj})$.*

Proof. Given a vertex cover C of G_f, we define a vertex cover C' of H_f as follows. C' consists of 1) all the empty strings of f; 2) $C \cap L = C \cap f^{-1}(1)$; and 3) $x \in f^{-1}(0)$ such that $h(x) \neq \mathsf{nil}$ and $h(x) \in C \cap R$.

By the definition of C' and $\mathrm{wt}(\cdot)$, we have $\mathrm{wt}(C) = \mathrm{wt}(C')$ ($\mathcal{D}$ is well-supported so has zero probability on empty strings). It suffices to show that C' is a vertex cover of H_f, and then apply Lemma 4.3.8.

Fix a hyperedge $\{x, y^1, \ldots, y^t\}$ in H_f. For the special case when $t = 0$, we have $x = 1^n$ and $f(1^n) = 0$. Thus 1^n is empty, and $1^n \in C'$. When $t \geq 1$, either $h(x) = \mathsf{nil}$, for which case we have $x \in C'$, or $h(x) \neq \mathsf{nil}$ and $h(x) \in \textsc{Zero}(x)$. The latter implies $h(x) \in \textsc{Zero}(y^k)$, for some $k \in [t]$, and thus, $(y^k, h(x))$ is an edge in G_f. Since C covers this edge, either $y^k \in C'$ or $x \in C'$. This finishes the proof of the lemma. □

Next, we extract from G_f a highly regular bipartite graph G_f^*, with the guarantee that any vertex cover of G_f^* still has total weight $\Omega(\epsilon)$ (recall that $dist_{\mathcal{D}}(f, \textsc{Mconj}) \geq \epsilon/2$). We start with some notation. Given a subgraph $G = (L(G) \cup R(G), E(G))$ induced by

$L(G) \subseteq L$ and $R(G) \subseteq R$, the *weight* of graph G is

$$\text{wt}(G) = \sum_{y \in L(G)} \text{wt}(y) \cdot \deg_G(y),$$

where $\deg_G(y)$ is the degree of y in G. Equivalently, one can assign each edge (y, j) in G_f an edge weight of $\text{wt}(y)$, and $\text{wt}(G)$ is its total edge weight. For each $j \in R(G)$, we define its *incoming weight* as

$$\text{in-wt}(j) = \sum_{y:\,(y,j) \in E(G)} \text{wt}(y),$$

which can be viewed as the total edge weight from edges incident to j.

Recall the parameter d in (4.3). We say a vertex $y \in L(G)$ is *heavy* in G if $\deg_G(y) \geq d \cdot \text{wt}(G)$; a vertex $j \in R(G)$ is *heavy* in G if $\text{in-wt}(j) \geq d \cdot \text{wt}(G) \cdot \text{wt}(j)$. In either cases, removing a heavy vertex u (and its incident edges) would reduce $\text{wt}(G)$ by at least $d \cdot \text{wt}(G) \cdot \text{wt}(u)$. We say a vertex is *light* if it is not heavy.

We run the following deterministic procedure on G_f to define a subgraph G_f^* of G_f. (This procedure is not new and has seen many applications in the literature, e.g., see [RM99].)

1. Let $G = G_f$ and $S = \emptyset$. Remove all vertices in G with degree zero.
2. Remove all heavy vertices on the left side of G and their incident edges, if any; move those heavy vertices to S. Also remove vertices on the right side that now have degree zero.
3. If G has a vertex cover C of total (vertex) weight $\text{wt}(C) \leq \epsilon/4$, exit.
4. Remove all heavy vertices on the right side of G and their incident edges, if any; move those heavy vertices to S. Also remove vertices on the left side that now have degree zero.
5. If G has a vertex cover C of total (vertex) weight $\text{wt}(C) \leq \epsilon/4$ or there exists no more heavy vertex in G, exit; otherwise go back to Step 2.

Let $G_f^* = (L^* \cup R^*, E^*)$ denote the subgraph of G_f induced by $L^* \subseteq L$ and $R^* \subseteq R$ we obtain at the end.

We show that G_f^* has no heavy vertex, and any vertex cover C of G_f^* still has a large total weight.

Lemma 4.3.11. *Assume that $\mathcal{D}$ is well-supported with respect to f and they satisfy $dist_{\mathcal{D}}(f, \textsc{Mconj}) \geq \epsilon/2$. Then G_f^* has no heavy vertex, and any of its vertex cover C has a total weight of $wt(C) \geq 3\epsilon/8$.*

Proof. The first part, i.e. G_f^* has no heavy vertex, follows from the second part of the lemma, which implies that the procedure exits because G contains no more heavy vertex.

The second part follows from the claim that $\mathrm{wt}(S) = o(\epsilon)$, since for any vertex cover C of G_f^*, $C \cup S$ is a vertex cover of G_f and by Corollary 4.3.10, $\mathrm{wt}(C \cup S) \geq \epsilon/2$. To prove the claim, we let $G_0, \ldots, G_s$ denote the sequence of graphs obtained by following the procedure, with $G_0 = G_f$ and $G_s = G_f^*$, and let S_i denote the set of vertices that are removed from G_i to obtain G_{i+1} and added to S (note that here S_i does not include those vertices removed because their degrees drop to zero). By the definition of heavy vertices, we have

$$\mathrm{wt}(G_i) - \mathrm{wt}(G_{i+1}) \geq d \cdot \mathrm{wt}(G_i) \cdot \mathrm{wt}(S_i).$$

Given this connection, we upperbound $\mathrm{wt}(S) = \sum_{i=0}^{s-1} \mathrm{wt}(S_i)$ by analyzing the following sum:

$$\sum_{i=0}^{s-1} \frac{\mathrm{wt}(G_i) - \mathrm{wt}(G_{i+1})}{\mathrm{wt}(G_i)} \leq 1 + \sum_{i=0}^{s-2} \int_{\mathrm{wt}(G_{i+1})}^{\mathrm{wt}(G_i)} (1/u)\, du = 1 + \int_{\mathrm{wt}(G_{s-1})}^{\mathrm{wt}(G_0)} (1/u)\, du = O\left(\log(n/\epsilon)\right),$$

where the last inequality follows from $\mathrm{wt}(G_0) \leq n$ and $\mathrm{wt}(G_{s-1}) \geq \epsilon/4$ (since any of its vertex cover, e.g., by taking all vertices on the left side, has weight at least $\epsilon/4$). Thus,

$$\mathrm{wt}(S) = \sum_{i=0}^{s-1} \mathrm{wt}(S_i) \leq \frac{1}{d} \cdot \sum_{i=0}^{s-1} \frac{\mathrm{wt}(G_i) - \mathrm{wt}(G_{i+1})}{\mathrm{wt}(G_i)} = o(\epsilon),$$

by the choice of d in (4.3). This finishes the proof of the lemma. □

Note that because any vertex cover of G_f^* has weight $\Omega(\epsilon)$, we have $\text{wt}(L^*) = \Omega(\epsilon)$. Let $W = \text{wt}(G_f^*)$. Then we also have $W = \Omega(\epsilon)$ simply because every vertex in G_f^* has degree at least one. Since all vertices are light, we have in G_f^* that $\deg(y) \leq d \cdot W$ for all $y \in L^*$ and $\text{in-wt}(j) \leq d \cdot W \cdot \text{wt}(j)$ for all $j \in R^*$.

The bipartite graph G_f^* is extremely useful for the analysis of our algorithm later. To gain some intuition, let's first make a short detour and sketch an informal analysis of the tester of Dolev and Ron [DR11] (note that we have a worse dependency on ϵ compared to their analysis though).

First, let $R' \subseteq R^*$ be the set of vertices $j \in R^*$ such that $\text{in-wt}(j) \geq \text{wt}(j) \cdot W/2$. Then

$$W = \sum_{j \in R^*} \text{in-wt}(j) \leq (W/2) \cdot \sum_{j \notin R'} \text{wt}(j) + d \cdot W \cdot \sum_{j \in R'} \text{wt}(j) \leq (W/2) + d \cdot W \cdot \text{wt}(R'),$$

which implies that $\text{wt}(R') = \Omega(1/d)$. Moreover, every $S \subseteq R'$ satisfies the following nice property (below we use $N(S)$ to denote the set of neighbors of vertices in S, within graph G_f^*):

Lemma 4.3.12. *In G_f^*, every $S \subseteq R'$ satisfies $\text{wt}(N(S)) = \Omega\left(\text{wt}(S)/d\right)$.*

Proof. The total edge weight between S and $N(S)$ in G_f* is

$$\sum_{j \in S} \text{in-wt}(j) \leq \sum_{y \in N(S)} \deg(y) \cdot \text{wt}(y).$$

Because $S \subseteq R'$, the LHS is at least

$$\sum_{j \in S} \text{in-wt}(j) \geq (W/2) \cdot \sum_{j \in S} \text{wt}(j) = (W/2) \cdot \text{wt}(S).$$

Since there is no heavy vertex in G_f^*, the RHS is at most

$$\sum_{y \in N(S)} \deg(y) \cdot \mathrm{wt}(y) \leq d \cdot W \cdot \sum_{y \in N(S)} \mathrm{wt}(y) = d \cdot W \cdot \mathrm{wt}(N(S)).$$

The lemma follows by combining all these inequalities. □

Remark 4. *We use G_f^* and R' to sketch an alternative and informal analysis of the tester of Dolev and Ron [DR11] for well-supported distributions $\mathcal{D}$, which can be similarly extended for general distributions as well. Let's assume that ϵ is a constant for convenience. (We also want to point out that our short analysis here has a worse dependency on ϵ than that of [DR11].) The tester starts by sampling a set $\mathbf{T}$ of $\tilde{\Theta}(\sqrt{n})$ strings from the sampling oracle. It then claims victory if there are two strings x and y from $\mathbf{T}$ such that $f(x) = 0$, $f(y) = 1$, and $(y, h(x))$ is an edge in G_f.*

Let $\mathbf{T}_1$ denote the set of 1-strings, and $\mathbf{T}_0$ denote the set of 0-strings from T. Let's assume both sets contain $\tilde{\Theta}(\sqrt{n})$ many strings, since otherwise the target function f is close to constant functions with respect to $\mathcal{D}$ and can be easily handled as special corner cases. Also let $\mathbf{R}'' \subseteq R'$ denote the set of $j \in R'$ such that $h(x) = j$ for some $x \in \mathbf{T}_0$. Since $\mathcal{D}(R') = wt(R') = \tilde{\Omega}(1)$, we have $wt(\mathbf{R}'') = \tilde{\Omega}(1/\sqrt{n})$ with high probability (here $\tilde{\Theta}(\sqrt{n})$ samples suffice because there are only n indices). When this happens, by Lemma 4.3.12 we have $wt(N(\mathbf{R}'')) = \tilde{\Omega}(1/\sqrt{n})$ as well. The tester then rejects if one of the (roughly $\tilde{\Theta}(\sqrt{n})$ many) samples in $\mathbf{T}_1$ lies in $N(\mathbf{R}'')$. This should happen with high probability if we set the hidden poly-logarithmic factor in the number of queries large enough.

