

Relative-Error Testing of Conjunctions and Decision Lists

Xi Chen ✉ 

Columbia University, New York, NY, USA

William Pires ✉ 

Columbia University, New York, NY, USA

Toniann Pitassi ✉ 

Columbia University, New York, NY, USA

Rocco A. Servedio ✉ 

Columbia University, New York, NY, USA

Abstract

We study the *relative-error* property testing model for Boolean functions that was recently introduced in the work of [10]. In relative-error testing, the testing algorithm gets uniform random satisfying assignments as well as black-box queries to f , and it must accept f with high probability whenever f has the property that is being tested and reject any f that is *relative-error* far from having the property. Here the relative-error distance from f to a function g is measured with respect to $|f^{-1}(1)|$ rather than with respect to the entire domain size 2^n as in the Hamming distance measure that is used in the standard model; thus, unlike the standard model, relative-error testing allows us to study the testability of *sparse* Boolean functions that have few satisfying assignments. It was shown in [10] that relative-error testing is at least as difficult as standard-model property testing, but for many natural and important Boolean function classes the precise relationship between the two notions is unknown.

In this paper we consider the well-studied and fundamental properties of being a *conjunction* and being a *decision list*. In the relative-error setting, we give an efficient one-sided error tester for conjunctions with running time and query complexity $O(1/\varepsilon)$.

Secondly, we give a two-sided relative-error $\tilde{O}(1/\varepsilon)$ tester for decision lists, matching the query complexity of the state-of-the-art algorithm in the standard model [4, 19].

2012 ACM Subject Classification Theory of computation $\rightarrow$ Streaming, sublinear and near linear time algorithms

Keywords and phrases Property Testing, Relative Error

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.52

Category Track A: Algorithms, Complexity and Games

Related Version Full version: <https://arxiv.org/abs/2504.08987>

Funding Xi Chen: Supported by NSF awards CCF-2106429 and CCF-2107187.

Toniann Pitassi: Supported by NSF award CCF-2212136.

Rocco A. Servedio: Supported by NSF awards CCF-2106429 and CCF-2211238.

1 Introduction

Background. Over the past few decades, property testing of Boolean functions has grown into a rich and active research area (see e.g. the extensive treatment in [23, 2]). Emerging from the study of program testing and self-correction [3, 21], the field has developed strong connections to many other topics in theoretical computer science including complexity theory, Boolean function analysis, and computational learning theory, see e.g. [24, 34, 36].



© Xi Chen, William Pires, Toniann Pitassi, and Rocco A. Servedio;
licensed under Creative Commons License CC-BY 4.0

52nd International Colloquium on Automata, Languages, and Programming (ICALP 2025).

Editors: Keren Censor-Hillel, Fabrizio Grandoni, Joël Ouaknine, and Gabriele Puppis

Article No. 52; pp. 52:1–52:18



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



In the “standard” model of Boolean function property testing, the algorithm for testing an unknown Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ has black-box oracle (also known as “membership query” and will be referred to as $\text{MQ}(f)$ in this paper) access as its only mode of access to f . The goal of a tester in this standard model is to distinguish between the two alternatives that (a) f has some property of interest, versus (b) f is “far” (in the sense of disagreeing on at least an ε -fraction of all the 2^n inputs) from every function with the property. A wide range of properties (or equivalently, classes of Boolean functions) have been studied from this perspective, including but not limited to monotone Boolean functions [25, 20, 6, 7, 14, 11, 1, 16, 30], unate Boolean functions [31, 8, 32, 15, 16, 17, 15], literals [35], conjunctions [35, 27], decision lists [19, 4], linear threshold functions [33], s -term monotone DNF [35, 19, 9, 4], s -term DNF, size- s decision trees, size- s branching programs, size- s Boolean formulas, size- s Boolean circuits, functions with Fourier degree at most d , s -sparse polynomials over $\mathbb{F}_2$ [19, 9, 4], and many others (see e.g. Figure 1 of [4]).

As the field of Boolean function property testing has grown more mature, an interest has developed in considering alternative variants of the “standard” property testing model discussed above. One early variant was the *distribution-free* property testing model [26, 28], inspired by Valiant’s distribution-free PAC learning model [37]. In this model the distance between two functions f and g is measured according to $\Pr_{\mathbf{x} \sim \mathcal{D}}[f(\mathbf{x}) \neq g(\mathbf{x})]$, where $\mathcal{D}$ is an unknown and arbitrary distribution over the domain $\{0,1\}^n$. In addition to making black-box oracle queries, distribution-free testing algorithms can also draw i.i.d. samples from the unknown distribution $\mathcal{D}$. The motivation behind this model was to align property testing more closely with the influential distribution-free PAC learning model and to capture scenarios of interest in which there is an underlying non-uniform distribution over the domain. Unfortunately, many of the results that have been established for the distribution-free PAC model are negative in nature [29, 22, 18, 13, 12], often showing that almost as many oracle calls are needed for distribution-free testing a class $\mathcal{C}$ as would be required for PAC learning $\mathcal{C}$. (As shown in [26], the PAC learning complexity of any class $\mathcal{C}$ gives an easy upper bound on its distribution-free testing complexity.)

Relative-error testing. Very recent work [10] introduced a new model of Boolean function property testing, called *relative-error testing*. The motivation for the model comes from the observation that the standard testing framework is not well suited for testing *sparse* Boolean functions, i.e., functions $f : \{0,1\}^n \rightarrow \{0,1\}$ that have $|f^{-1}(1)| = p2^n$ where p is very small¹ – perhaps as small as $1/n^{\omega(1)}$ or even $2^{-\Theta(n)}$ – simply because any such function will be p -close to the constant-0 function. (This is closely analogous to the well-known fact that the standard dense graph property testing model, where error ε corresponds to adding or deleting εn^2 potential edges in an n -vertex graph, is not well suited to testing sparse graphs, because any sparse graph trivially has small distance from the empty graph on n vertices.)

Motivated by the above considerations, the relative-error property testing model of [10] defines the distance between the unknown $f : \{0,1\}^n \rightarrow \{0,1\}$ and another function g to be

$$\text{rel-dist}(f, g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|};$$

so relative distance is measured *at the scale of* $|f^{-1}(1)|$ rather than at the “absolute” scale of $2^n = |\{0,1\}^n|$. In addition to the black-box oracle $\text{MQ}(f)$, the model also allows the

¹ Such functions can be of significant interest for practical as well as theoretical reasons; since Boolean functions correspond to classification rules, a sparse Boolean function corresponds to a classification rule which outputs 1 only on positive examples of some rare but potentially desirable phenomenon.

testing algorithm to sample i.i.d. uniform random satisfying assignments of f via a sampling oracle $\text{Samp}(f)$: each call to $\text{Samp}(f)$ returns a uniformly random sample from $f^{-1}(1)$ (see Section 2.2 and Remark 4 for a detailed description of the model and comparison with the distribution-free testing model).

The initial work [10] established some relations between the query complexity of standard-model testing and relative-error testing. In more detail, letting $q_{\text{std}}(\varepsilon, n)$ and $q_{\text{rel}}(\varepsilon, n)$ denote the number of oracle calls needed to test a property $\mathcal{C}$ in the standard model and the relative-error model, respectively, they showed (as Facts 8 and 9 in [10]) that

$$q_{\text{std}}(\varepsilon, n) \lesssim q_{\text{rel}}(\varepsilon, n)/\varepsilon \quad \text{and} \quad q_{\text{rel}}(\varepsilon, n) \lesssim q_{\text{std}}(p\varepsilon, n). \quad (1)$$

The first inequality shows that standard-model testing is never significantly harder than relative-error testing, but since the value of $p = |f^{-1}(1)|/2^n$ can be extremely small, the second inequality leaves open the possibility that relative-error testing can be much harder than standard-model testing. Indeed, [10] gave an example of an property which is constant-query testable in the standard model but requires *exponentially* many queries for relative-error testing. However, the class of functions exhibiting this separation² is arguably contrived.

This leads to the following question, which is the motivation for our work:

What is the complexity of relative-error testing
for *natural and well-studied* properties of Boolean functions?

