Testing Juntas and Junta Subclasses with Relative Error

Xi Chen XC2198@COLUMBIA.EDU
William Pires WP2294@COLUMBIA.EDU
Toniann Pitassi TONIPITASSI@GMAIL.COM
Rocco A. Servedio ROCCO@CS.COLUMBIA.EDU
Columbia University, New York, NY, USA

Editors: Nika Haghtalab and Ankur Moitra

Abstract

This paper considers the junta testing problem in a recently introduced "relative error" variant of the standard Boolean function property testing model. In relative-error testing we measure the distance from f to g, where $f, g : \{0,1\}^n \to \{0,1\}$, by the ratio of $|f^{-1}(1) \triangle g^{-1}(1)|$ (the number of inputs on which f and g disagree) to $|f^{-1}(1)|$ (the number of satisfying assignments of f), and we give the testing algorithm both black-box access to f and also access to independent uniform samples from $f^{-1}(1)$.

Chen et al. (2025a) observed that the class of k-juntas is $\mathrm{poly}(2^k, 1/\varepsilon)$-query testable in the relative-error model, and asked whether $\mathrm{poly}(k, 1/\varepsilon)$ queries is achievable. We answer this question affirmatively by giving a $\tilde{O}(k/\varepsilon)$-query algorithm, matching the optimal complexity achieved in the less challenging standard model. Moreover, as our main result, we show that any *subclass* of k-juntas that is closed under permuting variables is relative-error testable with a similar complexity. This gives highly efficient relative-error testing algorithms for a number of well-studied function classes, including size-k decision trees, size-k branching programs, and size-k Boolean formulas.

Keywords: Property testing, Boolean functions, relative-error testing

1. Introduction

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a *k-junta* if it depends on at most k of its n input variables. The class of k-juntas arises across many disparate contexts in theoretical computer science: for example, "junta theorems" are of fundamental importance in the analysis of Boolean functions Nisan and Szegedy (1992); Friedgut (1998); Bourgain (2002); Kindler and Safra (2004); Filmus (2004), "junta learning" is a paradigmatic problem capturing the difficulty of identifying relevant variables in learning theory Blum (1994, 2003); Mossell et al. (2003); Valiant (2015), and, closest to our concerns, juntas play a central role in the field of *property testing*.

Starting with the influential work of Fischer et al. Fischer et al. (2004), testing whether an unknown and arbitrary function $f : \{0,1\}^n \to \{0,1\}$ is a k-junta has emerged as one of the touchstone problems in Boolean function property testing. In the standard testing model introduced by Blum, Luby and Rubinfeld Blum et al. (1993); Rubinfeld and Sudan (1996), junta testing has been intensively studied from many perspectives, with upper and lower bounds for both adaptive and non-adaptive junta testing algorithms given in a long sequence of papers Chockler and Gutfreund (2004); Fischer et al. (2004); Blais (2008, 2009); Buhrman et al. (2013); Servedio et al. (2015); Sağlam (2018); Chen et al. (2018). Junta testing has also been studied in several more recently introduced and challenging property testing frameworks that have been proposed over the years. These include the model of *distribution-free* property testing Halevy and Kushilevitz (2007); Liu et al. (2019); Bshouty (2019); Belovs (2019), the *tolerant testing* model De et al. (2019); Iyer et al. (2021); Pallavoor et al. (2022); Levi and Waingarten (2019); Chen and Patel (2023); Chen et al. (2024a); Nadimpalli and Patel (2024), *quantum* property testing models Atıcı and Servedio (2007); Ambainis et al. (2016); Belovs (2019); Chen et al. (2023) and the model of *active property testing* Balcan et al. (2012); Alon (2012).

The contribution of this paper is to resolve (a generalization of) the junta testing problem in the recently introduced *relative-error* model of Boolean function property testing Chen et al. (2025a,c,b). Our main result is an efficient relative-error testing algorithm for k-juntas, and more generally for any subclass of k-juntas that is closed under permutations of the variables, resolving an open question posed in Chen et al. (2025a). Our algorithm for testing subclasses of juntas yields efficient relative-error testing algorithms for a number of natural function classes, including: size-s decision trees, size-s branching programs, and size-s Boolean formulas, whose standard-model and distribution-free testability was studied in Diakonikolas et al. (2007); Chakraborty et al. (2011); Bshouty (2020).

1.1. Prior results on relative-error testing

The relative-error property testing model for Boolean functions was introduced in Chen et al. (2025a) to provide an appropriate framework for testing *sparse* Boolean functions (ones with few satisfying assignments). Recall in the standard Boolean function property testing model,[1] the distance between functions $f, g : \{0,1\}^n \to \{0,1\}$ is the normalized Hamming distance $|f^{-1}(1) \triangle g^{-1}(1)|/2^n$, and hence any f that has fewer than $\varepsilon 2^n$ satisfying assignments will trivially be ε-close to the constant-0 function. Motivated by graph property testing models introduced in Goldreich and Ron (2002); Parnas and Ron (2002) which are suitable for *sparse* graphs, Chen et al. (2025a) defined the *relative-error* Boolean function property testing model to measure the distance between the function $f : \{0,1\}^n \to \{0,1\}$ that is being tested and any other function g to be $\text{rel-dist}(f,g) := |f^{-1}(1) \triangle g^{-1}(1)|/|f^{-1}(1)|$. With this definition, g is close to f only if the symmetric difference is small "at the scale of f," and it is reasonable to consider testing sparse Boolean functions. The model also allows the testing algorithm to obtain i.i.d. uniform elements of $f^{-1}(1)$ by calling a "random sample" oracle (since if only black-box $\text{MQ}(f)$ queries were allowed, as in the standard model, then for sparse f a very large number of queries would be required to obtain any information about f at all).

Chen et al. (2025a) established some general results about relative-error testing vis-a-vis standard-model property testing. They showed that any class of functions $\mathcal{C}$ can be tested in the standard model with essentially no more queries than are required for relative-error testing (so relative-error testing is always at least as hard as standard testing), and gave a simple example of an (artificial) class $\mathcal{C}'$ for which relative-error testing algorithms require $2^{\Omega(n)}$ queries but standard-model testers require only constantly many queries (so relative-error testing can be strictly harder than standard testing). However, the chief focus of Chen et al. (2025a) was on relative-error *monotonicity testing*. The main results of Chen et al. (2025a) showed that the complexity of relative-error monotonicity testing is within a polynomial factor of the complexity of standard-model monotonicity testing (see the first row of Table 1).

In subsequent work, Chen et al. (2025b) studied relative-error testing of *halfspaces*, and gave an $\tilde{\Omega}(\log n)$ lower bound on relative-error halfspace testing algorithms. In contrast, there are standard-model halfspace testing algorithms with query complexity $\text{poly}(1/\varepsilon)$ (independent of n) Matulef et al. (2010a), thus showing that the natural class of halfspaces witnesses an arbitrarily large gap between the query complexity in the relative-error model and the standard model. On the other hand, Chen et al. (2025c) showed that for two other natural classes of Boolean functions, namely *conjunctions* and *decision lists*, there are $\tilde{O}(1/\varepsilon)$-query relative-error testing algorithms, matching the performance of the best standard-model testers Parnas et al. (2002); Bshouty (2020). See Table 1 for a summary of how relative-error testing results contrast with standard-model testing results for various well-studied function classes.

1.2. Relative-error junta testing and our results

Chen et al. (2025a) posed a number of questions for future work on relative-error property testing, the first of which concerns junta testing. We recall that in the standard model, Blais (2009) gave an ε-testing

1. See Appendix A for detailed definitions of both this model and the relative-error testing model.

Class of functions	Standard-model testing complexity	Relative-error testing complexity
Monotone functions	$\tilde{O}(\sqrt{n}/\varepsilon^2)$ Khot et al. (2018), $\tilde{\Omega}(n^{1/3})$ Chen et al. (2017)	$O(n/\varepsilon)$ Chen et al. (2025a) $\tilde{\Omega}(n^{2/3})$ Chen et al. (2025a)
Halfspaces	$\mathrm{poly}(1/\varepsilon)$ Matulef et al. (2010b)	$\tilde{\Omega}(\log n)$ Chen et al. (2025b)
Conjunctions	$\Theta(1/\varepsilon)$ Parnas et al. (2002)	$\Theta(1/\varepsilon)$ Chen et al. (2025c)
Decision lists	$\tilde{\Theta}(1/\varepsilon)$ Bshouty (2020)	$\tilde{\Theta}(1/\varepsilon)$ Chen et al. (2025c)
k-juntas	$\tilde{O}(k/\varepsilon)$ Blais (2009), $\tilde{\Omega}(k)$ Chockler and Gutfreund (2004); Sağlam (2018)	$\tilde{O}(k/\varepsilon)$ **[This paper]**
Subclass $\mathcal{C}(k)$ of k-juntas	$\tilde{O}\left(\frac{k+\log\lvert\mathcal{C}(k)^*\rvert}{\varepsilon}\right)$ Bshouty (2020)	$\tilde{O}\left(\frac{k\log\lvert\mathcal{C}(k)^*\rvert}{\varepsilon}\right)$ **[This paper]**
size-k decision trees	$\tilde{O}(k/\varepsilon)$ Bshouty (2020)	$\tilde{O}(k^2/\varepsilon)$ **[This paper]**
size-k branching programs	$\tilde{O}(k/\varepsilon)$ Bshouty (2020)	$\tilde{O}(k^2/\varepsilon)$ **[This paper]**
size-k Boolean formulas	$\tilde{O}(k/\varepsilon)$ Bshouty (2020)	$\tilde{O}(k^2/\varepsilon)$ **[This paper]**

Table 1: Known results on the number of oracle calls needed for standard-model ε-testing and relative-error ε-testing of various classes of n-variable Boolean functions. For relative-error testing, the entry indicates the total number of oracle calls to either the random example oracle or the MQ oracle. As explained in Section 2, for our results $\mathcal{C}(k)$ is an arbitrary subclass of k-juntas that is closed under permuting input variables, and $\mathcal{C}(k)^* \subset \mathcal{C}(k)$ is the subset of $\mathcal{C}(k)$ consisting of the functions whose set of relevant variables is contained in $\{1,\ldots,k\}$.

algorithm for k-juntas that makes $O(k\log k + k/\varepsilon)$ queries, and Sağlam (2018) subsequently established an $\Omega(k\log k)$ lower bound, showing that the algorithm of Blais (2009) is optimal up to constant factors for constant ε. Chen et al. (2025a) made the simple observation that if every nonzero function in a class $\mathcal{C}$ has at least $p2^n$ satisfying assignments, then relative-error testing to accuracy ε reduces to standard-model testing to accuracy $p\varepsilon$. Since every nonzero k-junta has $p \geq 2^{-k}$, combining this with the standard-model junta testing result of Blais (2009) implies that k-juntas can be relative-error tested with a query complexity of $O(2^k/\varepsilon)$. As their first open question, Chen et al. (2025a) asked whether there is a $\mathrm{poly}(k,1/\varepsilon)$-query relative-error tester for k-juntas.

Our first result answers this question in the affirmative:

Theorem 1 (Relative-error junta testing) *For $0 < \varepsilon < 1/2$, there is an algorithm for relative-error ε-testing of k-juntas with $\tilde{O}(k/\varepsilon)$ queries and samples.*

Our main result goes beyond juntas and deals with *subclasses* of juntas.[2] To state the result cleanly, the following terminology and notation is helpful: We say that a class $\mathcal{C}(k)$ of Boolean functions from $\{0,1\}^n$

2. We remind the reader that the testability/non-testability of a class $\mathcal{C}_1$ does not imply anything about the testability/non-testability of a class $\mathcal{C}_2 \subseteq \mathcal{C}_1$. This is true in both the standard model and in the relative-error testing model: for example, the class of all Boolean functions is trivially relative-error testable with zero queries, but the subclass of halfspaces is "hard" to test, requiring $\tilde{\Omega}(\log n)$ queries as shown by Chen et al. (2025b). In the other direction, conjunctions are a subclass of halfspaces, but while halfspaces are "hard" to test conjunctions are "easy," requiring only $O(1/\varepsilon)$ queries as shown by Chen et al. (2025c).)

to $\{0,1\}$ is a *subclass of k-juntas* if (i) every $f \in \mathcal{C}(k)$ is a k-junta, and (ii) $\mathcal{C}(k)$ is closed under permuting variables. We further write $\mathcal{C}(k)^*$ to denote the subset

$$\mathcal{C}(k)^* := \{f \in \mathcal{C}(k) : \ f \text{ does not depend on variables } x_{k+1}, \ldots, x_n\}$$

so we may think of $\mathcal{C}(k)^*$ as a class of functions over variables $x_1, \ldots, x_k$; note that consequently the cardinality $|\mathcal{C}(k)^*|$ is completely independent of n and depends only on k.