Now we return to the analysis of our algorithm. This time we will focus on the left side of G_f^* rather than R' from the right side. Recall that $W = \mathrm{wt}(G_f^*)$. Let $L' \subseteq L^*$ denote the set of $y \in L^*$ such that $\deg(y) \geq W/2$ in G_f^*. Then similarly

$$W = \sum_{y \in L^*} \deg(y) \cdot \mathrm{wt}(y) \leq (W/2) \cdot \sum_{y \notin L'} \mathrm{wt}(y) + d \cdot W \cdot \sum_{y \in L'} \mathrm{wt}(y) \leq (W/2) + d \cdot W \cdot \mathrm{wt}(L'),$$

which implies that $\text{wt}(L') \geq 1/(2d)$. Our analysis of **MainDMconj** will heavily rely on G^*_f and $L' \subseteq L^*$.

We summarize below all the properties we need about G^*_f and L'.

Property 4.3.13. *Assume that $\mathcal{D}$ is well-supported with respect to f and $dist_{\mathcal{D}}(f, \text{MCONJ}) \geq \epsilon/2$. Then $G^*_f = (L^* \cup R^*, E^*)$ and $L' \subseteq L^*$ defined above have the following properties (letting $W = wt(G^*_f)$):*

1. *$W = \Omega(\epsilon)$ and $wt(L') \geq 1/(2d)$.*
2. *$\text{in-wt}(j) \leq d \cdot W \cdot wt(j)$ for all $j \in R^*$.*
3. *Every $y \in L'$ has $\deg(y) \geq \max(1, W/2)$.*

4.3.3.3 Proof of Lemma 4.3.5

We now prove Lemma 4.3.5. As assumed before, let $\mathcal{D}$ be a well-supported probability distribution with respect to $f : \{0,1\}^n \to \{0,1\}$, such that f is $(\epsilon/2)$-far from MCONJ with respect to $\mathcal{D}$. Let $G^*_f = (L^* \cup R^*, E^*)$ denote the bipartite graph defined using f and $\mathcal{D}$ in the previous section, with G^*_f and $L' \subseteq L^*$ satisfying Property 4.3.13.

Here is a sketch of the proof. We first analyze Stages 1 and 2 of **MainDMconj** and show that if a sequence of t samples $(\mathbf{y}^1, \ldots, \mathbf{y}^t)$ passes Stage 1 with high probability then it can be used to produce many sequences of sample strings that get rejected in Stage 2 with high probability. After that, we use G^*_f to show that, assuming $(f, \mathcal{D}, \epsilon)$ passes Stage 1 with high probability, then it must get rejected in Stage 2 with high probability, where Property 4.3.13 plays a crucial role. This completes the proof.

Analysis of Stages 1 and 2 First we assume without loss of generality that $f(1^n) = 1$; otherwise it is rejected at the beginning of Stage 0. As f is $(\epsilon/2)$-far from MCONJ with respect to $\mathcal{D}$, we have that both $\mathcal{D}(f^{-1}(0))$ and $\mathcal{D}(f^{-1}(1))$ are at least $\epsilon/2$. The former follows trivially from the fact that the all-1 function is in MCONJ. For the latter, we only

need to observe that the distance (with respect to $\mathcal{D}$) between f and the conjunction of all n variables is at most $\mathcal{D}(f^{-1}(1))$, given $f(1^n) = 1$.

Recall that since $\mathcal{D}$ is well-supported with respect to f, we can skip Stage 0 and have Stage 1 and each iteration of Stage 2 start by drawing $3t/\epsilon$ fresh samples from the sampling oracle. It follows directly from Chernoff bound that Stage 1 reaches Step 1.1 with probability $1-o(1)$. Let $\mathcal{D}^1$ denote the distribution of $\mathbf{y} \sim \mathcal{D}$ conditioning on $\mathbf{y} \in f^{-1}(1)$. Equivalently, we have that Stage 1 accepts with probability $o(1)$, and with probability $1-o(1)$ it draws a sequence of t samples $\mathbf{y}^1, \ldots, \mathbf{y}^t$ independently from $\mathcal{D}^1$ and then goes through Steps 1.1 and 1.2.

The same can be said about Stage 2: Stage 2 accepts with probability $o(1)$ by Chernoff bound and a union bound; with probability $1 - o(1)$, each iteration of Stage 2 draws a sequence of $t-1$ samples $\mathbf{y}^1, \ldots, \mathbf{y}^{t-1}$ from $\mathcal{D}^1$ as well as one sample $\mathbf{x} \sim \mathcal{D}$ conditioning on $\mathbf{x} \in f^{-1}(0)$. Since Steps 2.1 and 2.2 use only $\boldsymbol{\alpha} = h(\mathbf{x})$ but not the string $\mathbf{x}$ itself, this inspires us to introduce $\mathcal{D}^0$ as the distribution over R proportional to $\mathrm{wt}(j)$, $j \in R$. Hence equivalently, each iteration of Stage 2 draws an index $\boldsymbol{\alpha}$ from $\mathcal{D}^0$ and goes through Steps 2.1 and 2.2 using $\mathbf{y}^i$, $i \in [t-1]$ and $\boldsymbol{\alpha}$.

We introduce some more notation. Let $\mathcal{B} = (B_1, \ldots, B_t)$ be a sequence of t (not necessarily distinct) 1-sets of f (i.e., $f(B_i) = 1$). We refer to $\mathcal{B}$ as a 1-*sequence of length* t, and clearly we can view $\mathcal{B}$ as samples for Stage 1 of **MainDMconj** (by transforming each 1-set B_i into an 1-string y^i with $\textsc{Zero}(y^i) = B_i$). Let $B = \cup_i B_i$. We say $\mathcal{B}$ *passes Stage 1 with probability* c if B passes Steps 1.1 and 1.2 with probability c, without being rejected. Similarly, we let $\mathcal{B}' = (B_1, \ldots, B_{t-1})$ denote a 1-sequence of length $t-1$, with $B' = \cup_i B_i$, and let $\alpha \in R$. We also view $(\mathcal{B}', \alpha)$ as the samples for Stage 2, and we say $(\mathcal{B}', \alpha)$ *fails an iteration of Stage 2* with probability c if $(\mathcal{B}', \alpha)$ gets rejected in Steps 2.1 or 2.2 with probability c.

We now analyze 1-sequences $\mathcal{B} = (B_1, \ldots, B_t)$ that pass Stage 1 with high probability.

Let

$$B_i^* = B_i - \cup_{j \neq i} B_j, \quad \text{for each } i \in [t].$$

So B_i^* contains indices that are unique to B_i among all sets in $\mathcal{B}$. Let $I_{\mathcal{B}}$ denote the set of $i \in [t]$ such that $y_i \in L'$, where y_i is the 1-string with $\textsc{Zero}(y_i) = B_i$. Intuitively, $|I_{\mathcal{B}}|$ should be large with high probability since $\mathcal{D}(L') = \text{wt}(L')$ is large by Property 4.3.13. We say $\mathcal{B}$ is *strong* if $|I_{\mathcal{B}}| \geq t/(3d) = r/3$. Moreover, let $I_{\mathcal{B}}^*$ denote the set of $i \in I_{\mathcal{B}}$ such that $|B_i^*| \leq 6|B|/r$.

By an averaging argument we show that if $\mathcal{B}$ is strong then $|I_{\mathcal{B}}^*|$ is at least $r/6$.

Lemma 4.3.14. *If $\mathcal{B}$ is strong, then we have $|I_{\mathcal{B}}^*| \geq r/6$.*

Proof. As $\sum_i |B_i^*| \leq |B|$, the number of B_i with $|B_i^*| > 6|B|/r$ is at most $r/6$. The lemma then follows. □

Let $\mathcal{B} = (B_1, \ldots, B_t)$ denote a strong 1-sequence of length t and y_i denote the string with $\textsc{Zero}(y_i) = B_i$ for each $i \in [t]$. We use them to generate input pairs $(\mathcal{B}', \alpha)$ for Stage 2, where $\mathcal{B}'$ is a 1-sequence of length $t-1$ and $\alpha \in R$, as follows. For each pair (i, α) such that $i \in I_{\mathcal{B}}^*$ and $\alpha \in B_i \bigcap R^*$, we say $\mathcal{B}$ generates $(\mathcal{B}', \alpha)$ via (i, α) if

$$\mathcal{B}' = (B_1, \ldots, B_{i-1}, B_{i+1}, \ldots, B_t),$$

and we call such (i, α) a *valid* pair. Note that as B_i's are not necessarily distinct, $\mathcal{B}$ may generate the same pair $(\mathcal{B}', \alpha)$ via different (i, α) and (j, α), $i \neq j$. In the main technical lemma of this section below, Lemma 4.3.17, we show that if $\mathcal{B}$ is strong and passes Stage 1 with high probability, then there are many valid pairs (i, α) leading to pairs $(\mathcal{B}', \alpha)$ that fail Stage 2 with high probability. Before that we make a few observations. Recall $W = \text{wt}(G_f^*)$.

Observation 4.3.15. *Since $y_i \in L'$, we have $|B_i \cap R^*|$ equals to $\deg(y_i)$ in G_f^* and $|B_i \cap R^*| \geq \max(1, W/2)$. So the total number of valid pairs (i, α) is bounded from below by both $r/6$ and $rW/12$.*

Observation 4.3.16. *If a valid pair (i, α) satisfies $\alpha \in B_i \setminus B_i^*$ (i.e., α is shared by another B_j in $\mathcal{B}$), then it generates a pair $(\mathcal{B}', \alpha)$ that fails Stage 2 (in Step 2.1) with probability 1.*

Now we prove Lemma 4.3.17.

Lemma 4.3.17. *Assume that $\mathcal{B} = (B_1, \ldots, B_t)$ is a strong 1-sequence that passes Stage 1 with probability at least $1/2$. Then there are at least $\Omega(rW)$ many valid (i, α) such that the pair $(\mathcal{B}', \alpha)$ generated by $\mathcal{B}$ via (i, α) fails an iteration of Stage 2 with probability at least $\Omega(1)$ (a constant that does not depend on n or ϵ).*

Proof. For convenience, we use I to denote $I_{\mathcal{B}}^*$, with $|I| = \Omega(r)$ because $\mathcal{B}$ is strong (Lemma 4.3.14). We let $B^* = \cup_{i \in I} B_i^*$, and let $\Gamma = B^* \cap R^*$ (which can be empty). We first consider two special cases on $|\Gamma|$:

Case 1: $|\Gamma| > |B|/t$. Note that every $j \in \Gamma$ satisfies $f(\{j\}) = 0$. This implies that $\mathcal{B}$ would get rejected with probability $1 - o(1)$ in Step 1.1, contradicting the assumption that $\mathcal{B}$ passes it with probability $1/2$.