Progress in this direction was made by [10], which give a relative-error tester for monotonicity which makes at most quadratically more queries than the best standard-model tester.

The problems we consider. We attack the above broad question by studying the relative-error testability of two basic and natural classes of Boolean functions, namely *conjunctions* and *decision lists*.³

Conjunctions and decision lists have been thoroughly studied in both the standard model of Boolean function property testing and the distribution-free model [35, 19, 22, 18, 27, 4, 12]. For the standard model, unlike monotonicity, both problems belong to the regime of “efficiently testable” properties, where the number of queries needed depends on ε only but not the dimension n :

[35, 27] gave $O(1/\varepsilon)$ -query algorithms for testing conjunctions and [4] gave an $\tilde{O}(1/\varepsilon)$ -query algorithm for testing decision lists (improving on an earlier $\tilde{O}(1/\varepsilon^2)$ -query algorithm of [19]). In contrast, distribution-free testing is provably much more difficult: improving on [22], [18] gave an $\tilde{\Omega}(n^{1/3})$ lower bound for conjunctions (and a matching upper bound); [12] gave an $\tilde{\Omega}(n^{1/2})$ lower bound for decision lists (and an $\tilde{O}(n^{11/12})$ upper bound).

One intuitive reason for this disparity between the standard model and the distribution-free model is that, *under the uniform distribution*, (i) every conjunction is either ε -close to the constant-0 function or else is a $\log(1/\varepsilon)$ -junta, and (ii) every decision list is ε -close to a $\log(1/\varepsilon)$ -junta.⁴ In the distribution-free setting, though, these facts fail, and notably,

² The class consists of all functions g for which $|g^{-1}(1)|$ is a multiple of $2^{2n/3}$ (see Appendix A of [10]).

³ Recall that a *conjunction* is an AND of literals $\ell_1 \wedge \dots \wedge \ell_k$, where each ℓ_i is either some Boolean variable x_j or its negation $\bar{x}_j$. A *decision list* is a sequence of nested “if (condition) holds, then output (bit), else ...” rules, where each (condition) is a Boolean literal (see the beginning of Section 4 for the formal definition of decision lists).

⁴ A k -junta is a function that only depends on at most k of its variables.

they fail in the relative-error setting as well. Given this, it is *a priori* unclear whether one should expect the relative-error testability of conjunctions and decision lists to be more like the standard setting, or more like the distribution-free setting. Resolving this is one of the motivations of the present work.

1.1 Our results and techniques

Results. As our main results, we show that both classes of conjunctions and decision lists are efficiently testable in the relative-error model; indeed we give relative-error testing algorithms that have the same query complexity as the best standard-model testing algorithms.

For conjunctions we prove the following theorem:

► **Theorem 1** (Relative-error testing of conjunctions). *There is a one-sided non-adaptive algorithm which is an ε -relative-error tester for conjunctions on $\{0,1\}^n$. The algorithm makes $O(1/\varepsilon)$ calls to the sampling oracle $\text{Samp}(f)$ and the membership query oracle $\text{MQ}(f)$.*

Readers who are familiar with the literature on testing conjunctions in the standard setting may have noticed that the algorithm above is one-sided and uses $O(1/\varepsilon)$ queries, while only two-sided algorithms are known for testing conjunctions in the standard model with $O(1/\varepsilon)$ queries [35] and the current best one-sided algorithm needs $\tilde{O}(1/\varepsilon)$ queries [27]. It was posed as an open problem in [4] whether there exists a one-sided, $O(1/\varepsilon)$ -query tester for conjunctions in the standard model. We resolve this question by giving such a tester as a corollary of Theorem 1:

► **Theorem 2.** *There is a one-sided, adaptive algorithm which is an ε -error tester for conjunctions over $\{0,1\}^n$ in the standard setting. The algorithm makes $O(1/\varepsilon)$ calls to $\text{MQ}(f)$.*

Note that combining the first part of Equation (1) and Theorem 1 will only lead to a one-sided upper bound of $O(1/\varepsilon^2)$ for the standard model, where the extra $1/\varepsilon$ factor comes from the natural fact that whenever f has density at least ε , it takes $1/\varepsilon$ uniformly random samples to simulate one sample from $f^{-1}(1)$. To prove Theorem 2, we obtain a more efficient reduction from relative-error testing to standard testing which, informally, states

$$q_{\text{std}}(\varepsilon, n) \lesssim 1/\varepsilon + q_{\text{rel}}(\varepsilon/2, n), \quad (2)$$

from which Theorem 2 follows. The reduction is presented in Section 2.3.

Our second, and more involved, main result obtains an efficient relative-error testing algorithm for decision lists:

► **Theorem 3** (Relative-error testing of decision lists). *There is an algorithm which is an ε -relative-error tester for decision lists on $\{0,1\}^n$. The algorithm makes $\tilde{O}(1/\varepsilon)$ calls to $\text{Samp}(f)$ and $\text{MQ}(f)$.*

We remark that the algorithm of Theorem 3 crucially uses the relative-error testing algorithm for conjunctions, i.e. Theorem 1, as a subroutine; this is explained more below and a more detailed overview of the algorithm can be found in Section 4.2.

We also note that the algorithm of Theorem 3 is adaptive. However, an easy variant of our algorithm and analysis yields a non-adaptive ε -relative-error tester for decision lists that makes $O(1/\varepsilon)$ calls to $\text{Samp}(f)$ and $\tilde{O}(1/\varepsilon^2)$ calls to $\text{MQ}(f)$ (see the full version).

Finally, we give a lower bound of $\Omega(1/\varepsilon)$ for any two-sided, adaptive algorithm for the relative-error testing of both conjunctions and decision lists, thus showing that Theorem 1 is optimal and Theorem 3 is nearly optimal (see the full version).

Techniques. We give a brief and high-level overview of our techniques, referring the reader to Section 3.2 and Section 4.2 for more details. Our relative-error testing algorithm for conjunctions begins with a simple reduction to the problem of relative-error testing whether an unknown f is an anti-monotone conjunction⁵. We solve the relative-error anti-monotone conjunction testing problem with an algorithm which, roughly speaking, first checks whether $f^{-1}(1)$ is close to a linear subspace (as would be the case if f were an anti-monotone conjunction), and then checks whether f is close to anti-monotone (in a suitable sense to meet our requirements). This is similar, at a high level, to the approach from [35], but as discussed in Section 3.2 there are various technical differences; in particular we are able to give a simpler analysis, even for the more general relative-error setting, which achieves one-sided rather than two-sided error.

For decision lists, our starting point is the simple observation that when f is a sparse decision list then f can be written as $C \wedge L$ where C is a conjunction and L is a decision list (intuitively, since f is sparse, a large prefix of the list of output bits of the decision list must all be 0, which corresponds to a conjunction C); this decomposition plays an important role for us. Another simple but useful observation is that all of the variables in the conjunction C must be “unanimous,” in the sense that each one of them must always take the same value across all of the positive examples. Our algorithm draws an initial sample of positive examples from $\text{Samp}(f)$ and then works with a restriction of f which fixes the variables which were seen to be “unanimous” in that initial sample. Intuitively, if f is a decision list, then it should be the case that the resulting restricted function is close to a *non-sparse* decision list, so we can check that the restricted function is non-sparse, and, if the check passes, apply a standard-model decision list tester to the restricted function.

This gives some intuition for how to design an algorithm that accepts in the yes-case, but the fact that we must reject all functions that are relative-error far from decision lists leads to significant additional complications. To deal with this, we need to incorporate additional checks into the test, and we give an analysis showing that if a function f passes all of these checks then it must be close (in a relative-error sense) to a function of the form $C' \wedge L'$ where C' is relative-error close to a conjunction and L' is relative-error close to a decision list; this lets us conclude that f is relative-error close to a decision list, as desired. These checks involve applying our relative-error conjunction testing algorithm (which we show to have useful robustness properties against being run with a slightly imperfect sampling oracle – this robustness is crucial for our analysis); we refer the reader to Section 4.2 for more details.