By combining techniques from junta testing with ideas from computational learning theory, Diakonikolas et al. (2007) showed that any subclass $\mathcal{C}(k)$ of k-juntas can be tested using only $\mathrm{poly}(\log(|\mathcal{C}(k)^*|), 1/\varepsilon)$ queries in the standard model. The precise polynomial bounds achieved by Diakonikolas et al. (2007) were subsequently improved by Chakraborty et al. (2011) and by Bshouty (2020). Moreover, Bshouty (2020) extended these result to the *distribution-free* testing model; we will describe this model, and the relevance of Bshouty's approach to our work, in Section 1.3.

Our second and main result shows that any $\mathcal{C}(k)$ can be efficiently tested in the relative-error model:

Theorem 2 (Relative-error testing of junta subclasses) *Let $\mathcal{C}(k)$ be any subclass of k-juntas. For $0 < \varepsilon < 1/2$, there is an algorithm for its relative-error ε-testing with $\tilde{O}\big(k \log |\mathcal{C}(k)^*|/\varepsilon\big)$ queries and samples.*

(We remark that since there are 2^{2^k} juntas over variables $x_1, \ldots, x_k$, Theorem 1 does not follow in a black-box way from Theorem 2; the proof of Theorem 1 requires a more refined analysis.)

Since any size-k decision tree is a k-junta, and likewise for size-k Boolean formulas and size-k branching programs, as a consequence of Theorem 2 via standard counting arguments given in Diakonikolas et al. (2007) we get the following relative-error analogues of standard-model testing results first by Diakonikolas et al. (2007) (and subsequently with sharper parameters by Chakraborty et al. (2011); Bshouty (2020)):

Corollary 3 *Let $\mathcal{C}(k)$ be the class of size-k decision trees, or the class of size-k Boolean formulas, or the class of size-k branching programs. For $0 < \varepsilon < 1/2$, there is an algorithm for relative-error ε-testing of $\mathcal{C}(k)$ that makes $\tilde{O}\big(k^2/\varepsilon\big)$ queries and samples.*

1.3. Techniques

In this subsection we focus our discussion on the more general and challenging problem of testing subclasses of juntas (Theorem 2). (An overview of Theorem 1 is given in Appendix C.) Our approach builds on techniques that were used to give efficient algorithms for testing juntas and subclasses of juntas in the *distribution-free* testing model. We recall that in the distribution-free testing model, as defined by Goldreich et al. (1998) and subsequently studied by many authors Halevy and Kushilevitz (2007, 2004); Glasner and Servedio (2009); Dolev and Ron (2011); Chen and Xie (2016); Liu et al. (2019); Bshouty (2019); Belovs (2019); Chen and Patel (2022); Chen et al. (2024b), there is an unknown and arbitrary "background distribution" $\mathcal{D}$ over $\{0,1\}^n$, and the distance between two functions f and g is measured by $\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. In this model the testing algorithm can both make black-box queries to the unknown f that is being tested, and can also draw i.i.d. samples from the distribution $\mathcal{D}$.

Remark 4 (On distribution-free testing versus relative-error testing.) *We stress that while a relative-error testing algorithim gets access to independent uniform samples from $f^{-1}(1)$, akin to how a distribution-free tester gets access to independent samples from $\mathcal{D}$, relative-error testing does* not *correspond to the special case of distribution-free testing in which the "background distribution" $\mathcal{D}$ is uniform over $f^{-1}(1)$. This is because in the distribution-free model, the distance $\mathbf{Pr}_{\boldsymbol{x}\sim\mathcal{D}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$ between two functions f, g is measured vis-a-vis the unknown distribution $\mathcal{D}$ which the testing algorithm can draw samples from. This is quite different from the way distance is measured in the relative-error setting; in particular, the relative*

distance $\text{rel-dist}(f, g)$ *is* not *equal to* $\mathbf{Pr}_{\boldsymbol{x} \sim \mathcal{U}_{f^{-1}(1)}}[f(\boldsymbol{x}) \neq g(\boldsymbol{x})]$. *Indeed, a function* g *can have large relative distance from* f *because* $g(x) = 1$ *on many points* x *for which* $f(x) = 0$*, but all such points have zero weight under the distribution* $\mathcal{U}_{f^{-1}(1)}$*. This issue necessitates a number of changes to the techniques used to give prior distribution-free testers, as described below.*

Improving on Liu et al. (2019), Bshouty (2019) gave a distribution-free testing algorithm that uses $\tilde{O}(k/\varepsilon)$ queries and samples, and soon thereafter Bshouty (2020), extending and strengthening the learning-based approach of Diakonikolas et al. (2007), gave an efficient distribution-free tester for any subclass of k-juntas. At a high level our approach is similar to that of Bshouty (2020), but there are several significant differences (most significant for the last two phases of our algorithm), which we describe in the high-level sketch of our approach given below.

- Like Bshouty (2020), the first two phases of our main JUNTA-SUBCLASS-TESTER algorithm (Algorithm 1) checks that certain restrictions of the function $f : \{0,1\}^n \to \{0,1\}$ that is being tested are close to literals in a suitable sense. This is done in our algorithm as follows: in the first phase ("Partition $[n]$ and find relevant blocks"), the algorithm randomly partitions the n input variables $x_1, \ldots, x_n$ into $O(k^2)$ disjoint blocks, and performs a binary search procedure over blocks to find (i) a collection of "relevant blocks," and (ii) for each "relevant block," a string (denoted v^ℓ) that "witnesses" that the block is relevant. (The whole algorithm rejects if this phase identifies more than k such blocks.) Next, the second phase ("Test the relevant sets") uses a uniform-distribution 1-junta testing algorithm UNIFORMJUNTA due to Blais (2008) (see Theorem 11 in Appendix A.2) to check that certain restrictions of f (denoted f^ℓ, and defined using the witness strings v^ℓ) are suitably close to literals under the uniform distribution.

  At a high level these phases of our algorithm are similar to Bshouty (2020), but the differences between the relative-error framework and the distribution-free framework mean we must carry out these phases in a somewhat different way. Bshouty (2020) creates restrictions and uses witness strings which, roughly speaking, fix variables which "appear to be irrelevant" to 0.[3] This approach works in the distribution-free context, since roughly speaking, the distance between two functions can only come from examples x that are "likely to be drawn under $\mathcal{D}$." In contrast, in the relative-error setting, a function g can be very far from f because g has many satisfying assignments x for which $f(x) = 0$, and such points will never be drawn when we sample from f's satisfying assignments. Because of this, we need to proceed in a different way than Bshouty (2020); as detailed in Section 2, our approach is to fix the "seemingly irrelevant" variables to *uniformly random* values.

- In the third phase ("Trimming APPROX"), our algorithm creates a set of candidate functions which are possible "approximators" (in a suitable sense) for the unknown function f. Recall from Table 1 that $\mathcal{C}(k)^* \subset \mathcal{C}(k)$ is the subset of $\mathcal{C}(k)$ consisting of the functions in $\mathcal{C}(k)$ whose set of relevant variables is contained in $\{1, \ldots, k\}$; our algorithm carries out the third phase by "trimming" an initial set APPROX which consists of a specially constructed set of juntas that each have small relative distance to some function in $\mathcal{C}(k)^*$. This "trimming" is done by checking, for each candidate function g in APPROX, whether randomly sampled positive examples of f also correspond (after a suitable remapping of variables) to positive examples of the candidate function g; if this does not hold for one of the sampled positive examples, then the candidate g is discarded. (The algorithm rejects if all candidates in APPROX are discarded.)

  In its fourth and final phase ("Find k-variable approximator for f"), our algorithm determines whether the trimmed approximator set contains a suitable approximator for f. This is done by (i) identifying

3. Bshouty (2020)'s results only apply to subclasses of juntas that are closed under variable projections and under restrictions that fix variables to 0.

the function $\mathfrak{g}$ in the trimmed approximator set which has the fewest satisfying assignments, and (ii) verifying that $\mathfrak{g}$ is indeed close to f in a suitable sense (under a remapping of the variables). If this is the case then our algorithm accepts, and otherwise it rejects.

The most significant differences between our algorithm and that of Bshouty (2020) are in Phases 3 and 4. Bshouty (2020) starts by checking that f is close to a particular subfunction of f, denoted F, which sets all variables in "irrelevant" blocks of f to 0 and sets all variables in each "relevant" block of f to the same bit. Bshouty (2020) then checks F against a set of candidates, each of which belongs to the subclass $\mathcal{C}(k)$. Because the Bshouty (2020) (distribution-free) notion of two functions h, h' being ε-far is that $h(\boldsymbol{x}) \neq h'(\boldsymbol{x})$ with probability at least ε over $\boldsymbol{x} \sim \mathcal{D}$, it is enough to simply sample independent draws from $\mathcal{D}$ and reject if f and F disagree on any of them. Similarly for the second check of F against the candidates, if F and a candidate disagree on a sampled point $\boldsymbol{x}$ then the candidate is eliminated.

In contrast, in our relative-error setting, since we can only draw samples from $f^{-1}(1)$, after we trim the approximator set in Phase 3, it could still be possible that f is far in relative distance from some surviving g in the trimmed set, which could happen if $g^{-1}(1) \cap f^{-1}(0)$ is large.

To deal with this, our approach is as follows: we do not use an analogue of Bshouty (2020)'s function F, but rather we consider *only* the function $\mathfrak{g}$ in the trimmed set that has the *fewest* satisfying assignments, and we work solely with this function in Phase 4. In the yes-case, when $f \in \mathcal{C}(k)$, we must have $|\mathfrak{g}^{-1}(1)| \approx |f^{-1}(1)|$, and since $\mathfrak{g}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \sim f^{-1}(1)$ with high probability, it must also be the case that $f(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \sim \mathfrak{g}^{-1}(1)$ with high probability. And in the no case, when $\text{rel-dist}(f, h)$ is large for every $h \in \mathcal{C}(k)$, we cannot have both $\mathfrak{g}(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \sim f^{-1}(1)$ with high probability and also $f(\boldsymbol{x}) = 1$ for $\boldsymbol{x} \sim \mathfrak{g}^{-1}(1)$ with high probability. For this would imply that $\text{rel-dist}(f, \mathfrak{g})$ is small, but since $\mathfrak{g}$ is very close in relative distance to some function in $\mathcal{C}(k)$, a simple triangle inequality implies that f has small relative distance to some function in $\mathcal{C}(k)$, which contradicts the fact that we are in the no-case.

1.4. Future work: relative-error testing of classes of functions that are approximated by juntas?

A natural goal for future work is to obtain positive results for testing classes of functions that do not correspond exactly to subclasses of juntas. In particular, in the standard uniform-distribution model, the techniques of Diakonikolas et al. (2007) go beyond just testing subclasses of juntas and allow testing classes of functions which can be *approximated* to high accuracy by juntas under the uniform distribution. Several classes of interest are of this sort, including: size-s Boolean circuits (where circuit size is measured by the number of unbounded fan-in AND/OR/NOT gates), s-term DNF formulas, s-sparse polynomials over $\mathbb{F}_2^n$, and others; each of these can be approximated, but not exactly computed, by juntas. Can classes such as these be tested efficiently in the relative-error setting?