Case 2: $|\Gamma| < rW/24$. By Observation 4.3.15, the number of valid pairs (i, α) is at least $rW/12$. In this case, however, the number of valid pairs (i, α) such that $\alpha \in B_i^*$ is at most $rW/24$. Thus, the number of valid pairs (i, α) such that $\alpha \in B_i \setminus B_i^*$ is at least $rW/24$. The lemma follows from Observation 4.3.16.

In the rest of the proof we assume that $|B| \geq t|\Gamma|$ and $|\Gamma| = \Omega(rW)$. They together imply that

$$|B| \geq t|\Gamma| = \Omega(rtW). \tag{4.4}$$

For $\alpha \in \Gamma$ let $s_\alpha \in [t]$ be the unique index with $\alpha \in B_{s_\alpha}^*$. Now we will do some counting.

Let $\mathcal{Z}$ denote the set of all subsets $Z \subset B$ of size r such that $f(Z) = 1$. Since we assumed that $\mathcal{B}$ passes Stage 1 with probability at least $1/2$, it must be the case that

$$|\mathcal{Z}| \geq \left(1 - O\left(\frac{1}{s}\right)\right) \cdot \binom{|B|}{r}.$$

Fixing an $\alpha \in \Gamma$ with $\alpha \in B^*_{s_\alpha}$, we are interested in

$$\mathcal{S}_\alpha = \left\{P \cup \{\alpha\} : P \text{ is a subset of } B \setminus B^*_{s_\alpha} \text{ of size } r-1\right\} \quad \text{and} \quad N_\alpha = |\mathcal{S}_\alpha \cap \mathcal{Z}|.$$

We would like to prove a strong lower bound for $\sum_{\alpha\in\Gamma} N_\alpha$.

To give some intuition on the connection between N_α and the goal, notice that $B \setminus B^*_{s_\alpha} = \cup_{i\neq s_\alpha} B_i$. Let $(\mathcal{B}', \alpha)$ be the pair generated from $\mathcal{B}$ via (s_α, α). If a set $\mathbf{P}$ of size $r-1$ is drawn from $\cup_{i\neq s_\alpha} B_i$ uniformly at random, then the probability of $\mathbf{P}$ leading Step 2.2 to reject $(\mathcal{B}', \alpha)$, denoted by q_α, is

$$q_\alpha = \frac{N_\alpha}{\binom{|B\setminus B^*_{s_\alpha}|}{r-1}} \geq \frac{N_\alpha}{\binom{|B|}{r-1}} = \frac{N_\alpha}{\binom{|B|}{r} \cdot \frac{r}{|B|-r+1}} \geq \frac{N_\alpha}{\binom{|B|}{r}} \cdot \frac{|B|}{2r}, \tag{4.5}$$

where the last inequality used (4.4) that $|B| >> r$. So a strong bound for $\sum_{\alpha\in\Gamma} N_\alpha$ may lead us to the desired claim that q_α is large for most $\alpha \in \Gamma$. To bound $\sum_{\alpha\in\Gamma} N_\alpha$ and avoid double counting, let

$$\mathcal{S}'_\alpha = \left\{P \cup \{\alpha\} : P \text{ is a subset of } B \setminus (B^*_{s_\alpha} \cup \Gamma) \text{ of size } r-1\right\} \quad \text{and} \quad N'_\alpha = |\mathcal{S}'_\alpha \cap \mathcal{Z}|.$$

Since $\mathcal{S}'_\alpha \subseteq \mathcal{S}_\alpha$ and $\mathcal{S}'_\alpha$ are now pairwise disjoint, we have $\sum_\alpha N_\alpha \geq \sum_\alpha N'_\alpha$ and

$$\sum_{\alpha\in\Gamma} N'_\alpha = \left|(\cup_{\alpha\in\Gamma}\mathcal{S}'_\alpha) \cap \mathcal{Z}\right| \geq \left|\cup_{\alpha\in\Gamma} \mathcal{S}'_\alpha\right| + |\mathcal{Z}| - \binom{|B|}{r} \geq \sum_{\alpha\in\Gamma} |\mathcal{S}'_\alpha| - O\left(\frac{1}{s}\right) \cdot \binom{|B|}{r}.$$

On the other hand, by the definition of $I^*_{\mathcal{B}}$ we have $|B^*_{s_\alpha}| \leq 6|B|/r$. We also have

$\Gamma \leq |B|/t$. Thus

$$|\mathcal{S}'_\alpha| = \binom{|B \setminus (B^*_{s_\alpha} \cup \Gamma)|}{r-1} \geq \binom{|B| - (7|B|/r)}{r-1} = \Omega\left(\frac{r}{|B|} \cdot \binom{|B|}{r}\right), \tag{4.6}$$

where details of the last inequality can be found in Section 5.3.

Using $|\Gamma| = \Omega(rW)$ and $W = \Omega(\epsilon)$, $r = n^{1/3}$ and $|B| \leq n$, we have

$$\sum_{\alpha \in \Gamma} |\mathcal{S}'_\alpha| = \Omega\left(\frac{r|\Gamma|}{|B|} \cdot \binom{|B|}{r}\right) = \omega\left(\left(\frac{1}{s}\right) \cdot \binom{|B|}{r}\right).$$

As a result, we obtain the following lower bound for $\sum_{\alpha \in \Gamma} N_\alpha$:

$$\sum_{\alpha \in \Gamma} N_\alpha = \Omega\left(\frac{r|\Gamma|}{|B|} \cdot \binom{|B|}{r}\right).$$

Combining the connection between N_α and q_α from (4.5), we have $\sum_{\alpha \in \Gamma} q_\alpha = \Omega(|\Gamma|)$. Since $q_\alpha \leq 1$ (it is a probability) for all α, it follows easily that $q_\alpha = \Omega(1)$ for $\Omega(|\Gamma|)$ many α's in Γ. For each such α, (s_α, α) is a valid pair such that the pair $(\mathcal{B}', \alpha)$ generated from $\mathcal{B}$ via (s_α, α) gets rejected by Stage 2 with probability $\Omega(1)$.

The lemma then follows from $|\Gamma| = \Omega(rW)$. □

Finishing the proof of Lemma 4.3.5 Now based on Lemma 4.3.17 and properties of G^*_f and L', we are ready to finish the proof of Lemma 4.3.5.

Assume without loss of generality that Stage 1 of Algorithm 2 either accepts $(f, \mathcal{D})$ or passes it down to Stage 2 with probability at least 0.9; otherwise we are already done.

Recall that $\mathcal{D}^1$ is the distribution of $\mathbf{y} \sim \mathcal{D}$ conditioning on $\mathbf{y} \in f^{-1}(1)$. We abuse the notation a little bit and also use $\mathcal{D}^1$ to denote the corresponding distribution on 1-sets $\mathbf{B}$ (= $\textsc{Zero}(\mathbf{y})$). Given a 1-seqnence $\mathcal{B} = (B_1, \ldots, B_t)$ of length t, we write $p(\mathcal{B}) = \Pr_{\mathbf{B} \sim \mathcal{D}^1}[\mathbf{B} = B_1] \times \cdots \times \Pr_{\mathbf{B} \sim \mathcal{D}^1}[\mathbf{B} = B_t]$. From our discussion earlier, Stage 1 accepts $(f, \mathcal{D})$ with probability $o(1)$, and with probability $1 - o(1)$, it runs Steps 1.1 and 1.2 on a

1-sequence $\boldsymbol{\mathcal{B}}$ with each entry $\mathbf{B}_i$ drawn from $\mathcal{D}^1$ independently. This implies that

$$\sum_{\text{1-seq } \mathcal{B}} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \geq 0.8.$$

We focus on strong 1-sequences. We write S to denote the set of strong 1-sequences and let S' denote the set of strong 1-sequences that pass Stage 1 with probability at least $1/2$. Because $\mathcal{D}(L') = \text{wt}(L') \geq 1/(2d)$ we have that Stage 1 draws a strong $\mathcal{B}$ with probability $1 - o(1)$ by Chernoff bound. As a result, we have

$$\sum_{\mathcal{B} \in S} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \geq 0.8 - o(1) > 0.7.$$

But the LHS is at most

$$\sum_{\mathcal{B} \in S} p(\mathcal{B}) \cdot \Pr[\mathcal{B} \text{ passes Stage 1}] \leq (1/2) \cdot \sum_{\mathcal{B} \in S \setminus S'} p(\mathcal{B}) + \sum_{\mathcal{B} \in S'} p(\mathcal{B}) \leq (1/2) + \sum_{\mathcal{B} \in S'} p(\mathcal{B})$$

and thus, $\sum_{\mathcal{B} \in S'} p(\mathcal{B}) = \Omega(1)$. The remaining proof is to use this (combined with Lemma 4.3.17, G_f^* and L') to show that a random pair $(\mathcal{B}', \alpha)$ gets rejected in Stage 2 with high probability.

To this end, recall that $\mathcal{D}^0$ is the distribution over R proportional to $\text{wt}(j)$, $j \in R$. For each pair $(\mathcal{B}', \alpha)$, where $\mathcal{B}' = (B'_1, \ldots, B'_{t-1})$ is a 1-sequence of length $t-1$ and $\alpha \in R$, let $q(\mathcal{B}', \alpha) = \Pr_{\mathbf{B} \sim \mathcal{D}^1}[\mathbf{B} = B'_1] \times \cdots \times \Pr_{\mathbf{B} \sim \mathcal{D}^1}[\mathbf{B} = B'_{t-1}] \cdot \Pr_{\boldsymbol{\alpha} \sim \mathcal{D}^0}[\boldsymbol{\alpha} = \alpha]$.

Since Stage 2 consists of $d^* = d^2/\epsilon$ iterations, it suffices to show that

$$\sum_{(\mathcal{B}', \alpha)} q(\mathcal{B}', \alpha) \cdot \Pr[(\mathcal{B}', \alpha) \text{ fails an iteration of Stage 2}] = \Omega(\epsilon/d^2), \tag{4.7}$$

as Stage 2 either accepts with probability $o(1)$, or with probability $1 - o(1)$ each iteration of Stage 2 draws $(\mathcal{B}', \alpha)$ according to $q(\mathcal{B}', \alpha)$ and runs through Steps 2.1 and 2.2.

To take advantage of Lemma 4.3.17 we use T to denote the set of $(\mathcal{B}', \alpha)$ that is gener-

ated by a $\mathcal{B} = (B_1, \ldots, B_t)$ from S' via a pair (i, α) and fails an iteration of Stage 2 with probability $\Omega(1)$ (the same constant hidden in Lemma 4.3.17). For (4.7) it then suffices to show that

$$\sum_{(\mathcal{B}', \alpha) \in T} q(\mathcal{B}', \alpha) = \Omega(\epsilon / d^2). \tag{4.8}$$

Lemma 4.3.17 implies that for each $\mathcal{B}$ in S', there exist $\Omega(rW)$ many valid (i, α) such that the pair generated by $\mathcal{B}$ via (i, α) belongs to T (though these $(\mathcal{B}', \alpha)$'s are not necessarily distinct). We use $J_{\mathcal{B}}$ to denote these pairs of $\mathcal{B}$. We also write $(\mathcal{B}^i, \alpha)$ to denote the pair generated by $\mathcal{B}$ via (i, α) for convenience.