Looking ahead, while our results resolve the relative-error testability of conjunctions and decision lists, there are many open questions remaining about how relative-error testing compares against standard-error testing for other natural and well-studied Boolean function classes, such as s -term DNF formulas, k -juntas, subclasses of k -juntas, unate functions, and beyond. All of these classes have been intensively studied in the standard testing model, see e.g. [19, 9, 31, 8, 32, 15, 16, 17, 15, 4]; it is an interesting direction for future work to establish either efficient relative-error testing algorithms, or relative-error testing lower bounds, for these classes.

Organization. Section 2 contains preliminary notation and definitions, and the general reduction from relative-error testing to standard testing is presented in Section 2.3. We give an outline of the proof of Theorem 1 in Section 3, and likewise for Theorem 3 in Section 4. Because of space constraints full proofs are deferred to the full version in the appendix.

⁵ A conjunction is an anti-monotone conjunction if it only contains negated literals.

2 Notation, Definitions, and Preliminaries

2.1 Notation

For $x, y \in \{0, 1\}^n$, we write $y \preceq x$ to indicate that $y_i \leq x_i$ for all $i \in [n]$. We write $x \oplus y$ to denote the string in $\{0, 1\}^n$ whose i -th coordinate is the XOR of x_i and y_i . Given $S \subseteq [n]$ and $z \in \{0, 1\}^n$, we denote by z_S the string in $\{0, 1\}^S$ that agrees with z on every coordinate in S .

We use standard notation for restrictions of functions. For $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $u \in \{0, 1\}^S$ for some $S \subseteq [n]$, the function $f|_u : \{0, 1\}^{[n] \setminus S} \rightarrow \{0, 1\}$ is defined as the function

$$f|_u(x) = f(z), \quad \text{where} \quad z_i = \begin{cases} u_i & \text{if } i \in S, \\ x_i & \text{if } i \in [n] \setminus S. \end{cases}$$

We write $\mathcal{U}_S$ to denote the uniform distribution over a finite set S , and we write $\mathcal{U}_n$ to denote the uniform distribution over $\{0, 1\}^n$; we usually skip the subscript n when it is clear from context.

When we refer to vector spaces/subspaces, we will always be talking about the vector space $\mathbb{F}_2^n$, and we will frequently identify $\{0, 1\}^n$ with the vector space $\mathbb{F}_2^n$.

Finally, we write DL to denote the class of all decision lists over $\{0, 1\}^n$, and Conj to denote the class of all conjunctions over $\{0, 1\}^n$.

2.2 Background on property testing and relative-error testing

We recall the basics of the standard property testing model. In the *standard* testing model for a class $\mathcal{C}$ of n -variable Boolean functions, the testing algorithm is given oracle access $\text{MQ}(f)$ to an unknown and arbitrary function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. The algorithm must output “yes” with high probability (say at least $2/3$; this can be amplified using standard techniques) if $f \in \mathcal{C}$, and must output “no” with high probability (again, say at least $2/3$) if $\text{dist}(f, \mathcal{C}) \geq \varepsilon$, where

$$\text{dist}(f, \mathcal{C}) := \min_{g \in \mathcal{C}} \text{dist}(f, g) \quad \text{and} \quad \text{dist}(f, g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{2^n}.$$

As defined in [10], a *relative-error* testing algorithm for $\mathcal{C}$ has oracle access to $\text{MQ}(f)$ and also has access to a $\text{Samp}(f)$ oracle which, when called, returns a uniform random element $x \sim f^{-1}(1)$. A relative-error testing algorithm for $\mathcal{C}$ must output “yes” with high probability (say at least $2/3$; again this can be easily amplified) if $f \in \mathcal{C}$ and must output “no” with high probability (again, say at least $2/3$) if $\text{rel-dist}(f, \mathcal{C}) \geq \varepsilon$, where

$$\text{rel-dist}(f, \mathcal{C}) := \min_{g \in \mathcal{C}} \text{rel-dist}(f, g) \quad \text{and} \quad \text{rel-dist}(f, g) := \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|}.$$

Finally, recall that a property testing algorithm for a class of functions $\mathcal{C}$ (in either the standard model or the relative-error model) is said to be *one-sided* if it returns “yes” with probability 1 when the unknown function belongs to $\mathcal{C}$.

► **Remark 4 (Relative-error testing versus distribution-free testing).** Like the distribution-free model, the relative-error model gives the testing algorithm access to both black-box oracle calls to f and i.i.d. samples from a distribution $\mathcal{D}$. In distribution-free testing $\mathcal{D}$ is unknown and arbitrary, whereas in relative-error testing $\mathcal{D}$ is unknown but is guaranteed to be the uniform distribution over $f^{-1}(1)$. However, we remark that relative-error testing is

not a special case of distribution-free testing, because the distance between functions is measured differently between the two settings. In the distribution-free model, the distance $\Pr_{\mathbf{x} \sim \mathcal{D}}[f(\mathbf{x}) \neq g(\mathbf{x})]$ between two functions f, g is measured vis-a-vis the distribution $\mathcal{D}$ that the testing algorithm can sample from. In contrast, in the relative-error setting the distribution $\mathcal{D}$ that the tester can sample from is $\mathcal{D} = \mathcal{U}_{f^{-1}(1)}$, but $\text{rel-dist}(f, g)$ is *not* equal to $\Pr_{\mathbf{x} \sim \mathcal{U}_{f^{-1}(1)}}[f(\mathbf{x}) \neq g(\mathbf{x})]$; indeed, a function g can have large relative distance from f because g disagrees with f on many points outside of $f^{-1}(1)$ (which are points that have zero weight under the distribution $\mathcal{D}$).

We will need the following simple “approximate triangle inequality” for relative distance, proved in the full version:

► **Lemma 5** (Approximate triangle inequality for relative distance). *Let $f, g, h : \{0, 1\}^n \rightarrow \{0, 1\}$ be such that $\text{rel-dist}(f, g) \leq \varepsilon$ and $\text{rel-dist}(g, h) \leq \varepsilon'$. Then $\text{rel-dist}(f, h) \leq \varepsilon + (1 + \varepsilon)\varepsilon'$.*

Finally, we recall two results for testing decision lists in the standard model that we will use:

► **Theorem 6** ([4]). *There is an adaptive algorithm $\mathcal{A}$ for testing decision lists in the standard model with query complexity $\tilde{O}(1/\varepsilon)$. In particular, if f is a decision list then $\mathcal{A}$ accepts with probability at least $19/20$; if $\text{dist}(f, \text{DL}) \geq \varepsilon$ then $\mathcal{A}$ rejects with probability at least $19/20$.*

► **Theorem 7** ([19]). *There is a non-adaptive algorithm $\mathcal{B}$ for testing decision lists in the standard model with query complexity $\tilde{O}(1/\varepsilon^2)$. In particular, if f is a decision list then $\mathcal{B}$ accepts with probability at least $19/20$; if $\text{dist}(f, \text{DL}) \geq \varepsilon$ then $\mathcal{B}$ rejects with probability at least $19/20$.*

2.3 From relative-error testing to standard testing

In this section, we use our relative-error tester for conjunctions (see Theorem 1) to give an optimal one-sided tester for this class in the standard model. Fact 8 of [10] gives a way of transforming any relative-error tester for a property $\mathcal{C}$ into a tester in the standard model for $\mathcal{C}$. However, this transformation incurs a multiplicative $1/\varepsilon$ blowup in the number of oracle calls made by the tester. We show that if the all-0 function is in $\mathcal{C}$ and the number of oracle calls made by the relative-error tester grows at least linearly in $1/\varepsilon$, we can remove this blowup.