A challenge which immediately arises is that while these classes are well-approximated by juntas under the uniform distribution, this is not the case under relative error. For example, for any constant s it is easy to give an s-term DNF formula which cannot be well-approximated in relative error by any $O_s(1)$-junta. On the positive side, though, for both the class $\mathcal{C} =$ conjunctions and the class $\mathcal{C} =$ decision lists, functions in $\mathcal{C}$ are approximable by $O(1)$-juntas under the uniform distribution and cannot in general be well-approximated by juntas under relative error, but Chen et al. (2025c) gave efficient testing algorithms for each of these classes. Each of the two testers of Chen et al. (2025c) was carefully specialized to the particular class at hand; it would be very interesting, though perhaps quite challenging, to give a general extension of Theorem 2 to classes of functions that are only uniform-distribution approximable by juntas. A first goal in this direction is to either give an $O_{s,\varepsilon}(1)$-query algorithm for testing the class of s-term DNF, or alternatively to prove an $\omega_n(1)$-query lower bound for constant ε.

1.5. Organization

Because of space constraints we give standard background and preliminaries in Appendix A. Section 2 gives our algorithm for relative-error testing of subclasses of k-juntas and an overview of its analysis. The full analysis of our junta subclass tester, completing the proof of Theorem 2 and Corollary 3, is given in Appendix B. A self-contained algorithm and analysis for the simpler problem of relative-error testing of k-juntas, proving Theorem 1, is given in Appendix C.

2. Testing Subclasses of Juntas

Recall that a class $\mathcal{C}(k)$ of Boolean functions from $\{0,1\}^n$ to $\{0,1\}$ is a *subclass of k-juntas* if (i) every $f \in \mathcal{C}(k)$ is a k-junta, and (ii) $\mathcal{C}(k)$ is closed under permuting variables (that is, if $f(x) \in \mathcal{C}(k)$ then for any permutation $\pi : [n] \to [n]$ we have that $f_\pi(x) = f(\pi(x))$ is also in $\mathcal{C}(k)$). Throughout this section, $\mathcal{C}(k)$ denotes a fixed and arbitrary subclass of k-juntas. (For example, $\mathcal{C}(k)$ might be the class of decision trees over $x_1, \ldots, x_n$ with at most k nodes). We further recall that $\mathcal{C}(k)^* \subset \mathcal{C}(k)$ is the subset of $\mathcal{C}(k)$ consisting of the functions in $\mathcal{C}(k)$ that do not depend on variables $x_{k+1}, \ldots, x_n$. (Continuing the example from above, $\mathcal{C}(k)^*$ would be the class of size-k decision trees over variables $x_1, \ldots, x_k$.)

2.1. Notation

Let $h \leq k$ be a positive integer. Given an injective map $\sigma : [h] \to [n]$ and an assignment $y \in \{0,1\}^n$, we write $\sigma^{-1}(y)$ to denote the string $y \in \{0,1\}^h$ with $y_i = x_{\sigma(i)}$ for each $i \in [h]$. Given a $z \in \{0,1\}^h$, we write $\sigma(z)$ to denote the string $u \in \{0,1\}^S$, where $S = \{\sigma(i) : i \in [h]\}$ is the image set of σ and $u_{\sigma(i)} = z_i$ for each $i \in [h]$. Given a function $g : \{0,1\}^h \to \{0,1\}$ over h variables and an injective map $\sigma : [h] \to [n]$, we write g_σ to denote the Boolean function over $\{0,1\}^n$ with $g_\sigma(x) = g(\sigma^{-1}(x))$ (note that g_σ is an h-junta and hence also a k-junta). We will always use f, f', etc. to denote a Boolean function over $\{0,1\}^n$ and $g, g', \mathfrak{g}$, etc. to denote a Boolean function over $\{0,1\}^h$ for some $h \leq k$. With this notation, note that $\text{rel-dist}(f, g_\sigma)$ and $\text{rel-dist}(g_\sigma, f)$ are well defined, since f and g_σ are Boolean functions on $\{0,1\}^n$.

Given a positive integer $h \leq k$ and $0 < \kappa \leq 1/2$, we now define a collection of functions over $\{0,1\}^h$ called $\text{APPROX}(h, \kappa)$. The idea is that $\text{APPROX}(h, \kappa)$ contains functions over h variables that approximate functions in $\mathcal{C}(k)$ with error parameter κ, in a suitable sense that we define as follows:

> For each function $f \in \mathcal{C}(k)^*$, let $f' : \{0,1\}^n \to \{0,1\}$ be the function defined as follows:
>
> $$f'(x) := \arg\max_{b \in \{0,1\}} \left\{ \mathop{\mathbf{Pr}}_{\boldsymbol{w} \sim \{0,1\}^{n-h}} \left[f(x_{[h]} \circ \boldsymbol{w}) = b \right] \right\}. \tag{1}$$
>
> Clearly f' is an h-junta that only depends on the first h variables. Let $g : \{0,1\}^h \to \{0,1\}$ be the function with $g(z) = f'(z \circ 0^{n-h})$. The function g is in $\text{APPROX}(h, \kappa)$ iff $\text{rel-dist}(f, f') \leq \kappa$.

Note that a testing algorithm can construct the set $\text{APPROX}(h, \kappa)$ "on its own" without making any queries. At a high level, the reason we need to introduce $\text{APPROX}(h, \kappa)$, instead of working with $\mathcal{C}(k)^*$ directly, is because the algorithm may only be able to identify, implicitly, an h-subset of the k relevant variables.

The fact below follows directly from the definition of $\text{APPROX}(h, \kappa)$:

Fact 5 $|\text{APPROX}(h, \kappa)| \leq |\mathcal{C}(k)^*|$.

The next lemma shows that if $f : \{0,1\}^n \to \{0,1\}$ is close to some $g \in \text{APPROX}(h, \kappa)$ in relative distance under some injective map $\sigma : [h] \to [n]$, then it must be close to $\mathcal{C}(k)$ in relative distance as well.

Lemma 6 *If $f : \{0,1\}^n \to \{0,1\}$ satisfies* rel-dist$(f, g_\sigma) \leq 1$ *for some function $g \in$* APPROX(h, κ) *and injective map $\sigma : [h] \to [n]$, then we have* rel-dist$(f, \mathcal{C}(k)) \leq$ rel-dist$(f, g_\sigma) + 4\kappa$.

Proof By the construction of APPROX(h, κ), given that $g \in$ APPROX(h, κ), there exists a function $f' \in \mathcal{C}(k)$ that satisfies rel-dist$(f', g_\sigma) \leq \kappa$. Using Lemma 9 (in Appendix A.2) which shows that relative distance satisfies approximate symmetry, we have that rel-dist$(g_\sigma, f') \leq 2\kappa$. The lemma follows by bounding rel-dist(f, f') using the triangle inequality (Lemma 10 in Appendix A.2). ■

2.2. Overview of the Main Algorithm

We start with an overview of our main algorithm, Algorithm 1, for testing an arbitrary subclass $\mathcal{C}(k)$ of k-juntas that is closed under permutations. At a very high level, the approach is to perform "implicit learning" by searching a candidate set of hypotheses to see whether it contains a hypothesis that is consistent with the algorithm's data.

Let $f : \{0,1\}^n \to \{0,1\}$ be the function that is being tested. Algorithm 1 starts by drawing a partition of variables $[n]$ into $r = O(k^2)$ many blocks $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r$ uniformly at random. For the case when $f \in \mathcal{C}(k)$, because f only depends on k variables, a quick calculation shows that most likely every block $\boldsymbol{X}_\ell$ contains at most one relevant variable of f. For the overview we will assume that this is the case when $f \in \mathcal{C}(k)$.

In Phase 1, Algorithm 1 tries to find as many *relevant* blocks $\boldsymbol{X}_\ell$ of f as possible under a query complexity budget. Here intuitively we say $\boldsymbol{X}_\ell$ is a relevant block of f if the algorithm has found a string $v^\ell \in \{0,1\}^n$ such that $f^\ell : \{0,1\}^{\boldsymbol{X}_\ell} \to \{0,1\}$, defined as

$$f^\ell(x) := f\left(v^\ell_{\overline{\boldsymbol{X}_\ell}} \circ x\right), \tag{2}$$

is not a constant function. To search for a new relevant block, let $\{\boldsymbol{X}_\ell\}_{\ell \in \boldsymbol{I}}$ denote the set of relevant blocks found so far and $\boldsymbol{X}$ be their union (with $\boldsymbol{I} = \emptyset$ and $\boldsymbol{X} = \emptyset$ initially). Phase 1 draws a uniform satisfying assignment $\boldsymbol{u} \sim f^{-1}(1)$ and a random $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$; if $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) = 0$, then a binary search over blocks can be run on $\boldsymbol{u}$ and $\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}$, using $O(\log r) = O(\log k)$ many queries, to find a new relevant block $\boldsymbol{X}^\ell$ together with an accompanying v^ℓ (which is used to define f^ℓ). If Phase 1 finds too many relevant blocks (i.e., $|\boldsymbol{I}|$ becomes larger than k), then Algorithm 1 rejects; on the other hand, if it fails to make progress after $T_2 = O(\log k) \cdot T_1$ many rounds, where

$$T_1 = O\left(\frac{\log |\mathcal{C}(k)^*|}{\varepsilon}\right),$$

then Algorithm 1 stops searching and moves on to Phase 2. The latter helps ensure that the number of queries used in Phase 2 stays within the budget.

In Phase 2, Algorithm 1 checks if every f^ℓ, $\ell \in \boldsymbol{I}$, is $(1/30)$-close to a literal under the uniform distribution (i.e., x_i or $\overline{x_i}$ for some variable $i \in \boldsymbol{X}_\ell$), and rejects if any of them is not.

When Algorithm 1 reaches Phase 3, one may assume that the relevant blocks found satisfy the following two conditions (recall $\boldsymbol{X}$ is the union of relevant blocks $\boldsymbol{X}_\ell$, $\ell \in \boldsymbol{I}$, found in Phase 1):

1. First, we have
$$\Pr_{\substack{\boldsymbol{u} \sim f^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}}} \left[f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) = 0\right] \leq \kappa/4, \tag{3}$$
where κ is chosen to be $1/20T_1$, since otherwise it is unlikely for Phase 2 to fail to find a new relevant block after $T_2 = O(\log k) \cdot T_1$ repetitions before moving to Phase 3.

2. Second, for every $\ell \in \boldsymbol{I}$, f^ℓ is $(1/30)$-close to a literal $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ for some $\tau(\ell) \in \boldsymbol{X}_\ell$; since otherwise Phase 2 would reject with high probability. When this is the case, $\tau(\ell) \in \boldsymbol{X}_\ell$ for each $\ell \in \boldsymbol{I}$ is well defined and unique.

To give some intuition for Phase 3 and 4, let's consider the case when $f \in \mathcal{C}(k)$. Let $h = |\boldsymbol{I}|$ and let $\ell_1 < \ldots < \ell_h$ be indices of blocks in $\boldsymbol{I}$. Let $\sigma : [h] \to [n]$ be the injective map with $\sigma(i) = \tau(\ell_i)$ for each $i \in [h]$, let $S = \{\tau(\ell_i) : i \in [h]\}$, and let $\mathsf{G} : \{0,1\}^h \to \{0,1\}$ be the following function:

$$\mathsf{G}(x) := \arg\max_{b \in \{0,1\}} \left\{ \mathop{\mathbf{Pr}}_{\boldsymbol{w} \sim \{0,1\}^{\overline{S}}} \left[f(\sigma(x) \circ \boldsymbol{w}) = b \right] \right\}.$$

Lemma 15 shows that Equation (3) implies $\text{rel-dist}(f, \mathsf{G}_\sigma) \leq \kappa$ and thus, we have $\mathsf{G} \in \textsc{Approx}(h, \kappa)$ by the construction of $\textsc{Approx}(h, \kappa)$. On the other hand, if f is ε-far from every function in $\mathcal{C}(k)$ in relative distance, then by Lemma 6, every function g in $\textsc{Approx}(h, \kappa)$ has $\text{rel-dist}(f, g_\sigma) > \varepsilon/2$ given that $\kappa < \varepsilon/8$. The goal of Phase 3 and 4 is then to distinguish the following two cases: (i) there exists a $\mathsf{G} \in \textsc{Approx}(h, \kappa)$ such that $\text{rel-dist}(f, \mathsf{G}_\sigma) \leq \kappa$, versus (ii) every $g \in \textsc{Approx}(h, \kappa)$ has $\text{rel-dist}(f, g_\sigma) \geq \varepsilon/2$ (note that there is a big gap between the two distance parameters κ and ε).