Then there is the following connection between probabilities $p(\mathcal{B})$ and $q(\mathcal{B}^i, \alpha)$:

$$q(\mathcal{B}^i, \alpha) = \frac{p(\mathcal{B})}{\Pr_{\mathbf{B} \sim \mathcal{D}^1}[\mathbf{B} = B_i]} \cdot \Pr_{\boldsymbol{\alpha} \sim \mathcal{D}^0}[\boldsymbol{\alpha} = \alpha] = p(\mathcal{B}) \cdot \frac{\mathcal{D}(f^{-1}(1))}{\mathcal{D}(B_i)} \cdot \frac{\mathrm{wt}(\alpha)}{\mathrm{wt}(R)} \geq \frac{\epsilon}{2} \cdot p(\mathcal{B}) \cdot \frac{\mathrm{wt}(\alpha)}{\mathcal{D}(B_i)},$$

where the inequality follows from $\mathrm{wt}(R) \leq 1$ and $\mathcal{D}(f^{-1}(1)) \geq \epsilon/2$ as discussed at the beginning of this section. The only obstacle for (4.8) is to handle the double counting. This is where G_f^* and L' help.

Consider the following sum (and its connection to (4.8)):

$$\sum_{\mathcal{B} \in S'} p(\mathcal{B}) \cdot |J_{\mathcal{B}}|. \tag{4.9}$$

On the one hand, as $|J_{\mathcal{B}}| = \Omega(rW)$ and $\sum_{\mathcal{B} \in S'} p(\mathcal{B}) = \Omega(1)$, the sum is $\Omega(rW)$. On the other hand, following the connection between probabilities $p(\mathcal{B})$ and $q(\mathcal{B}^i, \alpha)$ above we have:

$$(4.9) = \sum_{\mathcal{B} \in S'} \sum_{(i,\alpha) \in J_{\mathcal{B}}} p(\mathcal{B}) \leq \frac{2}{\epsilon} \cdot \sum_{\mathcal{B} \in S'} \sum_{(i,\alpha) \in J_{\mathcal{B}}} q(\mathcal{B}^i, \alpha) \cdot \frac{\mathcal{D}(B_i)}{\mathrm{wt}(\alpha)}. \tag{4.10}$$

Focusing on any fixed pair $(\mathcal{B}', \alpha)$ in T, the coefficient of $q(\mathcal{B}', \alpha)$ in (4.10) is given by

$$\frac{2}{\epsilon \cdot \mathrm{wt}(\alpha)} \cdot \sum_{\substack{\mathcal{B} \in \mathcal{S}', (i,\alpha) \in J_{\mathcal{B}} \\ \mathcal{B}^i = \mathcal{B}'}} \mathcal{D}(B_i). \tag{4.11}$$

However, fixing an $i \in [t]$, for $\mathcal{B}$ to generate $(\mathcal{B}', \alpha)$ via (i, α), a necessary condition is $\alpha \in B_i$. This implies that the string y satisfying $\textsc{Zero}(y) = B_i$ must be a neighbor of α in G_f^* (since $y \in L'$ by assuming $\mathcal{B}$ is strong and $\alpha \in R^*$ for each $(i, \alpha) \in J_{\mathcal{B}}$ following the definition of valid pairs). Considering the in-weight for α in G_f^*, it follows from Property 4.3.13 that the sum of (4.11) with i fixed is at most $2dW/\epsilon$ (with $\mathrm{wt}(\alpha)$ cancelled) and thus, the coefficient of $q(\mathcal{B}', \alpha)$ for each $(\mathcal{B}', \alpha) \in T$ in (4.10) is $O(tdW/\epsilon)$ (with t choices for i).

Combining all these inequalities, we have

$$\Omega(rW) = \sum_{\mathcal{B} \in S'} p(\mathcal{B}) \cdot |J_{\mathcal{B}}| \leq O\left(\frac{tdW}{\epsilon}\right) \cdot \sum_{(\mathcal{B}', \alpha) \in T} q(\mathcal{B}', \alpha),$$

and (4.7) follows. This finishes the proof of Lemma 4.3.5, and completes the analysis of **MainDMconj** .

4.4 Extending the proofs

In this section we extend our proofs to other classes of Boolean functions. More precisely, we will prove Theorem 1.4.4 and Theorem 1.4.5 about distribution-free testing of general conjunctions, decision lists, and linear threshold functions.

Extending the upper bound to general conjunctions First, we prove Theorem 1.4.4 using a simple reduction based on the following connection between MConj and Conj (the class of general conjunctions). Given a probability distribution $\mathcal{D}$ over $\{0,1\}^n$, we use $\mathcal{D}^{(C)}$ to denote the distribution with $\mathcal{D}(x) = \mathcal{D}(x^{(C)})$ for all x. Then we can show the following lemma:

Lemma 4.4.1. *Let $\mathcal{D}$ be a probability distribution over $\{0,1\}^n$, $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function, and $x^* \in \{0,1\}^n$ be a string such that $f(x^*) = 1$. Let $C = \textsc{Zero}(x^*)$,*

and let $g : \{0,1\}^n \to \{0,1\}$ *denote the Boolean function with* $g(x) = f(x^{(C)})$ *for all* $x \in \{0,1\}^n$. *Then we have*

1. *If* $f \in$ CONJ, *then* $g \in$ MCONJ.
2. *If* $dist_{\mathcal{D}}(f, \text{CONJ}) \geq \epsilon$, *then* $dist_{\mathcal{D}^{(C)}}(g, \text{MCONJ}) \geq \epsilon$.

Proof. Assume that $f \in$ CONJ. Then

$$f(x) = \left(\bigwedge_{i \in S} x_i\right) \bigwedge \left(\bigwedge_{i \in S'} \overline{x_i}\right)$$

for some disjoint sets $S, S' \subseteq [n]$. Since $f(x^*) = 1$ and $C =$ ZERO(x^*), we also have that $C \cap S = \emptyset$ and $S' \subseteq C$. As a result,

$$g(x) = f(x^{(C)}) = \bigwedge_{i \in S \cup S'} x_i \in \text{MCONJ},$$

and the first part of the lemma follows.

We prove the contrapositive of the second part. Assume that $dist_{\mathcal{D}^{(C)}}(g, h) < \epsilon$, for some $h \in$ MCONJ. Let h' denote the Boolean function with $h'(x) = h(x^{(C)})$. Then we have $h' \in$ CONJ and

$$\begin{aligned} dist_{\mathcal{D}}(f, \text{CONJ}) \leq dist_{\mathcal{D}}(f, h') &= \Pr_{x \in \mathcal{D}}\left[f(x) \neq h'(x)\right] \\ &= \Pr_{x \in \mathcal{D}}\left[g(x^{(C)}) \neq h(x^{(C)})\right] \\ &= \Pr_{x \in \mathcal{D}^{(C)}}\left[g(x) \neq h(x)\right] \\ &= dist_{\mathcal{D}^{(C)}}(g, h) \\ &< \epsilon. \end{aligned}$$

This finishes the proof of the second part of the lemma. □

Now we prove Theorem 1.4.4.

Proof of Theorem 1.4.4. Given Lemma 4.4.1, a distribution-free testing algorithm for Conj on $(f, \mathcal{D})$ starts by drawing $O(1/\epsilon)$ samples from $\mathcal{D}$ to find a string x^* with $f(x^*) = 1$. If no such string is found, the algorithm accepts; otherwise the algorithm takes the first sample x^* with $f(x^*) = 1$ and let $C = \textsc{Zero}(x^*)$. Then it runs our algorithm for Mconj to test whether $g(x) = f(x^{(C)})$ is in Mconj , or g is ϵ-far from Mconj with respect to $\mathcal{D}^{(C)}$ (note that we can simulate queries on g using the black-box oracle of f query by query; we can also simulate samples drawn from $\mathcal{D}^{(C)}$ using the sampling oracle of $\mathcal{D}$ sample by sample), and returns the same answer.

This algorithm clearly only has one-sided errors given Lemma 4.4.1 and the fact that our algorithm for testing Mconj only has one-sided errors. When f is ϵ-far from Conj, we have that $\mathcal{D}(f^{-1}(1)) \geq \epsilon$ because the all-0 function is in Conj (when both x_i and $\overline{x_i}$ appear in the conjunction for some $i \in [n]$). As a result, the algorithm finds an x^* with $f(x^*) = 1$ within the first $O(1/\epsilon)$ samples with high probability. It then follows from Lemma 4.4.1 that $dist_{\mathcal{D}^{(C)}}(g, \text{Mconj}) \geq \epsilon$, and with high probability $(g, \mathcal{D}^{(C)})$ gets rejected by our algorithm for testing Mconj with high probability, which leads $(f, \mathcal{D})$ to get rejected as well. □

Extending the lower bound to general conjunctions and decision lists Recall we let Conj, Dlist and Ltf denote the classes of all general conjunctions, decision lists, and linear threshold functions, respectively. Then it is easy to see that Mconj $\subset$ Conj $\subset$ Dlist $\subset$ Ltf. Here we prove Theorem 1.4.5 for general conjunctions and decision lists. For this purpose we follow the same strategy used in [GS09] and prove the following property on the distributions $\mathcal{NO}$ defined in Section 4.2.2:

Lemma 4.4.2. *With probability* $1 - o(1)$, $(f, \mathcal{D}_f)$ *drawn from* $\mathcal{NO}$ *satisfies* $dist_{\mathcal{D}_f}(f, \textsc{Dlist}) \geq 1/12$.

The same lower bound for Conj and Dlist then follows directly from Lemma 4.2.1, given that Mconj $\subset$ Conj $\subset$ Dlist and the fact that any pair $(g, \mathcal{D}_g)$ drawn from $\mathcal{YES}$

satisfies $g \in \textsc{Mconj}$.

Proof of Lemma 4.4.2. Let $(f, \mathcal{D}_f)$ be a pair drawn from $\mathcal{NO}$. Given any $i, j \in [m]$ such that $C_i \cap C_j = \emptyset$, we follow the same argument from Glasner and Servedio [GS09] to show that no decision list agrees with f on all of the following six strings $a^i, b^i, c^i, a^j, b^j, c^j$.