► **Lemma 8.** *Let $\mathcal{C}$ be a class of Boolean functions over $\{0, 1\}^n$ that contains the all-0 function. Suppose that there is a relative-error ε -testing algorithm $\mathcal{A}$ for $\mathcal{C}$ that makes $q(\varepsilon, n)$ many calls to $\text{MQ}(f)$ and $\text{Samp}(f)$ such that $q(\varepsilon/\delta, n) \leq \delta q(\varepsilon, n)$ for every $0 < \delta < 1$. Then there is an adaptive standard-model ε -testing algorithm $\mathcal{B}$ for $\mathcal{C}$ which makes at most $O(1/\varepsilon + q(\varepsilon/2, n))$ calls to $\text{MQ}(f)$. Furthermore, if $\mathcal{A}$ is one-sided, so is $\mathcal{B}$.*

Before entering into the formal proof, we give some intuition. To prove Lemma 8, we want to run $\mathcal{A}$ on f to see if f is ε -far from the class. For this, we will need uniform samples from $f^{-1}(1)$; these can be obtained by randomly querying points of the hypercube to find 1s of f . However if $\Pr_{\mathbf{z} \sim \mathcal{U}_n}[f(\mathbf{z}) = 1] =: p_f$ is small, then we would need roughly $1/p_f$ queries per sample. The following key lemma shows that if f is sparse and ε -far from $\mathcal{C}$, then its relative distance to $\mathcal{C}$ is at least ε/p_f . This means we can simulate $\mathcal{A}$ on f with the distance parameter for $\mathcal{A}$ set to a higher value ε . Hence while getting samples is hard, this is canceled out by the fact we need less of them.

■ **Algorithm 1** Standard-model testing algorithm for $\mathcal{C}$ using a relative-error tester $\mathcal{A}$. Here both c_1 and c_2 are sufficiently large absolute constants.

Input: Oracle access to $\text{MQ}(f)$ and an accuracy parameter $\varepsilon > 0$.

Output: “Reject” or “Accept.”

Phase 1: Estimating $|f^{-1}(1)|$.

- 1: Draw c_1/ε many samples $\mathbf{z} \sim \mathcal{U}_n$ and let $\mathbf{S}$ be the set of samples.
- 2: For each $\mathbf{z} \in \mathbf{S}$ query $f(\mathbf{z})$ and let $\hat{p}$ be the fraction of $\mathbf{z} \in \mathbf{S}$ with $f(\mathbf{z}) = 1$.
- 3: If $\hat{p} \leq \varepsilon/2$, then halt and accept f .

Phase 2: Getting samples

- 4: Draw $c_2 \cdot q(\varepsilon/2, n)$ many samples $\mathbf{z} \sim \mathcal{U}_n$ and let $\mathbf{G}$ be set of samples with $f(\mathbf{z}) = 1$.
- 5: If $|\mathbf{G}| \leq q(\varepsilon/2\hat{p}, n)$, then halt and accept f .

Phase 3: Running the relative error tester

- 6: Run $\mathcal{A}$ using the samples in $\mathbf{G}$ to simulate $\text{Samp}(f)$ and distance parameter $\varepsilon/2\hat{p}$.
 - 7: Output what $\mathcal{A}$ outputs.
-

► **Lemma 9.** If $\text{dist}(f, \mathcal{C}) > \varepsilon$, then $\text{rel-dist}(f, \mathcal{C}) > \varepsilon/p_f$ where $p_f := |f^{-1}(1)|/2^n$.

Proof. Assume for a contradiction that $\text{rel-dist}(f, \mathcal{C}) \leq \varepsilon/p_f$ for some $g \in \mathcal{C}$. Then we have that:

$$\frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|} = \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{p_f 2^n} \leq \frac{\varepsilon}{p_f},$$

which implies that $|f^{-1}(1) \triangle g^{-1}(1)| \leq \varepsilon 2^n$ and thus, $\text{dist}(f, g) \leq \varepsilon$, a contradiction. ◀

We now prove Lemma 8.

Proof of Lemma 8. Our goal will be to show that Algorithm 1 is a standard testing algorithm for $\mathcal{C}$. Without loss of generality, we assume that $\mathcal{A}$ is correct with high enough probability, that is, if $\text{rel-dist}(f, \mathcal{C}) \geq \varepsilon$ the algorithm rejects with probability at least 0.9. Similarly, if $f \in \mathcal{C}$, we assume that $\mathcal{A}$ accepts with probability at least 0.9.

First, consider the case where $f \in \mathcal{C}$. Note that f can only be rejected if $\mathcal{A}$ rejects f . Since $\mathbf{G}$ is a set of uniformly random samples from $f^{-1}(1)$, $\mathcal{A}$ must accept with probability at least 0.9. And if $\mathcal{A}$ is one-sided, then f is always accepted, meaning that Algorithm 1 is also one-sided.

Now, assume that f is ε -far from any function in $\mathcal{C}$. Because the all-0 function is in $\mathcal{C}$ we must have $p_f := |f^{-1}(1)|/2^n \geq \varepsilon$. So on line 2, we have by a Chernoff bound that

$$\Pr[|\hat{p} - p_f| > 0.05p_f] \leq 2e^{-\frac{(0.05)^2 \varepsilon}{2+0.05} c_1/\varepsilon} \leq 0.05,$$

by picking a suitably large constant c_1 . Below we assume that $\hat{p} \in (1 \pm 0.05)p_f$.

Since $p_f \geq \varepsilon$, this implies $\widehat{p} \geq \varepsilon/2$. So, the algorithm doesn't accept on line 3. We also have that $\Pr_{\mathbf{z} \sim \mathcal{U}_n}[f(\mathbf{z}) = 1] \geq 0.9\widehat{p}$. Hence, the expected size of $|G|$ is at least $0.9c_2\widehat{p} \cdot q(\varepsilon/2, n)$. By our assumption on $q(\cdot, \cdot)$ this is at least $0.9c_2 \cdot q(\varepsilon/2\widehat{p}, n)$. Again, by a Chernoff bound and by picking c_2 to be a sufficiently large constant, we have that $|G| \leq q(\varepsilon/2\widehat{p}, n)$ with probability at most 0.05.

It remains to argue that $\mathcal{A}$ rejects with high probability on line 6. By Lemma 9, we have that $\text{rel-dist}(f, \mathcal{C}) \geq \varepsilon/p_f$. Using $\widehat{p} \geq 0.95p_f$, we have that $\text{rel-dist}(f, \mathcal{C}) \geq \varepsilon/2\widehat{p}$. Since we simulate $\mathcal{A}$ on line 6 using uniform samples from $f^{-1}(1)$, it must accept f with probability at most 0.1. As a result, the probability the algorithm accepts f is at most $0.05 + 0.05 + 0.1 \leq 0.2$.

Line 2 uses $O(1/\varepsilon)$ many queries, line 4 uses $O(q(\varepsilon/2, n))$ many queries, and simulating $\mathcal{A}$ on line 6 uses $q(\varepsilon/2\widehat{p}, n) = O(q(\varepsilon/2, n))$ queries. So, the overall query complexity is $O(1/\varepsilon + q(\varepsilon/2, n))$. ◀

Using the above, we can obtain a one-sided tester for conjunctions in the standard setting from our one-sided relative-error conjunction tester. The result of [5] implies an $\Omega(1/\varepsilon)$ lower bound for this problem, hence this result is tight.

Proof of Theorem 2. Our relative-error tester for conjunctions in Theorem 1 makes $O(1/\varepsilon)$ calls to $\text{Samp}(f)$ and $\text{MQ}(f)$ and is one-sided. As the all-0 function is a conjunction, the theorem follows by Lemma 8. ◀

3 A relative-error testing algorithm for conjunctions

In this section we prove Theorem 1. At the end of the section, we show that the main algorithm for conjunctions works even when its access to the sampling oracle is not perfect; this will be important when the algorithm is used as a subroutine to test decision lists in the next section.

► **Remark 10.** In the rest of this section, we give a relative-error tester for the class of *anti-monotone* conjunctions. That is, for conjunctions of *negated* variables. Using an observation from [35] (see also [27]), this implies a relative-error tester for general conjunctions. Specifically, a slightly modified version of Observation 3 from Parnas et al. [35] states that for any conjunction f (not necessarily anti-monotone) and for any $y \in f^{-1}(1)$, the function

$$f_y(x) := f(x \oplus y)$$

is an anti-monotone conjunction. Furthermore, it is easy to see that if f is ε -relative-error far from every conjunction, then f_y must be ε -relative-error far from every anti-monotone conjunction.