To gain some intuition, pretend for a moment that $\textsc{Approx}(h, \kappa)$ is a collection of functions over $\{0,1\}^n$, and the two cases we need to distinguish are simply either (1) there exists a function $\mathsf{G} \in \textsc{Approx}(h, \kappa)$ such that $\text{rel-dist}(f, \mathsf{G}) \leq \kappa$ or (2) every $g \in \textsc{Approx}(h, \kappa)$ satisfies $\text{rel-dist}(f, g) \geq \varepsilon/2$. This could be done easily as follows:

1. Draw $1/(20\kappa)$ satisfying assignments $\boldsymbol{u} \sim f^{-1}(1)$ and remove every $g \in \textsc{Approx}(h, \kappa)$ with $g(\boldsymbol{u}) = 0$ for some sampled $\boldsymbol{u}$. Reject if no function is left in $\textsc{Approx}(h, \kappa)$; otherwise, let $\mathfrak{g}$ be the function left with the smallest $\mathfrak{g}^{-1}(1)$ and pass it to the next phase.

   If we are in case (1), then with probability at least $19/20$ the function G survives so the algorithm does not reject. On the other hand, in both cases (1) and (2), by a union bound every g that survives must satisfy

   $$\frac{|f^{-1}(1) \setminus g^{-1}(1)|}{|f^{-1}(1)|} \leq \varepsilon/500.$$

   In particular the function $\mathfrak{g}$ passed down to the next phase must satisfy this condition in both cases. For case (1), one can use the choice of $\mathfrak{g}$ (i.e., $|\mathfrak{g}^{-1}(1)|$ is no larger than $|(\mathsf{G})^{-1}(1)|$) to show that

   $$\frac{|\mathfrak{g}^{-1}(1) \setminus f^{-1}(1)|}{|\mathfrak{g}^{-1}(1)|} \leq \varepsilon/400. \tag{4}$$

   For case (2), using $\text{rel-dist}(f, \mathfrak{g}) \geq \varepsilon/2$, one can show that the ratio above is at least $\varepsilon/7$.

2. Draw $20/\varepsilon$ samples $\boldsymbol{u} \sim \mathfrak{g}^{-1}(1)$; accept if $f(\boldsymbol{u}) = 1$ for all $\boldsymbol{u}$ sampled and reject otherwise.

   Given Equation (4), in case (1) we will accept with probability at least $19/20$. On the other hand, in case (2), given that the ratio in Equation (4) is at least $\varepsilon/7$ we will reject with probability at least

   $$1 - (1 - \varepsilon/7)^{20/\varepsilon} \geq 1 - e^{-20/7} \geq 0.9.$$

The discussion above for the simplified situation is exactly what Phase 3 and Phase 4 in Algorithm 1 aim to achieve, except that in the real situation, $\textsc{Approx}(h, \kappa)$ consists of functions over $\{0,1\}^h$ and we don't have direct access to the "hidden" injective map σ that connects f and g. In Phase 3, Algorithm 1 mimics step 1 above as follows: draw $\boldsymbol{u} \sim f^{-1}(1)$, use it to *obtain* $\sigma^{-1}(\boldsymbol{u}) \in \{0,1\}^h$, and remove those g with $g(\sigma^{-1}(\boldsymbol{u})) = 0$. (This is because $g_\sigma(\boldsymbol{u}) = g(\sigma^{-1}(\boldsymbol{u}))$.) Recall that $\mathfrak{g} \in \textsc{Approx}(h, \kappa)$ is the function

with the smallest $|\mathfrak{g}^{-1}(1)|$ that remains (recall the discussion in the final paragraph of Section 1.3). Phase 4 draws $\boldsymbol{z} \sim \mathfrak{g}^{-1}(1)$, uses it to *obtain* $\boldsymbol{u} = \sigma(\boldsymbol{z}) \circ \boldsymbol{w}$ with $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$, where $S := \{\sigma(1), \ldots, \sigma(h)\}$, and rejects if $f(\sigma(\boldsymbol{z}) \circ \boldsymbol{w}) = 0$. (This is because $\boldsymbol{u}$ obtained this way is distributed exactly the same as drawing uniformly from $\mathfrak{g}_\sigma^{-1}(1)$.)

The challenges are (1) how to obtain $\sigma^{-1}(u)$ given $u \in \{0,1\}^n$ and (2) how to obtain $\sigma(z) \circ \boldsymbol{w}$, $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$, given $z \in \{0,1\}^h$. Both of them rely on a simple subroutine called **RelVarValue**: Given a Boolean function ψ which is promised to be $(1/30)$-close to an (unknown) literal under the uniform distribution and any assignment, **RelVarValue** can reveal the value of the literal variable in that assignment with high probability. Given **RelVarValue**, obtaining $\sigma^{-1}(u)$ from $u \in \{0,1\}^n$ is easy: just run **RelVarValue** on f^ℓ and $u_{\boldsymbol{X}_\ell}$ for each $\ell \in I$.

Obtaining $\sigma(z) \circ \boldsymbol{w}$ from z is more involved. Given a $y \in \{0,1\}^n$ and a $z \in \{0,1\}^h$, we write $y_{\sigma,z}$ to denote the following n-bit string: For each block $\boldsymbol{X}_\ell$ with $\ell \notin \boldsymbol{I}$, $y_{\sigma,z}$ agrees with y in $\boldsymbol{X}_\ell$; for each $\ell_i \in \boldsymbol{I}$,

$$\text{the } \boldsymbol{X}_{\ell_i} \text{ block of } y_{\sigma,z} = \begin{cases} y_{\boldsymbol{X}_{\ell_i}} & \text{if } y_{\tau(\ell_i)} = z_i \\ \overline{y_{\boldsymbol{X}_{\ell_i}}} & \text{if } y_{\tau(\ell_i)} \neq z_i \end{cases}$$

A key observation is that, when $\boldsymbol{y} \sim \{0,1\}^n$, $\boldsymbol{y}_{\sigma,z}$ is distributed exactly the same as $\sigma(z) \circ \boldsymbol{w}$ with $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$. The MAPBACK subroutine, which calls on FINDVARVALUE, takes y and z as inputs and outputs $y_{\sigma,z}$ with high probability.

The rest of the argument for testing subclasses of juntas is organized as follows. We first focus on the two subroutines FINDVARVALUE and MAPBACK and analyze their performance guarantees in Section 2.3. In Appendix B.1 we show that when $f \in \mathcal{C}(k)$, Algorithm 1 accepts with probability at least $2/3$; in Appendix B.2, we show that when f is ε-far from $\mathcal{C}(k)$ in relative distance, Algorithm 1 rejects with probability at least $2/3$. Finally we bound the query complexity of Algorithm 1 in Appendix B.3.

2.3. The FINDVARVALUE and MAPBACK subroutines

The FINDVARVALUE subroutine takes three inputs: a Boolean function $\psi : \{0,1\}^m \to \{0,1\}$, an assignment $w \in \{0,1\}^m$, and an error parameter $\delta > 0$. When ψ is promised to be $(1/30)$-close to a literal (say x_i or $\overline{x_i}$) under the uniform distribution, FINDVARVALUE returns the value w_i of the literal variable in w with probability at least $1 - \delta$.

Lemma 7 FINDVARVALUE(ψ, w, δ) *makes* $O(\log(1/\delta))$ *queries and satisfies the following conditions:*

1. *If* $\psi : \{0,1\}^m \to \{0,1\}$ *is* $(1/30)$*-close to a literal* x_i *or* $\overline{x_i}$ *under the uniform distribution, then with probability at least* $1 - \delta$, FINDVARVALUE *returns the bit* w_i.
2. *If* ψ *is a literal* x_i *or* $\overline{x_i}$, *then* FINDVARVALUE *always returns* w_i.

Proof The case when ψ is a literal is simple as the correct counter always gets incremented.

Consider the case when ψ is $(1/30)$-close to either x_i or $\overline{x_i}$ with $w_i = 0$ (meaning $i \in Y_0$). Then

$$\mathbf{Pr}\left[\psi(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) = \psi(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1)\right] \leq \mathbf{Pr}\left[\psi(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq \boldsymbol{b}_i^0\right] + \mathbf{Pr}\left[\psi(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1) \neq \overline{\boldsymbol{b}_i^0}\right] \leq 1/15,$$

where the last inequality follows from the fact that both the marginal distributions of $\boldsymbol{b}^0 \circ \boldsymbol{b}^1$ and $\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1$ are uniform. As a result, in each round of FINDVARVALUE, $\boldsymbol{G}_0$ goes up by 1 with probability at least $14/15$ and $\boldsymbol{G}_1$ goes up by 1 with probability at most $1/15$. The claim then follows from a standard Chernoff bound in this case, by making the hidden constant large enough. The case when $w_i = 1$ and $i \in Y_1$ can be proved similarly, using

$$\mathbf{Pr}\left[\psi(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq \psi(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1)\right] \leq \mathbf{Pr}\left[\psi(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq \boldsymbol{b}_i^1\right] + \mathbf{Pr}\left[\psi(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1) \neq \boldsymbol{b}_i^1\right] \leq 1/15.$$

Algorithm 1: JUNTA-SUBCLASS-TESTER(f, ε)

Phase 1: Partition $[n]$ and find relevant blocks

Parameters used in the algorithm:

$$r = 20k^2, \quad T_1 = O\left(\frac{\log|\mathcal{C}(k)^*|}{\varepsilon}\right), \quad T_2 = 100\log(20k+1)\cdot T_1 \quad \text{and} \quad \kappa = \frac{1}{20T_1}.$$

Randomly partition $[n]$ into r blocks $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r$; Set $\boldsymbol{X} = \emptyset$, $\boldsymbol{I} = \emptyset$ and $t = 0$;
while $t \neq T_2$ **do**
  Draw $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$; set $t \leftarrow t+1$;
  if $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) \neq f(\boldsymbol{u})$ **then**
    Binary Search over blocks of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r$ that are subsets of $\overline{\boldsymbol{X}}$ to find a new relevant block $\boldsymbol{X}_\ell$ and a string $v^\ell \in \{0,1\}^n$ such that $f(v^\ell) \neq f\left(v^\ell_{\overline{\boldsymbol{X}_\ell}} \circ \boldsymbol{w}_{\boldsymbol{X}_\ell}\right)$.
    Denote by $f^\ell : \{0,1\}^{\boldsymbol{X}_\ell} \to \{0,1\}$ the following function:

$$f^\ell(x) := f\left(v^\ell_{\overline{\boldsymbol{X}_\ell}} \circ x\right).$$

    Set $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \boldsymbol{X}_\ell$, $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{\ell\}$, $t \leftarrow 0$, and reject if $|\boldsymbol{I}| > k$;
  end
end

Phase 2: Test the relevant sets

foreach $\ell \in \boldsymbol{I}$ **do**
  Run UNIFORMJUNTA$(f^\ell, 1, 1/15, 1/30)$; reject if it rejects;
  Draw $\boldsymbol{b} \sim \{0,1\}^{\boldsymbol{X}_\ell}$; Reject if $f^\ell(\boldsymbol{b}) = f^\ell(\overline{\boldsymbol{b}})$;
end

Phase 3: Trimming APPROX

Set $\boldsymbol{A} \leftarrow$ APPROX(h, κ), where $h = |\boldsymbol{I}|$ and $\boldsymbol{I} = \{\ell_1, \ldots, \ell_h\}$ with $\ell_1 < \ldots < \ell_h$;
repeat T_1 **times**
  Draw $\boldsymbol{u} \sim f^{-1}(1)$ and let $\boldsymbol{v}$ be the all-0 string in $\{0,1\}^h$;
  for $i = 1$ **to** h **do**
    Set $\boldsymbol{v}_i \leftarrow$ FINDVARVALUE$(f^{\ell_i}, \boldsymbol{u}_{\boldsymbol{X}_{\ell_i}}, 1/(20k))$;
  end
  Remove from $\boldsymbol{A}$ every $g \in \boldsymbol{A}$ with $g(\boldsymbol{v}) = 0$;
end
reject if $\boldsymbol{A} = \emptyset$; otherwise let $\mathfrak{g}$ be the function in $\boldsymbol{A}$ with the smallest $|\mathfrak{g}^{-1}(1)|$;