Assume for contradiction that a decision list h of length k (following the definition of $\textsc{Dlist}$ in Section 4.1):

$$(\ell_1, \beta_1), \ldots, (\ell_k, \beta_k), \beta_{k+1}$$

agrees with f on all six strings. Let $\textsc{first}(a)$ denote the index of the first literal ℓ_i in h that is satisfied by a string a, or $k+1$ if no literal is satisfied by a. Then we have

$$\min\left\{\textsc{first}(a^i), \textsc{first}(b^i)\right\} \leq \textsc{first}(c^i) \quad \text{and} \quad \min\left\{\textsc{first}(a^j), \textsc{first}(b^j)\right\} \leq \textsc{first}(c^j). \tag{4.12}$$

This is because by the definition of a^i, b^i and c^i, any literal satisfied by c^i is satisfied by either a^i or b^i. Next assume without loss of generality that

$$\textsc{first}(a^i) = \min\left\{\textsc{first}(a^i), \textsc{first}(b^i), \textsc{first}(a^j), \textsc{first}(b^j)\right\}. \tag{4.13}$$

By (4.12) we have that $\textsc{first}(c^i) \geq \textsc{first}(a^i)$. As $h(c^i) = f(c^i) = 0$ and $h(a^i) = f(a^i) = 1$, we have that $\textsc{first}(c^i) \neq \textsc{first}(a^i)$ and thus, $\textsc{first}(c^i) > \textsc{first}(a^i)$. This implies that the literal $\ell_{\textsc{first}(a^i)}$ must be x_k for some $k \in B_i$. As $C_i \cap C_j = \emptyset$, we have $B_i \cap C_j = \emptyset$ and thus, $c^j_k = 1$. This implies that $\textsc{first}(c^j) \leq \textsc{first}(a^i)$, and $\textsc{first}(c^j) < \textsc{first}(a^i)$ because they cannot be the same given that $h(c^j) = f(c^j) = 0$ and $h(a^i) = f(a^i) = 1$. However, $\textsc{first}(c^j) < \textsc{first}(a^i)$ contradicts with (4.12) and (4.13).

As a result, when C_i and C_j are disjoint, one has to flip at least one bit of f at the six strings to make it consistent with a decision list. The lemma then follows from the fact that, with probabiilty $1 - o(1)$, at least half of the pairs C_{2i-1} and C_{2i}, $i \in [m/2]$, are disjoint. □

Extending the lower bound to linear threshold functions Now we extend our lower bound to the distribution-free testing of linear threshold functions (LTF for short) and complete the proof for Theorem 1.4.5. We follow ideas from Glasner and Servedio [GS09] to construct a pair of probability distributions $\mathcal{YES}^*$ and $\mathcal{NO}^*$ with the following properties:

1. For each draw $(f, \mathcal{D}_f)$ from $\mathcal{YES}^*$, f is a LTF;
2. For each draw $(g, \mathcal{D}_g)$ from $\mathcal{NO}^*$, g is $(1/4)$-far from LTFs with respect to $\mathcal{D}_g$.

Let $q = n^{1/3}/\log^3 n$. We follow arguments from the proof of Lemma 4.2.1 to prove the following lemma:

Lemma 4.4.3. *Let T be a deterministic algorithm that, upon each input pair $(f, \mathcal{D})$, makes at most q queries to the black-box oracle of f and at most q queries to the sampling oracle of $\mathcal{D}$. Then we must have:*

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}}_\mathbf{f})\sim\mathcal{YES}^*} \left[T(\mathbf{f}, \boldsymbol{\mathcal{D}}_\mathbf{f}) \text{ accepts}\right] - \Pr_{(\mathbf{g},\boldsymbol{\mathcal{D}}_\mathbf{g})\sim\mathcal{NO}^*} \left[T(\mathbf{g}, \boldsymbol{\mathcal{D}}_\mathbf{g}) \text{ accepts}\right] \right| \leq \frac{1}{4}.$$

Our lower bound for LTFs then follows from Yao's mini-max principle.

We now define the two distributions $\mathcal{YES}^*$ and $\mathcal{NO}^*$. Recall the following parameters from the definition of $\mathcal{YES}$ and $\mathcal{NO}$ in Section 4.2.2:

$$\ell = n^{2/3} + 2, \quad m = n^{2/3}, \quad \text{and} \quad s = \log^2 n.$$

A draw $(\mathbf{f}, \boldsymbol{\mathcal{D}}_\mathbf{f})$ from the distribution $\mathcal{YES}^*$ is obtained using the following procedure:

1. Following the first five steps of the definition of $\mathcal{YES}$ in Section 4.2.2 to obtain $\mathbf{R}, \mathbf{C}_i, \mathbf{A}_i, \mathbf{B}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i$. For each $i \in [m]$, let $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$ be the strings with $\mathbf{A}_i = \textsc{Zero}(\mathbf{a}^i)$, $\mathbf{B}_i = \textsc{Zero}(\mathbf{b}^i)$, $\mathbf{C}_i = \textsc{Zero}(\mathbf{c}^i)$.

2. Define $\mathbf{u} : \{0,1\}^n \to \mathbb{Z}$ as following:

$$\mathbf{u}(x) = 10n^2 \sum_{k \in [n] \setminus \mathbf{R}} x_k + 5n \sum_{i \in [m]} x_{\boldsymbol{\alpha}_i} - \sum_{k \in [n]} x_k.$$

   Let $\theta = 10n^2(n/2 - 2m) + 5nm - (n - \ell/4)$.

3. Let $\mathbf{f} : \{0,1\}^n \to \{0,1\}$ be the function with $\mathbf{f}(x) = 1$ if $\mathbf{u}(x) \geq \theta$, and $\mathbf{f}(x) = 0$ otherwise. The distribution $\boldsymbol{\mathcal{D}}_{\mathbf{f}}$ is defined as follows: we put $1/4$ weight on 1^n, and for each $i \in [m]$, we put $1/(2m)$ weight on $\mathbf{b}^i$ and $1/(4m)$ weight on $\mathbf{c}^i$.

Clearly every pair $(\mathbf{f}, \boldsymbol{\mathcal{D}}_{\mathbf{f}})$ drawn from $\mathcal{YES}^*$ satisfies that $\mathbf{f}$ is an LTF. It is also easy to check that

$$\mathbf{f}(\mathbf{a}^i) = \mathbf{f}(\mathbf{c}^i) = \mathbf{f}(1^n) = 0 \quad \text{and} \quad \mathbf{f}(\mathbf{b}^i) = 1, \qquad \text{for each } i \in [m].$$

A draw $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}})$ from the distribution $\mathcal{NO}^*$ is obtained in the following procedure:

1. Following the definition of $\mathcal{YES}$ in Section 4.2.2 to obtain $\mathbf{R}, \mathbf{C}_i, \mathbf{A}_i, \mathbf{B}_i, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \mathbf{c}^i, \mathbf{a}^i, \mathbf{b}^i$.
2. We follow the same definition of a string being *i-special* for some $i \in [m]$ as in Section 4.2.2. Let

$$\mathbf{J}(x) = \big\{ i \in [m] : x \text{ is } i\text{-special} \big\}, \qquad \text{for each } x \in \{0,1\}^n.$$

3. Define $\mathbf{v} : \{0,1\}^n \to \mathbb{Z}$ as following:

$$\mathbf{v}(x) = 10n^2 \sum_{k \in [n] \setminus \mathbf{R}} x_k + 5n \left(|\mathbf{J}(x)| + \sum_{i \in [m] \setminus J(x)} x_{\boldsymbol{\alpha}_i} \right) - \sum_{k \in [n]} x_k.$$

   Let θ be the same threshold: $\theta = 10n^2(n/2 - 2m) + 5nm - (n - \ell/4)$.

4. Let $\mathbf{g} : \{0,1\}^n \to \{0,1\}$ be the function with $\mathbf{g}(x) = 1$ if $\mathbf{v}(x) \geq \theta$, and $\mathbf{g}(x) = 0$ otherwise. $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$ is defined as follows: we put $1/4$ weight on 1^n and $1/(4m)$ weight on each of $\mathbf{a}^i, \mathbf{b}^i, \mathbf{c}^i$, $i \in [m]$.

For each pair $(\mathbf{g}, \boldsymbol{\mathcal{D}}_{\mathbf{g}}) \sim \mathcal{NO}^*$ and each $i \in [m]$, we still have $\mathbf{g}(\mathbf{c}^i) = \mathbf{g}(1^n) = 0, \mathbf{g}(\mathbf{b}^i) = 1$ but $\mathbf{g}(\mathbf{a}^i)$ is flipped to 1 (since $\mathbf{a}^i$ is i-special). As $\mathbf{C}_i = \mathbf{A}_i \cup \mathbf{B}_i$, we have that at least one of $\mathbf{g}(\mathbf{a}^i), \mathbf{g}(\mathbf{b}^i), \mathbf{g}(\mathbf{c}^i), \mathbf{g}(1^n)$ needs to be flipped to make $\mathbf{g}$ an LTF. It follows from the definition of $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$ that $\mathbf{g}$ is $(1/4)$-far from LTFs with respect to $\boldsymbol{\mathcal{D}}_{\mathbf{g}}$, as desired.

Then it's enough to prove Lemma 4.4.3.

Let T be any deterministic algorithm that makes q queries to each of the two oracles. We follow Section 4.2.3 and assume that T has access to the following *strong sampling oracle*:

1. When the sampling oracle returns c^i for some $i \in [m]$, it returns the special index α_i as well;
2. For convenience we also assume without loss of generality that the oracle always returns a sample drawn from the marginal distribution of $\mathcal{D}$ within $\{a^i, b^i, c^i\}$ since samples of 1^n are not useful in distinguishing $\mathcal{YES}^*$ and $\mathcal{NO}^*$.

We show that Lemma 4.4.3 holds even if T receives q samples from the strong sampling oracle and makes q queries to the black-box oracle. We follow the same notation introduced in Section 4.2.3. Given a sequence $Q = ((D_i, \gamma_i) : i \in [q])$ of samples that T receives from the strong sampling oracle, let $\Gamma(Q)$ denote the set of integer γ_i's in Q, let $S(Q) = \cup_{i \in [q]} D_i$, and let $I(Q)$ denote the set of $i \in [q]$ with $|D_i| = \ell/2$.

Similarly as in Section 4.2.3, we derive from T a new deterministic oracle algorithm T' that has *no access* to the black-box oracle but receives R in addition to the sequence of samples Q at the beginning. We show that T' cannot distinguish the two distributions $\mathcal{YES}^*$ and $\mathcal{NO}^*$ (Lemma 4.4.4), but T' agrees with T most of the time (Lemma 4.4.5 and Lemma 4.4.6), from which Lemma 4.4.3 follows.