So to get a relative-error tester for general conjunctions, we simply draw a single $y \sim f^{-1}(1)$ and then run the tester for anti-monotone conjunctions (Algorithm 2) on f_y , where $\text{MQ}(f_y)$ and $\text{Samp}(f_y)$ can be simulated using $\text{MQ}(f)$ and $\text{Samp}(f)$, respectively; we refer to this algorithm as CONJ-TEST.

3.1 Preliminaries

► **Definition 11 (Subspaces).** A set $W \subseteq \{0, 1\}^n$ is an affine subspace over F_2^n if and only if W is the set of solutions to a system of linear equations over F_2^n . That is, iff W is the set of solutions to $Ax = b$, for some $b \in \{0, 1\}^n$. W is a linear subspace over F_2^n iff it is the set of solutions to $Ax = b$ where b is the all-0 vector. A well-known equivalent characterization states that W is an affine subspace if and only if for all $x, y, z \in W$, $x \oplus y \oplus z \in W$. And for linear subspaces, we have that W is linear if and only if for all $x, y \in W$, $x \oplus y \in W$.

We will need the following fact about the intersection of linear subspaces.

► **Fact 12.** *Let H, H' be two linear subspaces of $\{0, 1\}^n$ with $H \not\subseteq H'$. Then $\frac{|H \cap H'|}{|H|} \leq \frac{1}{2}$.*

► **Definition 13** (Anti-monotone set). *A set $H \subseteq \{0, 1\}^n$ is said to be anti-monotone if and only if whenever $x \in H$, we have $y \in H$ for all $y \preceq x$.*

We record the following simple result about anti-monotone conjunctions, which will be useful for understanding and analyzing Algorithm 2:

► **Lemma 14.** *A function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is an anti-monotone conjunction if and only if $f^{-1}(1)$ is anti-monotone and is a linear subspace of $\{0, 1\}^n$.*

Proof. The “if” direction follows immediately from the following easy result of [35]:

▷ **Claim 15** (Claim 18 of [35]). *If H is a monotone and affine subspace of $\{0, 1\}^n$, then $H = \{x \in \{0, 1\}^n : (x_{i_1} = 1) \wedge \dots \wedge (x_{i_k} = 1)\}$ for some k and some $i_1, \dots, i_k \in [n]$. Similarly if H is an anti-monotone and linear subspace, then $H = \{x \in \{0, 1\}^n : (x_{i_1} = 0) \wedge \dots \wedge (x_{i_k} = 0)\}$ for some k and some $i_1, \dots, i_k \in [n]$.*

For the other direction, suppose that $f(x) = x_{i_1} \wedge \dots \wedge x_{i_k}$ is an anti-monotone conjunction. Then clearly $f^{-1}(1)$ is an anti-monotone set. Moreover, for $x, y \in f^{-1}(1)$, $f(x \oplus y) = \neg(x_{i_1} \oplus y_{i_1}) \wedge \dots \wedge \neg(x_{i_k} \oplus y_{i_k})$. Since x, y are both in $f^{-1}(1)$, $x_{i_1}, \dots, x_{i_k}, y_{i_1}, \dots, y_{i_k}$ are all zero, and thus $f(x \oplus y) = 1$. Recalling Definition 11, we have that $f^{-1}(1)$ is a linear subspace, so the only-if direction also holds. ◀

3.2 Intuition: the tester of [35] and our tester

We begin with an overview of our algorithm and its proof. At a bird’s eye view, our algorithm has two phases: Phase 1 tests whether $f^{-1}(1)$ is close to a linear subspace, and Phase 2 tests whether f is anti-monotone. (Note that, while [10] already gave a general tester for monotonicity in the relative error setting, its complexity has a dependence on $2^n/|f^{-1}(1)|$ which is necessary in the general case. Here we want a much more efficient tester that is linear in $1/\varepsilon$, so we cannot use the monotonicity tester of [10].)

If f is an anti-monotone conjunction then both Phase 1 and Phase 2 tests will clearly pass; the hard part is to prove soundness, showing that when f is far from every anti-monotone conjunction, then the algorithm will reject with high probability. The crux of the analysis is to show that if f is close to a linear subspace but far from all anti-monotone conjunctions, then the Phase 2 anti-monotonicity test will fail with high probability. As in [35], we accomplish this with the aid of an intermediate function, g_f that we will use in the analysis. In Phase 1 of the algorithm, we will actually prove something stronger: if f passes Phase 1 with good probability, we show that f is close to a particular function, g_f (that is defined based on f), where $g_f^{-1}(1)$ is a linear subspace. The function g_f is chosen to help us downstream in the analysis, to argue that if Phase 2 passes, then g_f is anti-monotone with good probability; since $g_f^{-1}(1)$ is a linear subspace, by Lemma 14 this implies g_f is an anti-monotone conjunction, thereby contradicting the assumption that f is far from every anti-monotone conjunction.

Finally, we remark that as described in Section 2.3, our relative-error tester yields an $O(1/\varepsilon)$ -query one-sided tester in the standard model. This strengthened the result of [35], which gave an $O(1/\varepsilon)$ -query algorithm with two-sided error. Comparing our analysis with [35], while we borrow many of their key ideas, our analysis is arguably simpler in a couple of ways. First since we are forced to create a tester that doesn’t have the ability to estimate the size of the conjunction, this eliminates the initial phase in the [35] which does this estimation,

■ **Algorithm 2** Relative-error Anti-monotone Conjunction Tester. Here c_1 and c_2 are two sufficiently large absolute constants.

Input: Oracle access to $\text{MQ}(f)$, $\text{Samp}(f)$ and an accuracy parameter $\varepsilon > 0$.

Output: “Reject” or “Accept.”

Phase 1:

- 1: Repeat the following c_1/ε times :
- 2: Draw $\mathbf{x}, \mathbf{y} \sim \text{Samp}(f)$.
- 3: If $\text{MQ}(f)(\mathbf{x} \oplus \mathbf{y}) = 0$, halt and reject f .

Phase 2:

- 4: Repeat the following c_2 times :
 - 5: Draw $\mathbf{x} \sim \text{Samp}(f)$.
 - 6: Draw a uniform random $\mathbf{y} \preceq \mathbf{x}$.
 - 7: Draw $\mathbf{u} \sim \text{Samp}(f)$.
 - 8: If $\text{MQ}(f)(\mathbf{y} \oplus \mathbf{u}) = 0$, halt and reject f .
 - 9: Accept f .
-

giving a more streamlined argument and enabling us to give an $O(1/\varepsilon)$ -query algorithm with one-sided error. Second, by a simple transformation (see Remark 10), we test whether f is close to a monotone *linear* subspace rather than close to a monotone affine subspace, which makes both the test and analysis easier.

3.3 The relative-error anti-monotone conjunction tester

It remains to establish the following theorem, which implies Theorem 1 via Remark 10:

► **Theorem 16** (Relative-error anti-monotone conjunction tester). *Algorithm 2 is a one-sided non-adaptive algorithm which, given $\text{MQ}(f)$ and $\text{Samp}(f)$, is an ε -relative-error tester for anti-monotone conjunctions over $\{0, 1\}^n$, making $O(1/\varepsilon)$ oracle calls.*

The $O(1/\varepsilon)$ bound on the number of oracle calls is immediate from inspection of Algorithm 2, so it remains to establish correctness (including soundness and completeness). These are established in Theorem 17 and Theorem 18 respectively, which are proved in the full version.