Phase 4: Find k-variable approximator for f

repeat $20/\varepsilon$ **times**
  Draw $\boldsymbol{y} \sim \{0,1\}^n$ and $\boldsymbol{z} \sim \mathfrak{g}^{-1}(1)$ (note that $\boldsymbol{z} \in \{0,1\}^h$);
  Let $\boldsymbol{u} =$ MAPBACK$(f, \boldsymbol{I}, \{\boldsymbol{X}_\ell : \ell \in \boldsymbol{I}\}, \{v^\ell : \ell \in \boldsymbol{I}\}, \boldsymbol{y}, \boldsymbol{z})$;
  reject if $f(\boldsymbol{u}) = 0$;
end
Accept;

Algorithm 2: FINDVARVALUE(ψ, w, δ)

Let $Y_\zeta = \{j : w_j = \zeta\}$ for both $\zeta \in \{0,1\}$;
Set $\boldsymbol{G}_0 = \boldsymbol{G}_1 = 0$;
repeat $O(\log(1/\delta))$ **times**
  Draw $\mathbf{b}^\zeta \sim \{0,1\}^{Y_\zeta}$ for both $\zeta \in \{0,1\}$;
  If $\psi(\mathbf{b}^0 \circ \boldsymbol{b}^1) \neq \psi(\overline{\mathbf{b}^0} \circ \boldsymbol{b}^1)$ then set $\boldsymbol{G}_0 \leftarrow \boldsymbol{G}_0 + 1$; otherwise, set $\boldsymbol{G}_1 \leftarrow \boldsymbol{G}_1 + 1$;
  Return 0 if $\boldsymbol{G}_0 > \boldsymbol{G}_1$ and return 1 otherwise;
end

Algorithm 3: MAPBACK$(f, I, \{X_\ell : \ell \in I\}, \{v^\ell : \ell \in I\}, y, z)$

foreach $i \in [h]$ **do**
  Let $b_i =$ FINDVARVALUE$(f^{\ell_i}, y_{X_{\ell_i}}, 1/(20k))$;
  Flip the X_{ℓ_i}-block of y to $\overline{y_{X_{\ell_i}}}$ if $b_i \neq z_i$;
end
Return y;

This finishes the proof of the lemma. ■

Next we work on the subroutine MAPBACK used in Phase 4 of Algorithm 1, which makes calls to FINDVARVALUE. It takes as input the function f; the set I; for each $\ell \in I$ a collection of $h \leq k$ blocks X_ℓ together with a string $v^\ell \in \{0,1\}^n$ such that the function f^ℓ over $\{0,1\}^{X_\ell}$ defined using v^ℓ in Equation (2) is $(1/30)$-close to a literal $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ under the uniform distribution; and two strings $y \in \{0,1\}^n$ and $z \in \{0,1\}^h$. Let $\ell_1 < \cdots < \ell_h$ be the indices in I, and let σ and S be defined using τ as in Section 2.2.

Lemma 8 MAPBACK *uses $O(k \log k)$ queries and satisfies the following conditions:*

1. *If f^ℓ is $(1/30)$-close to a literal $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ for all $\ell \in I$ under the uniform distribution, then with probability at least $1 - 1/20$,* MAPBACK *returns $y_{\sigma,z}$.*
2. *If f^ℓ is a literal for all $\ell \in I$, then* MAPBACK *always returns $y_{\sigma,z}$.*

Proof The query complexity follows from the query complexity of FINDVARVALUE, and our choice of $\delta = 1/(20k)$. The probability follows from a union bound over the $h \leq k$ calls to FINDVARVALUE. ■

References

Noga Alon. Lower bound on active testing dimension of juntas. Personal communication to authors of Balcan et al. (2012) (see p. 3 of Balcan et al. (2012)), 2012.

Andris Ambainis, Aleksandrs Belovs, Oded Regev, and Ronald de Wolf. Efficient quantum algorithms for (gapped) group testing and junta testing. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 903–922, 2016.

A. Atıcı and R. Servedio. Quantum algorithms for testing and learning juntas. *Quantum Information Processing*, 6(5):323–348, 2007.

Maria-Florina Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active Property Testing. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 21–30, 2012.

Aleksandrs Belovs. Quantum algorithm for distribution-free junta testing. In *Computer Science–Theory and Applications: 14th International Computer Science Symposium in Russia, CSR 2019*, pages 50–59. Springer, 2019.

Eric Blais. Improved bounds for testing juntas. In *Proc. RANDOM*, pages 317–330, 2008.

Eric Blais. Testing juntas nearly optimally. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 2009.

A. Blum. Relevant examples and relevant features: Thoughts from computational learning theory. in AAAI Fall Symposium on 'Relevance', 1994.

Avrim Blum. Learning a function of r relevant variables. In Bernhard Schölkopf and Manfred K. Warmuth, editors, *Proc. 16th Annual Conference on Learning Theory (COLT)*, volume 2777 of *Lecture Notes in Computer Science*, pages 731–733. Springer-Verlag, 2003.

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.

J. Bourgain. On the distributions of the Fourier spectrum of Boolean functions. *Israel J. Math.*, 131:269–276, 2002.

Nader H. Bshouty. Almost optimal distribution-free junta testing. In *34th Computational Complexity Conference, CCC*, pages 2:1–2:13, 2019.

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.

Harry Buhrman, David García-Soriano, Arie Matsliah, and Ronald de Wolf. The non-adaptive query complexity of testing k-parities. *Chic. J. Theor. Comput. Sci.*, 2013, 2013. URL http://cjtcs.cs.uchicago.edu/articles/2013/6/contents.html.

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.

Thomas Chen, Shivam Nadimpalli, and Henry Yuen. Testing and learning quantum juntas nearly optimally. In *Proceedings of the 2023 ACM-SIAM Symposium on Discrete Algorithms, SODA 2023*, pages 1163–1185, 2023.

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)*, 2025a.

X. Chen, A. De, Y. Huang, S. Nadimpalli, R. A. Servedio, and T. Yang. Testing halfspaces with relative error. Manuscript, 2025b.

X. Chen, W. Pires, T. Pitassi, and R. A. Servedio. Relative-error testing of conjunctions and decision lists. Manuscript, 2025c.

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.

Xi Chen and Shyamal Patel. New lower bounds for adaptive tolerant junta testing. *FOCS*, pages 1778–1786, 2023.

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.

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.

Xi Chen, Rocco A. Servedio, Li-Yang Tan, Erik Waingarten, and Jinyu Xie. Settling the query complexity of non-adaptive junta testing. *J. ACM*, 65(6):40:1–40:18, 2018. doi: 10.1145/3213772. URL https://doi.org/10.1145/3213772.

Xi Chen, Anindya De, Yuhao Li, Shivam Nadimpalli, and Rocco A. Servedio. Mildly exponential lower bounds on tolerant testers for monotonicity, unateness, and juntas. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA 2024*, pages 4321–4337, 2024a.

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, 2024b.

H. Chockler and D. Gutfreund. A lower bound for testing juntas. *Information Processing Letters*, 90(6): 301–305, 2004.

Anindya De, Elchanan Mossel, and Joe Neeman. Junta correlation is testable. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 1549–1563. IEEE Computer Society, 2019.

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.

Elya Dolev and Dana Ron. Distribution-free testing for monomials with a sublinear number of queries. *Theory of Computing*, 7(11):155–176, 2011.

Y. Filmus. A simple proof of the Kindler–Safra theorem. Manuscript, available at https://yuvalfilmus.cs.technion.ac.il/Manuscripts/KindlerSafra.pdf, 2004.

E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *J. Computer & System Sciences*, 68(4):753–787, 2004.

E. Friedgut. Boolean functions with low average sensitivity depend on few coordinates. *Combinatorica*, 18 (1):474–483, 1998.

Dana Glasner and Rocco A. Servedio. Distribution-free testing lower bound for basic boolean functions. *Theory of Computing*, 5(1):191–216, 2009.

O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002.

Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45:653–750, 1998.

S. Halevy and E. Kushilevitz. Distribution-Free Connectivity Testing for Sparse Graphs. *Algorithmica*, 51 (1):24–48, 2004.

S. Halevy and E. Kushilevitz. Distribution-Free Property Testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007.

Vishnu Iyer, Avishay Tal, and Michael Whitmeyer. Junta distance approximation with sub-exponential queries. In Valentine Kabanets, editor, *36th Computational Complexity Conference, CCC 2021, July 20-23, 2021, Toronto, Ontario, Canada (Virtual Conference)*, volume 200 of *LIPIcs*, pages 24:1–24:38. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM J. Comput.*, 47(6):2238–2276, 2018.

G. Kindler and S. Safra. Noise resistant boolean functions are juntas. Manuscript, 2004.

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.

Zhengyang Liu, Xi Chen, Rocco A. Servedio, Ying Sheng, and Jinyu Xie. Distribution-free junta testing. *ACM Trans. Algorithms*, 15(1):1:1–1:23, 2019.

K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio. Testing halfspaces. *SIAM J. on Comput.*, 39(5): 2004–2047, 2010a.

Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010b.

E. Mossell, R. O'Donnell, and R. Servedio. Learning juntas. *Proceedings of the Thirty-Fifth Annual Symposium on Theory of Computing*, 2003.

Shivam Nadimpalli and Shyamal Patel. Optimal non-adaptive tolerant junta testing via local estimators. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 1039–1050, 2024.

N. Nisan and M. Szegedy. On the degree of Boolean functions as real polynomials. In *Proc. Twenty-Fourth Annual Symposium on Theory of Computing*, pages 462–467, 1992.

Ramesh Krishnan S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of boolean functions. *Random Struct. Algorithms*, 60(2):233–260, 2022.

M. Parnas, D. Ron, and A. Samorodnitsky. Testing Basic Boolean Formulae. *SIAM J. Disc. Math.*, 16: 20–46, 2002. URL citeseer.ifi.unizh.ch/parnas02testing.html.

Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Struct. Algorithms*, 20(2):165–183, 2002.

R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25:252–271, 1996.

Mert Sağlam. Near log-convexity of measured heat in (discrete) time and consequences. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 967–978. IEEE Computer Society, 2018. doi: 10.1109/FOCS.2018.00095. URL https://doi.org/10.1109/FOCS.2018.00095.

R.A. Servedio, L.-Y. Tan, and J. Wright. Adaptivity helps for testing juntas. In *Proceedings of the 30th IEEE Conference on Computational Complexity*, pages 264–279, 2015.

Gregory Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13:1–13:45, 2015.

Appendix A. Background and Preliminaries

A.1. Basic notation and terminology

We use $[n]$ to denote the set $\{1, \dots, n\}$. Given a set $S \subseteq [n]$, we denote by $\overline{S}$ the set $[n] \setminus S$.

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. Given $z \in \{0,1\}^{[n]}$ we denote by $\overline{z}$ the string with each bit of z flipped, so $\overline{z}_i = 1 - z_i$ for every $i \in [n]$. Given strings $x, y \in \{0,1\}^n$, we denote by $x \oplus y$ the bit-wise xor of x and y, that is $(x \oplus y)_i = x_i \oplus y_i$. Given a permutation $\pi : [n] \to [n]$ and $x \in \{0,1\}^n$, we denote by $\pi(x)$ the string $x_{\pi(1)} x_{\pi(2)} \dots x_{\pi(n)}$. Given $f : \{0,1\}^n \to \{0,1\}$, we denote by f_π the function $f_\pi(x) = f(\pi(x))$.

For $J \subseteq [n]$ we say that a function $f : \{0,1\}^n \to \{0,1\}$ is a *J-junta* if f depends only on the variables in J, i.e. $f(x_J \circ z_{\overline{J}}) = f(x_J \circ z'_{\overline{J}})$ for every $x, z, z' \in \{0,1\}^n$. If f is a J-junta for some set J of size k, we say that f is a *k-junta*.

A.2. Standard-model property testing and relative-error property testing

We briefly review the "standard" model for testing Boolean functions Goldreich (2017). A standard-model testing algorithm for a class $\mathcal{C}$ of n-variable Boolean functions has oracle access $\mathrm{MQ}(f)$ to an unknown and arbitrary function $f : \{0,1\}^n \to \{0,1\}$. The requirement on the algorithm is that it 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 f disagrees with every function $g \in \mathcal{C}$ on at least $\varepsilon 2^n$ inputs in $\{0,1\}^n$.