The new algorithm T' works as follows:

> Given R and Q, T' simulates T on Q as follows (note that T is not given R but receives only Q in the sampling phase): whenever T queries about $x \in \{0,1\}^n$, T' does not query the oracle but computes
>
> $$\phi(x) = 10n^2 \sum_{k \in [n] \setminus R} x_k + 5n\left(m - |I'(x)|\right) - \sum_{k \in [n]} x_k,$$
>
> where $I'(x) = \textsc{Zero}(x) \cap \Gamma(Q)$, i.e., the set of all α_i's in $\Gamma(Q)$ revealed in the sampling phase such that $x_{\alpha_i} = 0$. T' then computes the response for x as 1 if $\phi(x) \geq \theta$, and as 0 otherwise. It then proceeds as T dose upon this response. At the end of the simulation, T' returns the same answer as T.

Now we are ready to prove the three lemmas mentioned above.

The first lemma is to show that a deterministic oracle algorithm with no access to the black-box oracle cannot distinguish $\mathcal{YES}^*$ and $\mathcal{NO}^*$ distributions with high probability.

Lemma 4.4.4. *Let T^* be any deterministic oracle algorithm that, upon each input pair $(f, \mathcal{D})$ drawn from either $\mathcal{YES}^*$ or $\mathcal{NO}^*$, receives R and a sequence Q of q samples from $\mathcal{D}$ but has no access to the black-box oracle of f. Then*

$$\left| \Pr_{(\mathbf{f}, \boldsymbol{\mathcal{D}}_\mathbf{f}) \sim \mathcal{YES}} \left[T^* \textit{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}_\mathbf{f}) \right] - \Pr_{(\mathbf{g}, \boldsymbol{\mathcal{D}}_\mathbf{g}) \sim \mathcal{NO}} \left[T^* \textit{ accepts } (\mathbf{g}, \boldsymbol{\mathcal{D}}_\mathbf{g}) \right] \right| = o(1).$$

Proof. The proof of the lemma is essentially the same as the proof of Lemma 4.2.2. The only difference here is that the distribution $\mathcal{D}$ is also supported on 1^n. But because $\mathcal{D}_f(1^n) = \mathcal{D}_g(1^n) = 1/4$ in both $\mathcal{YES}^*$ and $\mathcal{NO}^*$, the same proof works here. □

Next we show that T' agrees with T most of the time when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{YES}^*$:

Lemma 4.4.5. *Let T be a deterministic oracle algorithm that, upon each input pair $(f, \mathcal{D})$, makes at most q queries to the strong sampling oracle of $\mathcal{D}$ and the black-box oracle of f*

each, and let T' be the algorithm defined using T as above. Then

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}^*} \left[T \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\right] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{YES}^*} \left[T' \text{ accepts } (\mathbf{f},\boldsymbol{\mathcal{D}})\right] \right| \leq 0.1.$$

Proof. Fix a sequence Q of q samples that T and T' receive at the beginning. We prove the same statement conditioning on receiving Q. Let $\mathcal{R}_Q$ denote the distribution of the random set $\mathbf{R}$, conditioning on Q. We let T_Q denote the binary decision tree of T of depth q upon receiving Q, and let $w'(R)$ denote the leaf that T' reaches given fixed R for each R in the support of $\mathcal{R}_Q$.

Following the same definition and argument used in the proof of Lemma 4.2.3 (as $\phi(x) < \theta$ if one of the variables outside of R is set to 0), it suffices to show for every R in the support of $\mathcal{R}_Q$ such that $w = w'(R)$ is a *good* leaf (see the definition in the proof of Lemma 4.2.3), we have that T reaches w with high probability (conditioning on both fixed Q and R). Note that $\mathbf{u}(x)$ in the $\mathcal{YES}^*$ distribution can also be written as:

$$\mathbf{u}(x) = 10n^2 \sum_{k\in[n]\setminus\mathbf{R}} x_k + 5n\left(m - |\mathbf{I}(x)|\right) - \sum_{k\in[n]} x_k,$$

where $\mathbf{I}(x)$ here is the set of all special indices $\boldsymbol{\alpha}_i$'s, $i \in [m]$, such that $x_{\boldsymbol{\alpha}_i} = 0$. Since $\boldsymbol{\phi}(x) \geq \mathbf{u}(x)$, T does not reach w if and only if one of the strings x along the path from the root of T_Q to w satisfies

$$|\mathbf{I}'(x)| < |\mathbf{I}(x)| \quad \text{and} \quad \boldsymbol{\phi}(x) \geq \theta > \mathbf{u}(x).$$

Given that $\Gamma(Q)$ contains all special indices α_i's in $S(Q)$ (as $\mathbf{a}^i$'s are not in the support of $\boldsymbol{\mathcal{D}}$) it must be the case that $x_{\boldsymbol{\alpha}_i} = 0$ for some special index $\boldsymbol{\alpha}_i \notin S(Q)$ and thus, $\boldsymbol{\alpha}_i \in H_w$ for some $i \in [m]$ (see the definition of H_w in the proof of Lemma 4.2.3). This is exactly the same event analyzed in the proof of Lemma 4.2.3, with its probability bounded from above by 0.1. This finishes the proof of the lemma. □

Finally we show that T' agrees with T most of the time when $(\mathbf{f}, \boldsymbol{\mathcal{D}}) \sim \mathcal{NO}^*$:

Lemma 4.4.6. *Let T be a deterministic oracle algorithm that, upon each input pair $(f, \mathcal{D})$, makes q queries to the strong sampling oracle of $\mathcal{D}$ and the black-box oracle of f each, and let T' be the algorithm defined using T as above. Then*

$$\left| \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}^*} \left[T \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}) \right] - \Pr_{(\mathbf{f},\boldsymbol{\mathcal{D}})\sim\mathcal{NO}^*} \left[T' \text{ accepts } (\mathbf{f}, \boldsymbol{\mathcal{D}}) \right] \right| \leq 0.1.$$

Proof. Following Definition 4.2.4 and Lemma 4.2.5, the event E of $\mathbf{Q}$ being *separated* (with respect to $(\mathbf{f}, \boldsymbol{\mathcal{D}})$) happens with probability $1 - o(1)$. Let $\mathcal{Q}_E$ denote the probability distribution of $\mathbf{Q}$ conditioning on event E happens. Fix a sequence Q in the support of $\mathcal{Q}_E$. Below we prove the statement of the lemma conditioning on both fixed Q and event E happens, and we let $\mathcal{R}_{Q,E}$ be the distribution of $\mathbf{R}$ under this condition.

Similar to the proof of Lemma 4.2.10, it suffices to show that for every R in the support of $\mathcal{R}_{Q,E}$ such that $w = w'(R)$ is a good leaf, T reaches w with high probability, conditioning on fixed R, Q and event E happens.

Note that $\mathbf{v}(x)$ from the $\mathcal{NO}^*$ distribution can be also written as:

$$\mathbf{v}(x) = 10n^2 \sum_{k\in[n]\setminus\mathbf{R}} x_k + 5n\left(m - |\mathbf{I}(x)|\right) - \sum_{k\in[n]} x_k,$$

where $\mathbf{I}(x)$ is the set of all $\boldsymbol{\alpha}_i$'s, $i \in [m]$, such that $x_{\boldsymbol{\alpha}_i} = 0$ and x is not i-special. Then T does not reach w only if for some x along the path from the root of T_Q to w, either $\boldsymbol{\phi}(x) \geq \theta > \mathbf{v}(x)$ or $\mathbf{v}(x) \geq \theta > \boldsymbol{\phi}(x)$.

When $\boldsymbol{\phi}(x) \geq \theta > \mathbf{v}(x)$, we have $|\mathbf{I}(x)| > |\mathbf{I}'(x)|$ and thus, one of the following two events must hold:

Event E_0^*: $\boldsymbol{\phi}(x) \geq \theta$ (so w is in the 1-subtree of the node making query x) and $x_{\boldsymbol{\alpha}_k} = 0$ for some $\boldsymbol{\alpha}_k \notin S(Q)$;

Event $E^*_{1,2}$: $\boldsymbol{\phi}(x) \geq \theta$, $x_{\boldsymbol{\alpha}_k} = 0$ for some $\boldsymbol{\alpha}_k \in S(Q)$ but $\boldsymbol{\alpha}_k \notin \Gamma(Q)$, and x is not k-special.

For the case when $\mathbf{v}(x) \geq \theta > \boldsymbol{\phi}(x)$, we have $|\mathbf{I}'(x)| > |\mathbf{I}(x)|$ and thus, the following event must hold:

Event E^*_3: $x_{\boldsymbol{\alpha}_k} = 0$ for some $\boldsymbol{\alpha}_k \in \Gamma(x)$ and x is k-special.

Note that E^*_0 is the same event as E_0, $E^*_{1,2}$ is the same event as the union of E_1 and E_2, and E^*_3 is the same event as E_3 in the proof of Lemma 4.2.10. The lemma follows from bounds on their probabilities given in the proof of Lemma 4.2.10. □

Lemma 4.4.3 then follows from Lemmas 4.4.4, 4.4.5, and 4.4.6.

This page intentionally left blank.

Chapter 5

Deferred Proofs

5.1 A claim about products

Recall Bernoulli's inequality: for every real number $a \geq 1$ and real number $x \geq -1$, we have

$$(1+x)^a \geq 1 + ax,$$

and for every real number $0 \leq a \leq 1$ and real number $x \geq -1$, we have

$$(1+x)^a \leq 1 + ax.$$

We prove the following claim used in Section 2.3.2.4.

Claim 5.1.1. *Let $t \leq n^{2/3}$ and $c_1, \ldots, c_t \in \mathbb{R}$ be numbers with $|c_i| \leq \log^2 n/\sqrt{n}$. We have*

$$\prod_{i \in [t]} (1 - c_i) \geq (1 - o(1)) \cdot \left(1 - \sum_{i \in [t]} c_i\right),$$

where the asymptotic notation is with respect to n.