► **Theorem 17.** *If $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is any anti-monotone conjunction, then Algorithm 2 accepts f with probability 1.*

► **Theorem 18.** *Suppose that $f : \{0, 1\}^n \rightarrow \{0, 1\}$ satisfies $\text{rel-dist}(f, f') > \varepsilon$ for every anti-monotone conjunction f' . Then Algorithm 2 rejects f with probability at least 0.9.*

We can now prove Theorem 1 about CONJ-TEST:

Proof of Theorem 1. Recall that CONJ-TEST first samples $\mathbf{y} \sim F$ and then runs Algorithm 2 on $f_{\mathbf{y}}$. When f is a conjunction, $f_{\mathbf{y}}$ is an anti-monotone conjunction and thus, by Theorem 17 CONJ-TEST accepts with probability 1. When f is far from any conjunction, $f_{\mathbf{y}}$ must be far from any anti-monotone conjunction. So by Theorem 18, CONJ-TEST rejects with probability at least 0.9. ◀

3.4 Robustness of CONJ-TEST to faulty oracles

CONJ-TEST will play a crucial role in the relative-error tester for decision lists in the next section. As will become clear there, CONJ-TEST will be run on a function f to test whether it is relative-error close to a conjunction when it can only access a “faulty” version of the $\text{Samp}(f)$ oracle. For later reference we record the following two explicit statements about the performance of CONJ-TEST; we show that as long as the faulty version of $\text{Samp}(f)$ satisfies some mild conditions, CONJ-TEST still outputs the correct answer with high probability. In both statements, we write $\mathcal{D}$ to denote the distribution over $\{0,1\}^n$ underlying the faulty sampling oracle $\text{Samp}^*(f)$, which is not necessarily uniform over $F = f^{-1}(1)$ (and in Theorem 20 need not even necessarily be supported over F). Both statements are proved in the full version of the paper.

► **Theorem 19.** *Let $f : \{0,1\}^n \rightarrow \{0,1\}$ be a conjunction. Assume that the oracle $\text{Samp}^*(f)$ always returns some string in F (i.e., the distribution $\mathcal{D}$ is supported over F). Then CONJ-TEST, running on $\text{MQ}(f)$ and $\text{Samp}^*(f)$, accepts f with probability 1.*

► **Theorem 20.** *Suppose that $f : \{0,1\}^n \rightarrow \{0,1\}$ satisfies $\text{rel-dist}(f, f') > \varepsilon$ for any conjunction f' . Let $\text{Samp}^*(f)$ be a sampling oracle for f such that the underlying distribution $\mathcal{D}$ satisfies*

$$\Pr_{x \sim \mathcal{D}} [x = z] \geq \frac{1}{20|F|}, \quad \text{for any } z \in F.$$

Then CONJ-TEST, running on $\text{MQ}(f)$ and $\text{Samp}^(f)$, rejects f with probability at least 0.9.*

4 A relative-error testing algorithm for decision lists

Let’s first recall how a decision list is represented and some basic facts about decision lists.

4.1 Basics about decision lists

A decision list $f : \{0,1\}^n \rightarrow \{0,1\}$ can be represented as a list of rules:

$$(x_{i_1}, b_1, v_1), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}},$$

where $i_1, \dots, i_k \in [n]$ are indices of variables and $b_1, \dots, b_k, v_1, \dots, v_k, v_{\text{default}} \in \{0,1\}$. For each $x \in \{0,1\}^n$, $f(x)$ is set to be v_j with the smallest j such that $x_{i_j} = b_j$, and is set to be v_{default} if $x_{i_j} \neq b_j$ for all $j \in [k]$. Without loss of generality, we will make the following assumptions:

- *Each variable appears at most once in the list of rules (i.e., no two i_j ’s are the same).* Indeed if two rules of the form (x_{i_j}, b_j, v_j) and $(x_{i_\ell}, b_\ell, v_\ell)$ with $j < \ell$ and $i_j = i_\ell$ are present, then either $b_j = b_\ell$, in which case the latter rule can be trivially deleted, or $b_\ell = 1 - b_j$, in which case the latter rule and all rules after it can be replaced by $v_{\text{default}} = v_\ell$, without changing the function f represented.
- *There exists at least one rule (x_{i_j}, b_j, v_j) with $v_j = 1$.* This is because if $v_1 = \dots = v_k = 0$, then either (1) $v_{\text{default}} = 0$, in which case f is the all-0 function; (2) $v_{\text{default}} = 1$ and at least one variable x_i does not appear in the list, in which case we can insert $(x_i, 1, 1)$ before v_{default} , which does not change the function and makes sure at least one v_j is 1; or (3) $v_{\text{default}} = 1$ but all variables appear in the list, in which case we have $|f^{-1}(1)| = 1$. Functions f with $|f^{-1}(1)| \leq 1$ are easy to recognize with the sampling oracle.

In the rest of the section, whenever we talk about decision lists, the function f is assumed to satisfy $|f^{-1}(1)| > 1$ so it can be represented by a list of rules that satisfies the two conditions above.

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a decision list represented by $(x_{i_1}, b_1, v_1), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}}$. The following decomposition of the list plays a crucial role both in the algorithm and its analysis:

- Let $p \in [k]$ be the smallest index with $v_p = 1$. We will call p the **pivot**.
- The prefix before (x_{i_p}, b_p, v_p) , i.e., $(x_{i_1}, b_1, 0), \dots, (x_{i_{p-1}}, b_{p-1}, 0)$ will be called the **head**. The key insight is that we can view the head as a conjunction $C : \{0, 1\}^n \rightarrow \{0, 1\}$, where

$$C(x) = (x_{i_1} = 1 - b_1) \wedge \dots \wedge (x_{i_{p-1}} = 1 - b_{p-1}).$$

And we write $\mathcal{H}_C := \{i_1, \dots, i_{p-1}\}$ to denote the set of $p-1$ variables in the conjunction.

- The part after (x_{i_p}, b_p, v_p) , i.e., $(x_{i_{p+1}}, b_{p+1}, v_{p+1}), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}}$ will be called the **tail**. Let $L : \{0, 1\}^n \rightarrow \{0, 1\}$ denote the decision list represented by the pivot and tail:

$$(x_{i_p}, b_p, v_p = 1), (x_{i_{p+1}}, b_{p+1}, v_{p+1}), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}}.$$

Then it is easy to verify that $f(x) = C(x) \wedge L(x)$ for all $x \in \{0, 1\}^n$.

We record the following simple fact about the decomposition $f = C \wedge L$:

► **Observation 21.** All $z \in f^{-1}(1)$ satisfy $C(z) = 1$, i.e., $z_{i_j} = 1 - b_j$ for all $i_j \in \mathcal{H}_C$.

We will need the following lemma about the decomposition $f = C \wedge L$ of a decision list, which says that coordinates in the tail are fairly unbiased. The intuition is that at least half the points in $f^{-1}(1)$ are accepted by the rule $(x_{i_p}, b_p, 1)$, and for these points the coordinates x_{i_j} with $j > p$, do not matter, and thus are distributed uniformly.

► **Lemma 22.** Let f be a decision list represented by $(x_{i_1}, b_1, v_1), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}}$, and let p be its pivot. Then we have

$$\Pr_{z \sim f^{-1}(1)} [z_{i_p} = b_p] \geq 1/2 \quad \text{and} \quad \Pr_{z \sim f^{-1}(1)} [z_{i_j} = 1] \in [1/4, 3/4], \quad \text{for all } j : p < j \leq k.$$

Proof. Using the decomposition $f = C \wedge L$, we have

$$\{z \in \{0, 1\}^n : C(z) = 1 \text{ and } z_{i_p} = b_p\} \subseteq f^{-1}(1) \subseteq \{z \in \{0, 1\}^n : C(z) = 1\}. \quad (3)$$

Given that the LHS is exactly half of the size of the set on the RHS, the first part follows.

Furthermore, consider any i_j with $p < j \leq k$. Exactly half of the strings z in the set on the LHS of Equation (3) have $z_{i_j} = 1$ and exactly half of them have $z_{i_j} = 0$. Thus, we have for any $b \in \{0, 1\}$,

$$\Pr_{z \sim f^{-1}(1)} [z_{i_j} = b] \geq \frac{|\{z : C(z) = 1, z_{i_p} = b_p \text{ and } z_{i_j} = b\}|}{|f^{-1}(1)|} = \frac{|\{z : C(z) = 1\}|}{4|f^{-1}(1)|} \geq \frac{1}{4},$$

where the last inequality used the second inclusion of Equation (3). This finishes the proof. ◀

4.2 The Algorithm

Our main algorithm for testing decision lists in relative distance is presented in Algorithm 3 of the full paper which uses two subroutines described in Algorithm 4 of the full paper and Algorithm 5 of the full paper, respectively. A complete description and analysis of Algorithm 3 of the full paper is given in the full paper; here we give an overview of the algorithm in this subsection, to give some intuition behind it about why it accepts functions that are decision lists and rejects functions that are far from decision lists in relative distance with high probability.