As defined and studied in Chen et al. (2025a,b,c), a *relative-error* testing algorithm for $\mathcal{C}$ has oracle access to $\mathrm{MQ}(f)$ and also has access to a "random satisfying assignment" oracle which, when called, returns a uniform random element $\boldsymbol{x} \sim f^{-1}(1)$. An ε-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

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

Thus, in the relative-error setting, the distance from f to $\mathcal{C}$ is measured "relative to the sparsity of f."

The following easy observation says that relative distance doesn't change when we apply a permutation:

Observation 1 *If $\pi : [n] \to [n]$ is a permutation, then* $\text{rel-dist}(f, g) = \text{rel-dist}(f_\pi, g_\pi)$.

We will need the following two easy lemmas about relative distance:

Lemma 9 (Approximate symmetry of relative distance) *Let $f, g : \{0,1\}^n \to \{0,1\}$ be functions such that* $\text{rel-dist}(f, g) \leq \varepsilon$ *where $\varepsilon \leq 1/2$. Then* $\text{rel-dist}(g, f) \leq 2\varepsilon$.

Proof Since $\text{rel-dist}(f, g) \leq \varepsilon$ we have that

$$\frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|} \leq \varepsilon \leq \frac{1}{2}.$$

which means we must have that $|g^{-1}(1)| \geq \frac{|f^{-1}(1)|}{2}$. Hence

$$\text{rel-dist}(g, f) = \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|g^{-1}(1)|} \leq \frac{|f^{-1}(1) \triangle g^{-1}(1)|}{2|f^{-1}(1)|} \leq 2\text{rel-dist}(f, g) \leq 2\varepsilon.$$

■

Lemma 10 (Approximate triangle inequality for relative distance) *Let* $f, g, h : \{0,1\}^n \to \{0,1\}$ *be such that* $\text{rel-dist}(f,g) \le \varepsilon$ *and* $\text{rel-dist}(g,h) \le \varepsilon'$. *Then* $\text{rel-dist}(f,h) \le \varepsilon + (1+\varepsilon)\varepsilon'$.

Proof Since $\text{rel-dist}(f,g) \le \varepsilon$ we have that

$$\frac{|f^{-1}(1) \triangle g^{-1}(1)|}{|f^{-1}(1)|} \le \varepsilon,$$

which means we must have $|g^{-1}(1)| \le (1+\varepsilon)|f^{-1}(1)|$. Hence

$$\begin{aligned}
\text{rel-dist}(f,h) &= \frac{|f^{-1}(1) \triangle h^{-1}(1)|}{|f^{-1}(1)|} \\
&= \frac{1}{|f^{-1}(1)|}\left|\{z \,:\, f(z) \neq h(z)\}\right| \\
&\le \frac{1}{|f^{-1}(1)|}\left|\{z \,:\, f(z) \neq g(z) \vee g(z) \neq h(z)\}\right| \\
&\le \frac{1}{|f^{-1}(1)|}\left|\{z \,:\, f(z) \neq g(z)\}\right| + \frac{1}{|f^{-1}(1)|}\left|\{z : g(z) \neq h(z)\}\right| \\
&\le \text{rel-dist}(f,g) + (1+\varepsilon)\frac{1}{|g^{-1}(1)|}\left|\{z : g(z) \neq h(z)\}\right| \\
&= \text{rel-dist}(f,g) + (1+\varepsilon)\text{rel-dist}(g,h) \\
&\le \varepsilon + (1+\varepsilon)\varepsilon'.
\end{aligned}$$

■

We will use the following one-sided algorithm for junta testing in the standard setting Blais (2009):

Theorem 11 *There exists a one-sided adaptive algorithm,* UNIFORMJUNTA$(f, k, \varepsilon, \delta)$, *for* ε*-testing* k*-juntas that makes* $O((k/\varepsilon + k\log k)\log(1/\delta))$ *queries. The algorithm always accepts* f *if it is a* k*-junta and rejects* f *with probability at least* $1-\delta$ *if it is* ε*-far from every* k*-junta with respect to the uniform distribution.*

A key subroutine our algorithm will use is binary search over "blocks" of variables to find new relevant blocks. In more detail, let $X_1, \ldots X_r$ be a partition of $[n]$ (we will frequently refer to a set X_i as a "block"). Fix some $I \subseteq [r]$, and let $X = \cup_{\ell \in I} X_\ell$. Given $u \in \{0,1\}^n$ and $w \in \{0,1\}^{\overline{X}}$ such that $f(u) \neq f(u_X \circ w)$, it is straightforward, using binary search, to identify an $X_\ell, \ell \notin I$, and a string $v \in \{0,1\}^n$ with $f(v) \neq f(w_{X_\ell} \circ v_{\overline{X_\ell}})$, using $O(\log r)$ queries to f. The (simple and standard) idea is as follows: Let $R = [r] \setminus I$ be indices of blocks outside X, and let $a = u, b = u_X \circ w$. We repeat the following until $|R| = 1$:

> Let R' be the first half of R, and $Y = \cup_{\ell \in R'} X_\ell$. Query $c = u_{[n]\setminus Y} \circ w_Y$ (this is u with blocks in S' set to w). If $f(a) \neq f(c)$, we set $R = R'$ and $b = c$. Else we set $R = R \setminus R'$ and $a = c$.

(See Subroutine 2 of Liu et al. (2019) for an detailed implementation of the above sketch.)

Appendix B. Soundness and completeness analysis of the junta subclass tester: Proof of Theorem 2 and Corollary 3

Throughout Appendix B , we assume that $0 < \varepsilon < 1/2$.

B.1. The analysis when f is a k-junta

Throughout this subsection we assume that $f : \{0,1\}^n \to \{0,1\}$ is a k-junta in $\mathcal{C}(k)$, and let J be its set of relevant variables with $|J| \le k$. The main result of this subsection is Theorem 20, which establishes that Algorithm 1 accepts in this case with probability at least $2/3$.

B.1.1. PHASES 1 AND 2

We start by noting that most likely, every block $\boldsymbol{X}_\ell$ contains at most one variable in J:

Lemma 12 *In Phase 1, with probability at least $1 - 1/20$, we have $|\boldsymbol{X}_i \cap J| \le 1$ for all $i \in [r]$.*

Proof Fix any $j_1, j_2 \in J$. The probability that they lie in the same block is $1/r$. By a union bound and the assumption that $|J| \le k$, the probability of $|\boldsymbol{X}_i \cap J| > 1$ for some i is at most

$$\binom{k}{2} \cdot \frac{1}{r} \le \frac{1}{20}.$$

■

From now on we fix the partition $X_1, \dots, X_r$ of $[n]$ and assume that $|X_i \cap J| \le 1$ for all $i \in [r]$. The main lemma for Phase 1 is as follows:

Lemma 13 *Assume that $f \in \mathcal{C}(k)$ and $|X_i \cap J| \le 1$ for all i. Then Algorithm 1 always reaches Phase 3 and when this happens, f^ℓ is a literal for every $\ell \in \boldsymbol{I}$. Moreover, with probability at least $1 - 1/20$, Algorithm 1 reaches Phase 3 with $\boldsymbol{I}$ and $\boldsymbol{X}$ satisfying the following condition:*

$$\Pr_{\substack{\boldsymbol{y} \sim f^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}}} \left[f(\boldsymbol{y}_{\boldsymbol{X}} \circ \boldsymbol{w}) = 0 \right] \le \kappa/4. \tag{5}$$

Proof For each ℓ added to $\boldsymbol{I}$, the algorithm has found a v^ℓ such that f^ℓ is not a constant function and thus, X_ℓ contains exactly one relevant variable in J and f^ℓ must be a literal of that variable.

Given that $|J| \le k$, $\boldsymbol{I}$ can never be larger than k and thus, Algorithm 1 always reaches Phase 2. Using Theorem 11 and the fact that $f^\ell(x) = \overline{f^\ell(\overline{x})}$ for any string x when f^ℓ is a literal, we have that Algorithm 1 always reaches Phase 3.

Now for Algorithm 1 to violate Equation (5) when reaching Phase 3, it must be the case that it has drawn $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$ for $T_2 = (5\log(20k+1))/\kappa$ many times, and $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) = 1$ every time. But given Equation (5), the probability that this happens is at most

$$\left(1 - \kappa/4\right)^{T_2} \le e^{-(5/4)\log(20k+1)} \le 1/(20k).$$

By a union bound (over the at most k different $\boldsymbol{X}$'s that are in play across all executions of line 4) the probability of Algorithm 1 reaching Phase 3 while violating Equation (5) is at most $1/20$. ■

In the rest of the proof, we fix I and X and assume the following:

Assumption 14 *Algorithm 1 reaches Phase 3 with I and X that satisfies the following conditions:*

1. *For each $\ell \in I$, we have $|X_\ell \cap J| = 1$, and denote the unique element in $X_\ell \cap J$ by $\tau(\ell)$;*
2. *Every f^ℓ, $\ell \in I$, is a literal that is either $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$.*

3. Equation (5) holds.

Let $h = |I|$ and $\ell_1 < \ldots < \ell_h$ be elements in I. Let $\sigma : [h] \to [n]$ be the injective map defined by $\sigma(i) = \tau(\ell_i)$, and let S be $\{\tau(\ell) : \ell \in I\}$ (or the image of σ) with $|S| = h$.

Let $\mathsf{G} : \{0,1\}^h \to \{0,1\}$ be the following h-variable function:

$$\mathsf{G}(z) := \arg\max_{b\in\{0,1\}} \left\{ \mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{\overline{S}}} \left[f(\sigma(z) \circ \boldsymbol{w}) = b \right] \right\}. \tag{6}$$

To see that $\mathsf{G} \in \textsc{Approx}(h,\kappa)$ whenever $\text{rel-dist}(f, \mathsf{G}_\sigma) \le \kappa$, recall that J is the set of relevant variables of f. So let $\{\alpha_1, \ldots, \alpha_{|J|-|I|}\}$ be the elements of $J \setminus S$. And denote $[n] \setminus J$ as $\{\beta_1, \ldots, \beta_{n-|J|}\}$. Consider the permutation $\pi : [n] \to [n]$ defined as follow:

$$\pi(j) = \begin{cases} \tau(\ell_j) \text{ if } 1 \le j \le |I| \\ \alpha_{j-|I|} \text{ if } |I|+1 \le j \le |J| \\ \beta_{j-|J|} \text{ if } |J|+1 \le j \le n \end{cases}$$

Since $\mathcal{C}(k)$ is closed under permutation $f_\pi \in \mathcal{C}(k)$, more specifically we have $f_\pi \in \mathcal{C}(k)^*$ (since f_π only depends on the first $|J|$ variables). So letting $\sigma' : [h] \to [n]$ be the injective map such that $\sigma'(i) = i$. We have $(\mathsf{G}_\sigma)_\pi = \mathsf{G}_{\sigma'}$, we have

$$\mathsf{G}_{\sigma'}(z) = \arg\max_{b\in\{0,1\}} \left\{ \mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{n-h}} \left[f_\pi(z_{[h]} \circ \boldsymbol{w}) = b \right] \right\}.$$

By Observation 1 we have $\text{rel-dist}(f_\pi, \mathsf{G}_{\sigma'}) = \text{rel-dist}(f, \mathsf{G}_\sigma) \le \kappa$. Finally since $\mathsf{G}(z) = \mathsf{G}_{\sigma'}(z \circ 0^{n-h})$, by the construction of $\textsc{Approx}(h,\kappa)$, we indeed have that $\mathsf{G} \in \textsc{Approx}(h,\kappa)$.

The lemma below shows that condition 3 in Assumption 14 (Equation (5)) implies $\text{rel-dist}(f, \mathsf{G}_\sigma) \le \kappa$.