Proof. Let $\beta = \log^2 n/\sqrt{n}$. Assume without loss of generality that

$$c_1, \ldots, c_k \geq 0 \quad \text{and} \quad c_{k+1}, \ldots, c_t < 0$$

for some $k \leq t$. Let $\delta_i = c_i/\beta$ for $i \leq k$ and $\tau_j = -c_j/\beta$ for $j > k$. Thus, $\delta_i, \tau_j \in [0, 1]$

and

$$\sum_{i\in[t]} c_i = \beta\left(\sum_{i\le k}\delta_i - \sum_{j>k}\tau_j\right).$$

Let $\Delta = \sum_{i\le k}\delta_i - \sum_{j>k}\tau_i$. By Bernoulli's inequality, we also have

$$1-c_i \ge (1-\beta)^{\delta_i} \quad \text{and} \quad 1-c_j \ge (1+\beta)^{\tau_j}.$$

As a result, it remains to show that

$$(1-\beta)^{\sum_{i\le k}\delta_i}\cdot(1+\beta)^{\sum_{j>k}\tau_j} \ge (1-o(1))\,(1-\beta\Delta).$$

We consider two cases: $\Delta > 0$ or $\Delta \le 0$. If $\Delta > 0$, we have

$$(1-\beta)^{\sum_i\delta_i}\cdot(1+\beta)^{\sum_j\tau_i} = (1-\beta)^{\Delta}\cdot\left(1-\beta^2\right)^{\sum_j\tau_j} \ge (1-o(1))\cdot(1-\beta)^{\Delta}$$

using $\beta^2 = \log^4/n$ and $\sum_j \tau_j \le n^{2/3}$. When $\Delta \ge 1$ it follows by Bernoulli's inequality that $(1-\beta)^{\Delta} \ge 1-\beta\Delta$ and we are done. When $0 < \Delta < 1$, we have from $\beta = o(1)$ and $\beta\Delta = o(1)$ that

$$(1-\beta)^{\Delta} > 1-\beta \ge (1-o(1))\cdot(1-\beta\Delta).$$

The case when $\Delta \le 0$ is similar:

$$(1-\beta)^{\sum_i\delta_i}\cdot(1+\beta)^{\sum_j\tau_i} = (1+\beta)^{-\Delta}\cdot\left(1-\beta^2\right)^{\sum_i\delta_i} \ge (1-o(1))\cdot(1+\beta)^{-\Delta}.$$

When $\Delta \le -1$, it follows from Bernoulli's inequality that $(1+\beta)^{-\Delta} \ge 1-\beta\Delta$ and we are done. If $-1 < \Delta \le 0$, we have from $-\beta\Delta = o(1)$ that $(1+\beta)^{-\Delta} > 1 > (1-o(1))\cdot(1-\beta\Delta)$. □

5.2 Proof of Claim 4.2.6

We use the following folklore extension of the standard Chernoff bound:

Lemma 5.2.1. *Let $p \in [0,1]$ and $\mathbf{X}_1, \ldots, \mathbf{X}_n$ be a sequence of (not necessarily independent) $\{0,1\}$-valued random variables. Let $\mathbf{X} = \sum_{i\in[n]} \mathbf{X}_i$. If for any $i \in [n]$ and any $b_1, \ldots, b_{i-1} \in \{0,1\}$:*

$$\Pr\left[\mathbf{X}_i = 1 \mid \mathbf{X}_1 = b_1, \cdots, \mathbf{X}_{i-1} = b_{i-1}\right] \leq p,$$

then we have $\Pr[\mathbf{X} \geq (1+\delta)\cdot pn] \leq e^{-\delta^2 pn/3}$.

Now we prove Claim 4.2.6. Let's fix an $i \in [q]$ and the $2\log^2 n$ blocks in L_i. Then we sample all other $q-1$ many $\mathbf{L}_j$'s and bound the probability that the number of blocks in L_i that appear in $\cup_{k\neq i}\mathbf{L}_k$ is more than $\log^2 n/16$. We use the following procedure to sample $\mathbf{L}_j$'s: for each $j \neq i$ sample a sequence of $4\log^2 n$ blocks uniformly at random with replacement and set $\mathbf{L}_j$ to be the union of the first $2\log^2 n$ distinct blocks sampled. This procedure, denoted by $\mathcal{A}$, fails if for some j, there are less than $2\log^2 n$ distinct blocks from the $4\log^2 n$ samples for $\mathbf{L}_j$. When it succeeds, $\mathcal{A}$ yields the desired uniform and independent distribution. We claim that $\mathcal{A}$ succeeds with probability $1 - e^{-\Omega(r)}$.

To see this, for each j, its kth sample is the same as one of the previous $k-1$ samples with probability at most $(k-1)/r \leq 4\log^2 n/r$, no matter what the outcomes of the first $k-1$ samples are. By Lemma 5.2.1, $\mathcal{A}$ failed at $\mathbf{L}_j$ with probability $e^{-\Omega(r)}$ because this happens only if more than $2\log^2 n$ samples have appeared before. By a union bound on j, $\mathcal{A}$ succeeds with probability $1 - e^{-\Omega(r)}$.

Let $\mathbf{U}$ denote the union of all $(q-1)\cdot(4\log^2 n)$ blocks sampled by $\mathcal{A}$. Then the probability that the number of blocks in L_i that appear in $\cup_{k\neq i}\mathbf{L}_k$ is more than $\log^2 n/16$ is at most:

$$\begin{aligned}&\Pr\left[\mathbf{U} \text{ has } > \log^2 n/16 \text{ blocks of } L_i \mid \mathcal{A} \text{ succeeds}\right]\\ \leq &\frac{\Pr\left[\mathbf{U} \text{ has } > \log^2 n/16 \text{ blocks of } L_i\right]}{\Pr\left[\mathcal{A} \text{ succeeds}\right]}.\end{aligned}$$

Using Chernoff bound, the probability of $\mathbf{U}$ having more than $\log^2 n/16$ blocks of L_i is at

most $n^{-\Omega(\log n)}$. Claim 4.2.6 then follows from $\Pr[\mathcal{A} \text{ succeeds}] \geq 1 - e^{-\Omega(r)}$ and a union bound on $i \in [q]$.

5.3 Proof of Inequality (4.6)

We prove the last step of (4.6) in this section. Let $k = |B| = \Omega(rtW) >> r$ and let $\delta = 7/r$. Then we have:

$$\begin{aligned}
\frac{\binom{k}{r}}{\binom{k-\delta k}{r-1}} &= \frac{1}{r} \cdot \frac{(k-\delta k+1)(k-\delta k+2)\cdots k}{(k-\delta k-r+2)(k-\delta k-r+3)\cdots(k-r)} \\
&= \frac{k}{r} \cdot \frac{k-\delta k+1}{k-\delta k-r+2} \cdot \frac{k-\delta k+2}{k-\delta k-r+3} \cdots \frac{k-1}{k-r} \\
&\leq \frac{k}{r} \cdot \left(\frac{k-\delta k+1}{k-\delta k-r+2}\right)^{\delta k-1} \leq \frac{k}{r} \cdot \left(1+\frac{2r}{k}\right)^{\delta k} = O\left(\frac{k}{r}\right).
\end{aligned}$$

Conclusion

In this thesis we study property testing of Boolean functions under both standard setting (measuring the distance between functions with respect to the uniform distribution) and distribution-free setting (measuring the distance with respect to a fixed but unknown distribution). Here we summarize our results and also discuss open problems and directions for related future work.

In Chapter 2 we present new lower bounds for testing of monotonicity and unateness under the standard model:

- We prove a lower bound of $\tilde{\Omega}(n^{1/3})$[1] for the query complexity of adaptive testing of monotonicity, improving the previous best lower bound of $\tilde{\Omega}(n^{1/4})$ by Belov and Blais [BB16]. We also show that the analysis based on our current techniques and construction is tight: there is an $\tilde{O}(n^{1/3})$-query algorithm that distinguishes the two distributions we construct for the proof.
- We also prove a lower bound of $\tilde{\Omega}(n^{2/3})$ for adaptive testing of unateness. This result shows for the first time that the standard property testing of unateness is strictly harder (with a polynomial gap) than testing of monotonicity.
- For non-adaptive testing of unateness with one-sided errors, we also prove a lower bound of $\tilde{\Omega}(n)$, which matches the upper bound of $\tilde{O}(n)$ (for constant ϵ) by Chakrabarty and Seshadhri [CS16], up to poly-logarithmic factors of n. Combined with the $\tilde{O}(n^{3/4})$-query algorithm from [CWX17b], it also shows that adaptivity

[1] All our lower bounds, just like most previous lower bounds, hold for some constant distance parameters $\epsilon \in (0, 1)$.

helps for testing unateness.

In Chapter 3 we discuss about distribution-free testing of k-juntas:

- We present an adaptive distribution-free tester for k-juntas with query complexity $\tilde{O}(k^2/\epsilon)$, improving the previous upper bound of $O(2^k/\epsilon)$ that follows by combining the work of Halevy and Kushilevitz [HK07] and the work of Alon and Weinstein [AA12].
- We also show an exponential lower bound of $\Omega(2^{k/3})$ for the query complexity of non-adaptive distribution-free testers for k-juntas. This result illustrates the hardness of distribution-free testing, and combining our polynomial-query adaptive algorithm it shows that adaptivity provides an exponential improvement for testing k-juntas under the distribution-free setting.

In Chapter 4 we discuss about distribution-free testing of other basic Boolean functions:

- We show a lower bound of $\tilde{\Omega}(n^{1/3})$ for distribution-free testing of monotone conjunctions, general conjunctions, decision lists and linear threshold functions.
- For distribution-free testing of monotone conjunctions and general conjunctions, we also show an $\tilde{O}(n^{1/3}/\epsilon^5)$-query adaptive algorithm, which pins down the optimal query complexity of these tasks at $\tilde{\Theta}(n^{1/3})$ for some constant ϵ, if we ignore the poly-logarithmic factors.

Open problems Here are some related open problems, as well as our thoughts on them at the moment (for simplicity let's fix ϵ as some constant):

- For the adaptive testing of monotonicity, the current best upper bound for this problem still remains from the $\tilde{O}(\sqrt{n})$-query non-adaptive algorithm by Khot et al. [KMS15], and there is a gap between this bound and our lower bound of $\tilde{\Omega}(n^{1/3})$. We

conjecture that this non-adaptive algorithm is optimal even in the case of adaptive testing, and it is the lower bound side that can be improved, with a new construction for the distributions. Here are some very high-level (maybe incorrect) intuition why we think this is the case: (1) we improve the lower bound of $\tilde{\Omega}(n^{1/4})$ to $\tilde{\Omega}(n^{1/3})$ mainly by restricting the speed of searching special variables (that lead to violation to monotonicity) both among 0's and among 1's in our argument, but not at the same time. We can only restrict the speed of searching over 1's in the first layer and over 0's in the second layer of our functions (though we can also derive some "soft" restrictions on 1's in the second layer), and a restriction for both at the same time will give us a lower bound of $\tilde{\Omega}(n^{1/2})$; (2) we have a lower bound of $\tilde{\Omega}(\sqrt{n})$ that matches the current upper bound for the case of non-adaptive testing of monotonicity, and we are not aware of many adaptive procedures that can be useful for testing monotonicity. One of the major candidates is the binary search procedure, which is however hard to analyze.

- We are facing the similar situation for the case of adaptive unateness testing. The current best upper bound comes from the $\tilde{O}(n^{3/4})$-query algorithm given in our work [CWX17b]. Assuming the $\tilde{O}(\sqrt{n})$-query algorithm from [KMS15] is optimal for adaptive testing of monotonicity, it is reasonable to conjecture that this $\tilde{O}(n^{3/4})$-query algorithm also has the best query complexity for testing unateness adaptively: roughly speaking, [KMS15] shows that one can use $\tilde{O}(\sqrt{n})$ queries to find a violation to monotonicity of a far-from-monotone function, by identifying an anti-monotone edge along one of $\tilde{O}(\sqrt{n})$ many candidate directions randomly selected at the beginning. Assuming this is also the best one can do for unateness testing (but each time an anti-monotone edge or a monotone edge is identified with probability $1/2$, following the definition of unateness), and assuming these candidate directions are fixed, then we need to repeat above process about $\tilde{O}(n^{1/4})$ times to find edges along a certain direction twice and get a violation to unateness (one time with a monotone

edge, and the other with an anti-monotone edge), following birthday paradox. Then in total we need $\tilde{O}(n^{3/4})$ many queries. Of course, there are a lot of details to be filled in this argument, and it is also possible that one closes this gap for adaptive testing of unateness before the gap for testing of monotonicity.