We will start with the case when $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is a decision list, and explain how the algorithm extracts information about the underlying decomposition $C \wedge L$ of f so that f passes all the tests in Algorithm 3 of the full paper with high probability. Towards the end of this subsection, we will switch to the case when f is far from any decision list in relative distance, and discuss why passing all the tests in Algorithm 3 of the full paper gives evidence that f is close in relative distance to some decomposition $C \wedge L$ of a decision list, which should not happen given the assumption that f is far from every decision list in relative distance.

The case when f is a decision list. Let f be a decision list represented by $(x_{i_1}, b_1, v_1), \dots, (x_{i_k}, b_k, v_k), v_{\text{default}}$. Let p be the pivot and $f = C \wedge L$ be the decomposition, where C is a conjunction over variables in $\mathcal{H}_C := \{i_1, \dots, i_{p-1}\}$ and L is a decision list formed by the pivot and the tail of the list.

The first step of Algorithm 3 of the full paper draws samples from $\text{Samp}(f)$ to partition variables into $U \cup R$:

2: Draw $O(1/\varepsilon)$ samples $\mathbf{z} \sim \text{Samp}(f)$, and call the set of samples $\mathbf{S}$. Let

$$U = \{i \in [n] : \exists b \in \{0, 1\} \text{ s.t. } \mathbf{z}_i = b \text{ for all } \mathbf{z} \in \mathbf{S}\} \quad \text{and} \quad R = [n] \setminus U$$

and $\mathbf{u} \in \{0, 1\}^U$ be the unique string such that $\mathbf{u}_i = \mathbf{z}_i$ for all $i \in U$ and $\mathbf{z} \in \mathbf{S}$.

So U is the set of *unanimous coordinates* of samples in $\mathbf{S}$ and thus, we always have $\mathcal{H}_C \subseteq U$ and $\mathbf{u}_{i_j} = 1 - b_j$ for all $i_j \in \mathcal{H}_C$ by Observation 21. Ideally, we would hope for R to contain all variables in the tail, i.e., $i_{p+1}, \dots, i_k$; however, this won't be the case in general, unfortunately, since k can be much bigger than p (e.g., $k - p$ could be polynomial in n). But by Lemma 22 and given that $\mathbf{S}$ contains $O(1/\varepsilon)$ samples from $\text{Samp}(f)$, it is easy to show that with high probability, the variables occurring in the beginning part of the tail (excluding those variables that are hidden too deep in the tail) all lie in R with high probability, and it turns out that this is all we will need for the proof of completeness to go through.

Now, the pivot variable x_{i_p} can go either way, since it may be highly biased towards b_p . However, if it is too biased towards b_p so that none of the samples $\mathbf{z}$ in $\mathbf{S}$ have $\mathbf{z}_p = 1 - b_p$, then one can argue that f must be very close to a conjunction over $\{0, 1\}^n$ in relative distance. Because of this, Step 0 of Algorithm 3 of the full paper (which consists in running CONJ-TEST on f with distance ε) will accept f with high probability.

Proceeding to Step 2, assume without loss of generality that Algorithm 3 of the full paper has found U and R such that $\mathcal{H}_C \subseteq U$ and R contains variables in the beginning part of L , including i_p , and the string $\mathbf{u} \in \{0, 1\}^U$ satisfies $\mathbf{u}_{i_j} = 1 - b_j$ for all $i_j \in \mathcal{H}_C$. This implies the following properties:

1. The function $f|_{\mathbf{u}}$ over $\{0,1\}^{\mathbf{R}}$ is a “dense” decision list, where “dense” means the number of its satisfying assignments is at least half of $2^{|\mathbf{R}|}$, and it is a decision list because the restriction of any decision list remains a decision list.
2. For any $\alpha \in \{0,1\}^{\mathbf{U}}$, $f|_{\alpha}$ is either the all-0 function (because α does not satisfy the conjunction C), or very close to $f|_{\mathbf{u}}$ (because $\mathbf{R}$ contains the beginning part of the tail) and thus, both $f|_{\mathbf{u}}$ and $f|_{\alpha}$ are close to L (when viewed as a decision list over $\mathbf{R}$) under the uniform distribution.

Given these properties of $\mathbf{U}$, $\mathbf{R}$ and $\mathbf{u}$, it is easy to see that f passes both tests in Step 2 with high probability:

-
- 3: Draw $O(1)$ points $\mathbf{w} \sim \{0,1\}^{\mathbf{R}}$; halt and reject f if less than $1/4$ of the sampled $\mathbf{w}$'s have $f(\mathbf{u} \circ \mathbf{w}) = 1$.
 - 4: Run the algorithm of Theorem 6 on $f|_{\mathbf{u}}$ with $\varepsilon/100$; halt and reject f if it rejects.
-

since line 3 checks whether $f|_{\mathbf{u}}$ is “dense,” and line 4 checks if $f|_{\mathbf{u}}$ is close to a decision list under the uniform distribution. The function f would also pass the tests in Step 3 with high probability:

-
- 5: Repeat $O(1/\varepsilon)$ times:
 - 6: Draw $\mathbf{z} \sim \text{Samp}(f)$ and $\mathbf{w} \sim \{0,1\}^{\mathbf{R}}$, if $f(\mathbf{z}_{\mathbf{U}} \circ \mathbf{w}) \neq f(\mathbf{u} \circ \mathbf{w})$, halt and reject f .
-

as for any $z \in f^{-1}(1)$, $\mathbf{z}_{\mathbf{U}}$ satisfies C and thus, $f|_{\mathbf{z}_{\mathbf{U}}}$ is close to $f|_{\mathbf{u}}$ under the uniform distribution.

For step 4, let $\mathbf{\Gamma} : \{0,1\}^{\mathbf{U}} \rightarrow \{0,1\}$ be defined as follows:

$$\mathbf{\Gamma}(\alpha) = \begin{cases} 1 & \text{if } \Pr_{\mathbf{w} \sim \{0,1\}^{\mathbf{R}}} [f(\alpha \circ \mathbf{w}) = 1] \geq 1/16 \\ 0 & \text{otherwise} \end{cases}$$

Now, observe that for any $\alpha \in \{0,1\}^{\mathbf{U}}$, either α does not satisfy C , in which case $f|_{\alpha}$ is the all-0 function, or α satisfies C , in which case $f|_{\alpha}$ is close to $f|_{\mathbf{u}}$ and thus $\mathbf{\Gamma}(\alpha) = 1$ (because $f|_{\mathbf{u}}$ is “dense”). As a result, $\mathbf{\Gamma}$ is indeed the same as C and thus it is a conjunction over $\{0,1\}^{\mathbf{U}}$. From this, it may seem straightforward to conclude that $\mathbf{\Gamma}$ always passes the test in Step 4 below:

-
- 7: Run CONJ-TEST with $\text{Samp}(\mathbf{\Gamma}) \leftarrow (\text{Algorithm 5 of the full paper})$, $\text{MQ}(\mathbf{\Gamma}) \leftarrow (\text{Algorithm 4 of the full paper})$ to test if $\mathbf{\Gamma}$ is $(\varepsilon/100)$ -close to a conjunction in relative distance; halt and return the same answer.
-

except that we don't actually have direct access to sampling and query oracles of $\mathbf{\Gamma}$. They are simulated by Algorithm 5 of the full paper and Algorithm 4 of the full paper, respectively. We show in the proof that (1) Algorithm 5 of the full paper (for the sampling oracle of $\mathbf{\Gamma}$) only returns samples in $\mathbf{\Gamma}^{-1}(1)$; and (2) with high probability, Algorithm 4 of the full paper returns the correct answer to all $O(1/\varepsilon)$ membership queries made by CONJ-TEST. From this we can apply Theorem 19 to conclude that CONJ-TEST accepts with high probability.