Lemma 15 *If* $\text{rel-dist}(f, \mathsf{G}_\sigma) > \kappa$*, then we have*

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1) \\ \boldsymbol{w}\sim\{0,1\}^{\overline{X}}}} \left[f(\boldsymbol{u}_X \circ \boldsymbol{w}) = 0 \right] > \kappa/4.$$

We delay the proof to Appendix B.4. We can now replace condition 3 in Assumption 14 by

$3'$. The function G defined in Equation (6) satisfies $\text{rel-dist}(f, \mathsf{G}_\sigma) \le \kappa$ and is in $\textsc{Approx}(h,\kappa)$.

B.1.2. PHASES 3 AND 4

Assume that Algorithm 1 reaches Phase 3 with I and X satisfying Assumption 14 with condition $3'$. We first show that $\mathsf{G} \in \textsc{Approx}(h,\kappa)$ survives the loop on lines 14–20 with high probability:

Lemma 16 *Under Assumption 14, with probability at least* $1 - 1/20$*,* G *remains in* $\boldsymbol{A}$ *when Algorithm 1 reaches line 21.*

Proof By Lemma 7, the $\boldsymbol{v}$ on line 19 is always $\sigma^{-1}(\boldsymbol{u})$ with $\boldsymbol{u} \sim f^{-1}(1)$ and thus $\mathsf{G}(\boldsymbol{v}) = \mathsf{G}_\sigma(\boldsymbol{u})$. Given that $\text{rel-dist}(f, \mathsf{G}_\sigma) \le \kappa$ by Assumption 14, we have that each iteration of the loop removes G from $\boldsymbol{A}$ with probability at most κ. Hence, by a union bound over all $T_1 = 1/(20\kappa)$ loops, the probability that G is removed is at most $T_1\kappa = 1/20$. ■

Lemma 17 *Under Assumption 14, with probability at least* $1 - 1/20$*, on line 21 every* $g \in \boldsymbol{A}$ *has*

$$\Pr_{\boldsymbol{u} \sim f^{-1}(1)} \left[g_\sigma(\boldsymbol{u}) = 0 \right] \leq \varepsilon/500.$$

Proof By Lemma 7, the $\boldsymbol{v}$ on line 19 is always $\sigma^{-1}(\boldsymbol{u})$ with $\boldsymbol{u} \sim f^{-1}(1)$ and thus $g(\boldsymbol{v}) = g_\sigma(\boldsymbol{u})$. Let g be any function in $\textsc{Approx}(h, \kappa)$ with

$$\Pr_{\boldsymbol{u} \sim f^{-1}(1)} \left[g_\sigma(\boldsymbol{u}) = 0 \right] \geq \varepsilon/500.$$

Each iteration of the loop removes g with probability at least $\varepsilon/500$ and thus, g survives with probability at most $(1 - \varepsilon/500)^{T_1}$. Given the choice of

$$T_1 = O\left(\frac{\log |\mathcal{C}(k)^*|}{\varepsilon} \right),$$

by making the hidden constant large enough, g survives with probability at most $1/(20|\mathcal{C}(k)^*|)$. So the lemma follows from Fact 5 and a union bound over all functions g. ■

Hence we get the following corollary about the function $\mathfrak{g}$ passed down to Phase 4:

Corollary 18 *Under Assumption 14, with probability at least* $1 - 1/10$*, Algorithm 1 reaches Phase 4 with* $\mathfrak{g} \in \textsc{Approx}(h, \kappa)$ *that satisfies*

$$\Pr_{\substack{\boldsymbol{z} \sim \mathfrak{g}^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{\overline{S}}}} \left[f(\sigma(\boldsymbol{z}) \circ \boldsymbol{w}) = 0 \right] \leq \varepsilon/400. \tag{7}$$

Proof Assume that the conclusions of both Lemma 16 and Lemma 17 hold. So $\mathfrak{g}$ is picked and satisfies

$$|\mathfrak{g}_\sigma^{-1}(1)| \leq |\mathsf{G}_\sigma^{-1}(1)| \quad \text{and} \quad \frac{|f^{-1}(1) \setminus \mathfrak{g}_\sigma^{-1}(1)|}{|f^{-1}(1)|} \leq \varepsilon/500. \tag{8}$$

Let $N = |f^{-1}(1)|$. Since $\text{rel-dist}(f, \mathsf{G}_\sigma) \leq \kappa$, we have that

$$|\mathfrak{g}_\sigma^{-1}(1)| \leq |\mathsf{G}_\sigma^{-1}(1)| \leq (1 + \kappa)N.$$

On the other hand, from the second part of Equation (8) we have that

$$|\mathfrak{g}_\sigma^{-1}(1)| \geq |f^{-1}(1) \cap \mathfrak{g}_\sigma^{-1}(1)| \geq (1 - \varepsilon/500)N.$$

As a result, the probability on the left-hand side of Equation (7) is

$$\frac{|\mathfrak{g}_\sigma^{-1}(1) \setminus f^{-1}(1)|}{|\mathfrak{g}_\sigma^{-1}(1)|} \leq \frac{(1 + \kappa)N - (1 - \varepsilon/500)N}{(1 - \varepsilon/500)N} \leq \varepsilon/400,$$

where we used the choice that κ can be made sufficiently small compared to ε by making the hidden constant in T_1 large enough. ■

Finally we show that Phase 4 accepts with high probability:

Lemma 19 *Under Assumption 14 and assuming that Algorithm 1 reaches Phase 4 with a function* $\mathfrak{g}$ *that satisfies Equation (7), then Algorithm 1 rejects in Phase 4 with probability at most* $1/20$*.*

Proof By Lemma 8 and the observation made in the overview, $\boldsymbol{u}$ on line 24 has the same distribution as $\sigma(\boldsymbol{z}) \circ \boldsymbol{w}$ where $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$. As a result, Algorithm 1 rejects with probability at most $1/20$ by a union bound over the $20/\varepsilon$ many loops. ■

B.1.3. PUTTING THE PIECES TOGETHER

We summarize the above step-by-step analysis with the following theorem:

Theorem 20 *If $f \in \mathcal{C}(k)$, then Algorithm 1 accepts f with probability at least $2/3$.*

Proof By Lemma 12 and Lemma 13, with probability at least $1 - 1/10$ Algorithm 1 reaches Phase 3 with I, X satisfying Assumption 14. When this happens, by Corollary 18 and Lemma 19, Algorithm 1 passes through Phases 3 and 4 and accepts with probability at least $1-3/20$. Thus, the probability that the algorithm rejects is at most $1/10 + 3/20 < 1/3$. ■

B.2. The analysis when f is far from $\mathcal{C}(k)$.

Throughout this subsection we assume that f is ε-far from every function in $\mathcal{C}(k)$ in relative distance, and we show in Theorem 25 that Algorithm 1 rejects f with probability at least 2/3.

B.2.1. PHASES 1 AND 2

Fix any partition $X_1, \ldots, X_\ell$ of $[n]$. Clearly $|\boldsymbol{I}| \leq k$ when Algorithm 1 reaches Phase 2; otherwise f would have been rejected in Phase 1.

The lemma below shows that it is unlikely for Algorithm 1 to reach Phase 3 if any f^ℓ, $\ell \in \boldsymbol{I}$, is far from every literal under the uniform distribution:

Lemma 21 *The probability of Algorithm 1 reaching Phase 3 with $\boldsymbol{I}$ such that f^ℓ for some $\ell \in \boldsymbol{I}$ is $(1/30)$-far from every literal under the uniform distribution is at most $1/15$.*

Proof If f^ℓ is $1/30$-far from every literal with respect to the uniform distribution then it is either

1. $1/30$-far from every 1-Junta,
2. or $1/30$ far from every literal and $1/30$ close to a 0-Junta (i.e. the constant-0 or constant-1 function).

In case 1, by Theorem 11, with probability at least $14/15$, UNIFORMJUNTA$(f^\ell, 1, 1/30, 1/15)$ rejects, in which case Algorithm 1 rejects. In case 2, f^ℓ is $(1/30)$-close to a constant function; say w.l.o.g. that it is close to the constant-0 function. The probability we fail to reject on line 11 is :

$$\mathop{\mathbf{Pr}}_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\boldsymbol{b}) \neq f^\ell(\overline{\boldsymbol{b}})\right] \leq \mathop{\mathbf{Pr}}_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\overline{\boldsymbol{b}}) = 1\right] + \mathop{\mathbf{Pr}}_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\boldsymbol{b}) = 1\right] \leq 1/15,$$

where the last inequality follows from the fact f^ℓ is $(1/30)$-close to the constant-0 function under the uniform distribution and $\boldsymbol{b}$, $\overline{\boldsymbol{b}}$ are both sampled uniformly at random. ■

Fix I and X with $|I| = h \leq k$, and let $\ell_1 < \cdots < \ell_h$ be the elements of I. We make the following assumption in the rest of this subsection:

Assumption 22 *Algorithm 1 reaches Phase 3 with I and X satisfying the following condition:*

For every $\ell \in I$, f^ℓ is $(1/30)$-close to a literal $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ under the uniform distribution.

Similar to the previous subsection, we let $\sigma : [h] \to [n]$ be the injective map with $\sigma(i) = \tau(\ell_i)$ for each $i \in [h]$, and let $S = \{\tau(\ell) : \ell \in I\}$ with $|S| = h$.

B.2.2. PHASES 3 AND 4

Under Assumption 22, we show in the next lemma that most likely, every function $g \in \text{APPROX}(h, \kappa)$ that survives in $\boldsymbol{A}$ up to line 21 satisfies that $g(\sigma^{-1}(\boldsymbol{u})) = 0$ with low probability when $\boldsymbol{u} \sim f^{-1}(1)$:

Lemma 23 *Under Assumption 22, with probability at least $1 - 1/20$, when Algorithm 1 reaches line 21 every $g \in \boldsymbol{A}$ satisfies*

$$\mathop{\mathbf{Pr}}_{\boldsymbol{u} \sim f^{-1}(1)} \left[g_\sigma(\boldsymbol{u}) = 0 \right] \leq \varepsilon/500.$$

Proof The proof is very similar to that of Lemma 17. The only difference is that it is not always the case that $\boldsymbol{v} = \sigma^{-1}(\boldsymbol{u})$; by a union bound over the at most k calls to FINDVARVALUE (as well as the choice of $\delta = 1/(20k)$), we have $\boldsymbol{v} = \sigma^{-1}(\boldsymbol{u})$ with probability at least $19/20$. Therefore, any g that violates the inequaility above would be removed with probability at least $(19/20)\varepsilon/500$ in each loop. The rest of the proof is the same as that of Lemma 17. ■

Assuming the conclusion of Lemma 23 holds, we have that either Algorithm 1 rejects because $\boldsymbol{A}$ is empty at the end of Phase 3 (in which case we are done), or the function $\mathfrak{g} \in \text{APPROX}(h, \kappa)$ passed down to Phase 4 satisfies

$$\mathop{\mathbf{Pr}}_{\boldsymbol{u} \sim f^{-1}(1)} \left[\mathfrak{g}_\sigma(\boldsymbol{u}) = 0 \right] \leq \varepsilon/500.$$

Letting $N = |f^{-1}(1)|$, we have that

$$|f^{-1}(1) \setminus \mathfrak{g}_\sigma^{-1}(1)| \leq \varepsilon N/500.$$

On the other hand, given that $\mathfrak{g} \in \text{APPROX}(h, \kappa)$, by our assumption in this subsection that $\text{rel-dist}(f, \mathcal{C}(k) > \varepsilon$ and Lemma 6 (and the fact that κ is sufficiently smaller than ε), we have $\text{rel-dist}(f, \mathfrak{g}_\sigma) \geq \varepsilon/2$, and we hope to show that

$$\frac{|\mathfrak{g}_\sigma^{-1}(1) \setminus f^{-1}(1)|}{|\mathfrak{g}_\sigma^{-1}(1)|} \geq \varepsilon/7. \tag{9}$$

To this end, consider two cases. (1) If $|\mathfrak{g}_\sigma^{-1}(1)| \geq 2 \cdot |f^{-1}(1)|$, then the ratio above is at least

$$\frac{|\mathfrak{g}_\sigma^{-1}(1)| - |f^{-1}(1)|}{|\mathfrak{g}_\sigma^{-1}(1)|} \geq \frac{|\mathfrak{g}_\sigma^{-1}(1)|/2}{|\mathfrak{g}_\sigma^{-1}(1)|} \geq 1/2 > \varepsilon/7.$$