- For the distribution-free testing of k-juntas, an interesting open problem comes from the adaptive setting. The best lower bound is still $\tilde{\Omega}(k)$ from the standard uniform distribution setting [CG04], and it is not clear whether we can improve our upper bound of $\tilde{O}(k^2)$ to $\tilde{O}(k)$ to match this lower bound. When we design testing algorithms for k-juntas, in order to remove the dependency on n in the query complexity we usually randomly partition $[n]$ into $poly(k)$ many small blocks, conduct binary search over blocks, and eventually find relevant blocks rather than relevant variables, while the definition of k-juntas is based on the latter. Blais managed to bridge these two notions in [Bla09] with a technical analysis based on the influence of indices and the Efron-Stein orthogonal decomposition of functions under the standard uniform distribution setting, while for the distribution-free setting our $\tilde{O}(k^2)$-query algorithm, in some sense, circumvents such issue with an artificial way of sampling strings, which also leads to our final query complexity being quadratic in k. We are interested in whether Blais's argument can be generalized to the distribution-free setting and gives us an $\tilde{O}(k)$-query algorithm .

Bibliography

[AA12] N. Alon and A.Weinstein. "Local correction of juntas." In: *Inf. Process. Lett.* 112(6) (2012), pp. 223–226.

[AC06] N. Ailon and B. Chazelle. "Information theory in property testing and monotonicity testing in higher dimension." In: *Information and Computation* 204 (2006), pp. 1704–1717.

[Ail+07] N. Ailon et al. "Estimating the distance to a monotone function." In: *Random Structures and Algorithms* 31.3 (2007), pp. 371–383.

[BB16] A. Belovs and E. Blais. "A Polynomial Lower Bound for Testing Monotonicity." In: *Proceedings of the 48th ACM Symposium on Theory of Computing*. 2016.

[BBM12] E. Blais, J. Brody, and K. Matulef. "Property Testing Lower Bounds via Communication Complexity." In: *Computational Complexity* 21.2 (2012), pp. 311–358.

[BKR04] T. Batu, R. Kumar, and R. Rubinfeld. "Sublinear algorithms for testing monotone and unimodal distributions." In: *Proceedings of the 36th ACM Symposium on Theory of Computing*. 2004, pp. 381–390.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. "Self-testing/correcting with applications to numerical problems." In: *J. Comp. Sys. Sci.* 47 (1993). Earlier version in STOC'90, pp. 549–595.

[BRY13] E. Blais, S. Raskhodnikova, and G. Yaroslavtsev. "Lower Bounds for Testing Properties of Functions on Hypergrid Domains." In: *Electronic Colloquium on Computational Complexity (ECCC)* 20 (2013), p. 36.

[Bal+16] R. Baleshzar et al. "Testing unateness of real-valued functions." arXiv:1608.07652. 2016.

[Bal+17] R. Baleshzar et al. "Optimal Unateness Testers for Real-Valued Functions: Adaptivity Helps." In: *Proceedings of the 44th International Colloquium on Automata, Languages and Programming (ICALP '2017)*. 2017.

[Bla08] E. Blais. "Improved bounds for testing juntas." In: *Proc. RANDOM*. 2008, pp. 317–330.

[Bla09] E. Blais. "Testing juntas nearly optimally." In: *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*. 2009, pp. 151–158.

[Bri+12] J. Briët et al. "Monotonicity testing and shortest-path routing on the cube." In: *Combinatorica* 32.1 (2012), pp. 35–53.

[Buh+13] H. Buhrman et al. "The non-adaptive query complexity of testing k-parities." In: *Chicago Journal of Theoretical Computer Science* 2013 (2013).

[CG04] H. Chockler and D. Gutfreund. "A lower bound for testing juntas." In: *Information Processing Letters* 90.6 (2004), pp. 301–305.

[CS13a] D. Chakrabarty and C. Seshadhri. "A $o(n)$ monotonicity tester for boolean functions over the hypercube." In: *Proceedings of the 45th ACM Symposium on Theory of Computing*. 2013, pp. 411–418.

[CS13b] D. Chakrabarty and C. Seshadhri. "Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids." In: *Proceedings of the 45th ACM Symposium on Theory of Computing*. 2013, pp. 419–428.

[CS14] D. Chakrabarty and C. Seshadhri. "An Optimal Lower Bound for Monotonicity Testing over Hypergrids." In: *Theory of Computing* 10.17 (2014), pp. 453–464.

[CS16] D. Chakrabarty and C. Seshadhri. "An $\tilde{O}(n)$ non-adaptive tester for unateness." arXiv:1608.06980. 2016.

[CST14] X. Chen, R.A. Servedio, and L.-Y. Tan. "New Algorithms and Lower Bounds for Monotonicity Testing." In: *Proceedings of the IEEE 55th Annual Symposium on Foundations of Computer Science*. 2014, pp. 286–295.

[CWX17a] X. Chen, E. Waingarten, and J. Xie. "Beyond Talagrand Functions: New Lower Bounds for Testing Monotonicity and Unateness." In: *Proceedings of the 49th ACM Symposium on the Theory of Computing (STOC '2017)*. 2017.

[CWX17b] X. Chen, E. Waingarten, and J. Xie. "Boolean Unateness Testing with $\widetilde{O}(n^{3/4})$ Adaptive Queries." In: *The 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2017)*. 2017.

[CX15] X. Chen and J. Xie. "Tight Bounds for the Distribution-Free Testing of Monotone Conjunctions." Manuscript. 2015.

[Che+15] X. Chen et al. “Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries.” In: *Proceedings of the 47th ACM Symposium on Theory of Computing*. 2015, pp. 519–528.

[Che+17] X. Chen et al. “Settling the Query Complexity of Non-adaptive Junta Testing.” In: *Proceedings of the 32nd Conference on Computational Complexity (CCC ’2017)*. 2017.

[Che+18] X. Chen et al. “Distribution-free Junta Testing.” In: *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC ’2018)*. 2018.

[DR11] E. Dolev and D. Ron. “Distribution-Free Testing for Monomials with a Sublinear Number of Queries.” In: *Theory of Computing* 7.1 (2011), pp. 155–176.

[Dod+99] Y. Dodis et al. “Improved Testing Algorithms for Monotonocity.” In: *Proceedings of the 3rd International Workshop on Randomization and Approximation Techniques in Computer Science*. 1999, pp. 97–108.

[Fis+02] E. Fischer et al. “Monotonicity testing over general poset domains.” In: *Proceedings of the 34th Annual ACM Symposium on the Theory of Computing*. 2002, pp. 474–483.

[Fis+04] E. Fischer et al. “Testing juntas.” In: *Journal of Computer & System Sciences* 68 (2004), pp. 753–787.

[Fis04] E. Fischer. “On the strength of comparisons in property testing.” In: *Information and Computation* 189.1 (2004), pp. 107–116.

[GGR98] O. Goldreich, S. Goldwasser, and D. Ron. “Property testing and its connection to learning and approximation.” In: *Journal of the ACM* 45 (1998), pp. 653–750.

[GS09] D. Glasner and R. A. Servedio. “Distribution-Free Testing Lower Bound for Basic Boolean Functions.” In: *Theory of Computing* 5.1 (2009), pp. 191–216.

[Gol+00] O. Goldreich et al. “Testing Monotonicity.” In: *Combinatorica* 20.3 (2000), pp. 301–337.

[HK04] S. Halevy and E. Kushilevitz. “Distribution-Free Connectivity Testing for Sparse Graphs.” In: *Algorithmica* 51.1 (2004), pp. 24–48.

[HK05] S. Halevy and E. Kushilevitz. “A Lower Bound for Distribution-Free Monotonicity Testing.” In: *Proceedings of the Ninth International Workshop on Randomization and Computation*. 2005, pp. 330–341.

[HK07] S. Halevy and E. Kushilevitz. "Distribution-Free Property Testing." In: *SIAM J. Comput.* 37.4 (2007), pp. 1107–1138.

[HK08] S. Halevy and E. Kushilevitz. "Testing monotonicity over graph products." In: *Random Structures and Algorithms* 33.1 (2008), pp. 44–67.

[KMS15] S. Khot, D. Minzer, and M. Safra. "On Monotonicity Testing and Boolean Isoperimetric type Theorems." In: *Proceedings of the 56th Annual Symposium on Foundations of Computer Science*. 2015, pp. 52–58.

[KS16] S. Khot and I. Shinkar. "An $\tilde{O}(n)$ queries adaptive tester for unateness." In: *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*. 2016, 37:1–37:7.

[MO03] E. Mossel and R. O'Donnell. "On the noise sensitivity of monotone functions." In: *Random Structures and Algorithms* 23.3 (2003), pp. 333–350.

[Mat+09] K. Matulef et al. "Testing$\pm$1-weight halfspace." In: *APPROX-RANDOM*. 2009, pp. 646–657.

[Mat+10] K. Matulef et al. "Testing halfspaces." In: *SIAM Journal on Comput.* 39.5 (2010), pp. 2004–2047.

[RM99] R. Raz and P. McKenzie. "Separation of the Monotone NC Hierarchy." In: *Combinatorica* 19.3 (1999), pp. 403–435.

[RS09] R. Rubinfeld and R.A. Servedio. "Testing monotone high-dimensional distributions." In: *Random Structures and Algorithms* 34.1 (2009), pp. 24–44.

[RS96] R. Rubinfeld and M. Sudan. "Robust characterizations of polynomials with applications to program testing." In: *SIAM J. on Comput.* 25 (1996), pp. 252–271.

[Ron+12] D. Ron et al. "Approximating the Influence of Monotone Boolean Functions in $O(\sqrt{n})$ Query Complexity." In: *ACM Transactions on Computation Theory* 4.4 (2012), pp. 1–12.

[STW15] R.A. Servedio, L.-Y. Tan, and J. Wright. "Adaptivity helps for testing juntas." In: *Proceedings of the 30th IEEE Conference on Computational Complexity*. 2015, pp. 264–279.

[Tal96] M. Talagrand. "How Much Are Increasing Sets Positively Correlated?" In: *Combinatorica* 16.2 (1996), pp. 243–258.

[Val84] L. G. Valiant. “A theory of the learnable.” In: *Proc. 16th Annual ACM Symposium on Theory of Computing (STOC)*. ACM Press, 1984, pp. 436–445.