The case when f is far from every decision list. Let's now switch to the case when f is ε -far from any decision list in relative distance. Because conjunctions are special cases of decision lists, Step 0 (which consists in running CONJ-TEST on f) does not accept f with high probability and Algorithm 3 of the full paper goes through Step 1 to obtain $\mathbf{U}$, $\mathbf{R}$ and $\mathbf{u} \in \{0, 1\}^{\mathbf{U}}$. Let's consider what conditions f needs to satisfy in order for it to pass tests in Steps 2 and 3 with high probability:

1. To pass Step 2, $f|_{\mathbf{u}}$ over $\{0, 1\}^{\mathbf{R}}$ needs to be “dense:” $|f|_{\mathbf{u}}^{(-1)}(1)| \geq 2^{|\mathbf{R}|}/8$, and must also be $(\varepsilon/100)$ -close to a decision list under the uniform distribution.
2. To understand the necessary condition posed on f to pass Step 3, let's introduce the following function $\mathbf{g} : \{0, 1\}^n \rightarrow \{0, 1\}$, which is closely related to $\mathbf{\Gamma}$ defined earlier and will play a crucial role in the analysis:

$$\mathbf{g}(x) = \mathbf{\Gamma}(x_{\mathbf{U}}) \wedge f(\mathbf{u} \circ x_{\mathbf{R}}).$$

We show that to pass Step 3 with high probability, it must be the case that $\text{rel-dist}(f, \mathbf{g})$ is small. This means that for each $\alpha \in \{0, 1\}^{\mathbf{U}}$, one can replace $f|_{\alpha}$ by the all-0 function if $\mathbf{\Gamma}(\alpha) = 0$ (which means that $f|_{\alpha}$ is not dense enough), and replace $f|_{\alpha}$ by $f|_{\mathbf{u}}$ otherwise, and this will only change a small number of bits of f (relative to $|f^{-1}(1)|$). Note that $\mathbf{g}$ is getting closer to the decomposition of a decision list since we know $f|_{\mathbf{u}}$ is close to a decision list over $\{0, 1\}^{\mathbf{R}}$.

Assuming all the necessary conditions summarized above for f to pass Steps 2 and 3 and using the assumption that f is far from decision lists in relative distance, we show that $\mathbf{\Gamma}$ must be far from conjunctions in relative distance. Finally, given the latter, it can be shown that CONJ-TEST in Step 4 rejects with high probability, again by analyzing performance guarantees of Algorithm 5 of the full paper and Algorithm 4 of the full paper as simulators of sampling and query oracles of $\mathbf{\Gamma}$, respectively, so that Theorem 20 for CONJ-TEST can be applied.

References

- 1 A. Belovs and E. Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th ACM Symposium on Theory of Computing*, pages 1021–1032, 2016.
- 2 Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing - Problems and Techniques*. Springer, 2022. doi:10.1007/978-981-16-8622-1.
- 3 Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47:549–595, 1993. Earlier version in STOC'90. doi:10.1016/0022-0000(93)90044-W.
- 4 Nader H. Bshouty. Almost optimal testers for concise representations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM*, pages 5:1–5:20, 2020. doi:10.4230/LIPICS.APPROX/RANDOM.2020.5.
- 5 Nader H. Bshouty and Oded Goldreich. On properties that are non-trivial to test. *Electron. Colloquium Comput. Complex.*, TR22-013, 2022. URL: <https://eccc.weizmann.ac.il/report/2022/013>.
- 6 Deeparnab 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*, pages 411–418, 2013. doi:10.1145/2488608.2488660.
- 7 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Symposium on Theory of Computing Conference, STOC*, pages 419–428, 2013. doi:10.1145/2488608.2488661.

- 8 Deeparnab Chakrabarty and C. Seshadhri. A $\tilde{O}(n)$ non-adaptive tester for unateness. *CoRR*, abs/1608.06980, 2016. [arXiv:1608.06980](#).
- 9 Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *Automata, Languages and Programming - 38th International Colloquium, ICALP*, pages 545–556, 2011. doi:10.1007/978-3-642-22006-7_46.
- 10 X. Chen, A. De, Y. Huang, S. Nadimpalli, R. Servedio, and T. Yang. Relative error monotonicity testing. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2025.
- 11 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean Function Monotonicity Testing Requires (Almost) $n^{1/2}$ Non-adaptive Queries. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015*, pages 519–528, 2015. doi:10.1145/2746539.2746570.
- 12 Xi Chen, Yumou Fei, and Shyamal Patel. Distribution-free testing of decision lists with a sublinear number of queries. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1051–1062, 2024. doi:10.1145/3618260.3649717.
- 13 Xi Chen and Shyamal Patel. Distribution-free testing for halfspaces (almost) requires PAC learning. In *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1715–1743, 2022. doi:10.1137/1.9781611977073.70.
- 14 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for testing monotonicity. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science*, pages 286–295, 2014.
- 15 Xi Chen and Erik Waingarten. Testing unateness nearly optimally. In *Proceedings of the 51th ACM Symposium on Theory of Computing*, 2019.
- 16 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 523–536, 2017. doi:10.1145/3055399.3055461.
- 17 Xi Chen, Erik Waingarten, and Jinyu Xie. Boolean unateness testing with $\tilde{O}(n^{3/4})$ adaptive queries. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 868–879, 2017. doi:10.1109/FOCS.2017.85.
- 18 Xi Chen and Jinyu Xie. Tight bounds for the distribution-free testing of monotone conjunctions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 54–71, 2016. doi:10.1137/1.9781611974331.CH5.
- 19 I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. Testing for concise representations. In *Proc. 48th Ann. Symposium on Computer Science (FOCS)*, pages 549–558, 2007.
- 20 E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proc. 34th Annual ACM Symposium on the Theory of Computing*, pages 474–483, 2002.
- 21 Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-testing/correcting for polynomials and for approximate functions. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 32–42. ACM, 1991. doi:10.1145/103418.103429.
- 22 Dana Glasner and Rocco A. Servedio. Distribution-free testing lower bound for basic boolean functions. *Theory of Computing*, 5(1):191–216, 2009. doi:10.4086/toc.2009.v005a010.
- 23 O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.
- 24 Oded Goldreich, editor. *Computational Complexity and Property Testing - On the Interplay Between Randomness and Computation*, volume 12050 of *Lecture Notes in Computer Science*. Springer, 2020. doi:10.1007/978-3-030-43662-9.
- 25 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samordinsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. doi:10.1007/S004930070011.

- 26 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998. doi:10.1145/285055.285060.
- 27 Oded Goldreich and Dana Ron. One-sided error testing of monomials and affine subspaces. *Electron. Colloquium Comput. Complex.*, TR20-068, 2020. URL: <https://eccc.weizmann.ac.il/report/2020/068>.
- 28 S. Halevy and E. Kushilevitz. Distribution-Free Property Testing. In *Proceedings of the Seventh International Workshop on Randomization and Computation*, pages 302–317, 2003.
- 29 S. Halevy and E. Kushilevitz. A lower bound for distribution-free monotonicity testing. In *Proceedings of the Ninth International Workshop on Randomization and Computation*, pages 330–341, 2005.
- 30 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM J. Comput.*, 47(6):2238–2276, 2018. doi:10.1137/16M1065872.
- 31 Subhash Khot and Igor Shinkar. An $\tilde{o}(n)$ queries adaptive tester for unateness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 37:1–37:7, 2016.
- 32 Amit Levi and Erik Waingarten. Lower bounds for tolerant junta and unateness testing via rejection sampling of graphs. In Avrim Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 52:1–52:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPICS.ITCS.2019.52.
- 33 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010. doi:10.1137/070707890.
- 34 R. O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
- 35 M. Parnas, D. Ron, and A. Samorodnitsky. Testing Basic Boolean Formulae. *SIAM J. Disc. Math.*, 16:20–46, 2002. doi:10.1137/S0895480101407444.
- 36 D. Ron. Property Testing: A Learning Theory Perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008. doi:10.1561/22000000004.
- 37 L. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.