(2) If $|\mathfrak{g}_\sigma^{-1}(1)| < 2 \cdot |f^{-1}(1)|$, then the ratio is at least

$$\frac{|\mathfrak{g}_\sigma^{-1}(1) \triangle f^{-1}(1)| - |f^{-1}(1) \setminus \mathfrak{g}_\sigma^{-1}(1)|}{2N} \geq \frac{\varepsilon N/2 - \varepsilon N/500}{2N} \geq \varepsilon/7.$$

Finally, we show that Algorithm 1 rejects in Phase 4 with high probability:

Lemma 24 *Under Assumption 22, assume that Algorithm 1 reaches Phase 4 with $\mathfrak{g}$ satisfying Equation (9). Then Phase 4 rejects with probability at least 0.8.*

Proof Recall the observation in the overview that $\boldsymbol{y}_{\sigma,\boldsymbol{z}}$ has the same distribution as $\sigma(\boldsymbol{z}) \circ \boldsymbol{w}$ with $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$. Hence the probability of $f(\boldsymbol{y}_{\sigma,\boldsymbol{z}}) = 0$ for at least one loop, by Equation (9), is at least

$$1 - (1 - \varepsilon/7)^{20/\varepsilon} \geq 1 - e^{-20/7}.$$

For that loop, by Lemma 8, we have $\boldsymbol{u} = \boldsymbol{y}_{\sigma,\boldsymbol{z}}$ with probability at least $19/20$, in which case $f(\boldsymbol{u}) = 0$ and Algorithm 1 rejects. Overall Algorithm 1 rejects with probability at least

$$(19/20)(1 - e^{-20/7}) \geq 0.8.$$

■

B.2.3. PUTTING THE PIECES TOGETHER

Theorem 25 *If $f : \{0,1\}^n \to \{0,1\}$ is ε-far from every function in $\mathcal{C}(k)$ in relative distance, then Algorithm 1 rejects with probability at least $2/3$.*

Proof By a union bound over the bad events in Lemma 21, Lemma 23 and Lemma 24, we have that Algorithm 1 accepts with probability at most $1/15 + 1/20 + 0.2 < 1/3$. ■

B.3. Query complexity of Algorithm 1

Finally, we show that the query complexity of Algorithm 1 (the number of random samples drawn from $f^{-1}(1)$ and membership queries on f) is at most $\tilde{O}\big((k/\varepsilon)\log|\mathcal{C}(k)^*|\big)$.

Recall the choice of T_2 in Algorithm 1. In Phase 1, we repeat lines 3–7 until either $|\boldsymbol{I}| > k$ or $t = T_2$. In one iteration of the loop, if the "if" condition on line 4 is true, then lines 5–7 are executed which makes $O(\log r) = O(\log k)$ queries, and $|\boldsymbol{I}|$ increases by one and t is set to 0. And if the "if" condition on line 4 is false, then line 4 only makes two queries and $|\boldsymbol{I}|$ does not increase, but t increases by one. Therefore, the query complexity of this phase is

$$O\big(k(T_2 + \log k)\big) = \tilde{O}\big((k/\varepsilon)\log|\mathcal{C}(k)^*|\big).$$

For Phase 2, given that $|\boldsymbol{I}| \le k$, the query complexity is $O(k)$ by Theorem 11.

In Phase 3, given that the number of queries made by each call to FINDVARVALUE is $O(\log k)$ and $h \le k$, the query complexity is

$$O(kT_1 \log k) = \tilde{O}\big((k/\varepsilon)\log|\mathcal{C}(k)^*|\big).$$

Given that MAPBACK makes $O(k \log k)$ queries, the query complexity of Phase 4 is $O((k/\varepsilon)\log k)$. Putting things together, the query complexity of Algorithm 1 is $\tilde{O}\big((k/\varepsilon)\log|\mathcal{C}(k)^*|\big)$ as claimed.

B.4. Proof of Lemma 15

Proof Recall that in the context of Lemma 15, f is a J-junta, $S = X \cap J$, and each block of X contains exactly one relevant variable. We consider the following equivalent way of drawing $\boldsymbol{u}_X \circ \boldsymbol{w}$ with $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{X}}$: draw $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{S}}$ and return $\boldsymbol{u}_S \circ \boldsymbol{w}$. The two distributions are the same because every variable in $X \setminus S$ is irrelevant. As a result,

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{X}}}}\big[f(\boldsymbol{u}_X \circ \boldsymbol{w}) = 0\big] = \mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\big[f(\boldsymbol{u}_S \circ \boldsymbol{w}) = 0\big]. \tag{10}$$

Let $N = |f^{-1}(1)|$, $F = f^{-1}(1)$ and $G = \mathsf{G}_\sigma^{-1}(1)$. Given that $|F \triangle G| \ge \kappa N$, we consider the two cases of $|F \setminus G| > \kappa N/2$ and $|G \setminus F| > \kappa N/2$.

For the case when $|F \setminus G| > \kappa N/2$, we have

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\big[f(\boldsymbol{u}_S \circ \boldsymbol{w}) = 0\big] \ge \mathop{\mathbf{Pr}}_{\boldsymbol{u}\sim f^{-1}(1)}\big[\boldsymbol{u} \in F \setminus G\big] \cdot \mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\big[f(\boldsymbol{u}_S \circ \boldsymbol{w}) = 0 \,|\, \boldsymbol{u} \in F \setminus G\big].$$

The first probability on the RHS is at least $\kappa/2$; the second probability on the RHS is at least $1/2$ given the definition of G from f and the condition that $\boldsymbol{u} \in F \setminus G$ and thus, $\mathsf{G}_\sigma(\boldsymbol{u}) = 0$.

For the second case when $|G \setminus F| \ge \kappa N/2$, consider any fixed $a \in G \setminus F$. Since $a \in G$, we have

$$\mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{\overline{S}}}\big[f(a_S \circ \boldsymbol{w}) = 1\big] \ge 1/2$$

and thus there must exist $2^{|\overline{S}|}/2$ many $y \in F$ with $y_S = a_S$. So

$$\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\left[\boldsymbol{u}_S \circ \boldsymbol{w} = a\right] \geq \Pr_{\boldsymbol{u}\sim f^{-1}(1)}\left[\boldsymbol{u}_S = a_S\right] \cdot \Pr_{\boldsymbol{w}\sim\{0,1\}^{\overline{S}}}\left[\boldsymbol{w} = a_{\overline{S}}\right] \geq \frac{2^{|\overline{S}|}}{2|A|} \times 2^{-|\overline{S}|} \geq \frac{1}{2|A|}.$$

As a result, we have

$$\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\left[f(\boldsymbol{u}_S \circ \boldsymbol{w}) = 0\right] \geq \sum_{a\in G\setminus F}\left(\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{S}}}}\left[\boldsymbol{y}_S \circ \boldsymbol{w} = a\right]\right) \geq \frac{\kappa|A|}{2} \cdot \frac{1}{2|A|} = \frac{\kappa}{4}.$$

This finishes the proof of the lemma. ■

B.5. Applications of Theorem 2: Proof of Corollary 3

For the class $\mathcal{C}(k)$ of size-k decision trees, we observe that every size-k decision tree is a k-junta and we use the fact that $|\mathcal{C}(k)^*|$, the number of size-k decision trees over $x_1, \ldots, x_k$, is at most $k^{O(k)}$ (see Diakonikolas et al. (2007) for the simple proof). The claimed testing result follows immediately by applying Theorem 2. A similar argument is used for the class of size-k branching programs, using the fact that there are at most $k^{O(k)}$ size-k branching programs, and likewise for the class of size-k Boolean formulas; again see Diakonikolas et al. (2007) for the simple counting arguments giving these upper bounds. (Here we are using the standard definition that the size of a Boolean formula is the number of leaves.)

Finally, we note that Theorem 2 can be applied to obtain efficient relative-error testing algorithms for a number of other subclasses of juntas that were studied in the uniform-distribution setting; as detailed in Bshouty (2020), these include s-term monotone r-DNF, s-term unate r-DNF, length-k-decision list, parities of length at most k, conjunctions of length at most k, s-sparse polynomials over $\mathbb{F}_2$ of degree d, and functions with Fourier degree at most d. We leave the straightforward application of Theorem 2 to these classes as an exercise for the interested reader.

Appendix C. Testing Juntas with Relative Error: Proof of Theorem 1

In this section we present a two-sided $\tilde{O}(k/\varepsilon)$-tester for k-juntas under relative distance. We assume that $0 < \varepsilon < 1/2$.

Before presenting the tester, we prove a lemma (Lemma 27) essential for both the design and analysis of our tester. We start with a relative-distance analogue of Lemma 3.2 of Liu et al. (2019); in words, it says that if $\text{rel-dist}(f, g)$ is large for every k-junta g, then for any fixed set J of at most k variables, rerandomizing all variables outside of J in a random satisfying assignment of f is fairly likely to make it into an unsatisfying assignment of f.

Lemma 26 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function such that $\text{rel-dist}(f, g) \geq \varepsilon$ for every k-junta g. Then for any $J \subseteq [n]$ with $|J| \leq k$, we have*

$$\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[f(\boldsymbol{u}_J \circ \boldsymbol{w}) = 0\right] = \Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[f(\boldsymbol{u}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{w})\right] \geq \varepsilon/4.$$

Proof Consider the function $h : \{0,1\}^n \to \{0,1\}$ defined as follows:

$$h(x) = \arg\max_{b\in\{0,1\}}\left\{\Pr_{\boldsymbol{w}\sim\{0,1\}^{\overline{J}}}\left[f(x_J \circ \boldsymbol{w})) = b\right]\right\}$$

where ties are broken arbitrarily. Since h is a J-junta, letting $N := |f^{-1}(1)|$, we have

$$\left|f^{-1}(1) \triangle h^{-1}(1)\right| \geq \varepsilon N.$$

Let Z_1 be the set of $z \in \{0,1\}^n$ with $f(z) = 1$ and $h(z) = 0$, and let Z_2 be the set of $z \in \{0,1\}^n$ with $f(z) = 0$ and $h(z) = 1$. Then either Z_1 or Z_2 is of size at least $\varepsilon N/2$.

In the first case ($|Z_1| \geq \varepsilon N/2$), we have

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[f(\boldsymbol{u}_J \circ \boldsymbol{w}) = 0\right] \geq \frac{1}{N}\sum_{z\in Z_1}\mathop{\mathbf{Pr}}_{\boldsymbol{w}\sim\{0,1\}^{\overline{J}}}\left[f(z_J \circ \boldsymbol{w}) = 0\right] \geq \frac{1}{N}\cdot|Z_1|\cdot\frac{1}{2} \geq \varepsilon/4,$$

where the second inequality used $h(z) = 0$ and the definition of h.

In the second case ($|Z_2| \geq \varepsilon N/2$), given the definition of Z_2 we have

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[f(\boldsymbol{u}_J \circ \boldsymbol{w}) = 0\right] \geq \sum_{z\in Z_2}\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[\boldsymbol{u}_J \circ \boldsymbol{w} = z\right]. \tag{11}$$

Fixing any point $z \in Z_2$, we have

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{J}}}}\left[\boldsymbol{u}_J \circ \boldsymbol{w} = z\right] \geq \frac{2^{|\overline{J}|-1}}{N}\cdot\frac{1}{2^{|\overline{J}|}} \geq \frac{1}{2N}, \tag{12}$$

where $2^{|\overline{J}|-1}$ lowerbounds the number of $x \in f^{-1}(1)$ with $x_J = z_J$ given the definition of h. The second case then follows by combining $|Z_2| \geq \varepsilon N/2$, Equation (11), and Equation (12). ■

We are now ready to prove Lemma 27.

Lemma 27 *Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function that is ε-far from every k-junta in relative distance. Let $X \subseteq [n]$ be such that*

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{X}}}}\left[f(\boldsymbol{u}) \neq f(\boldsymbol{u}_X \circ \boldsymbol{w})\right] \leq \varepsilon/20$$

and let J be a subset of at most k variables from X. Then we have

$$\mathop{\mathbf{Pr}}_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{X}},\\ \boldsymbol{y}\sim\{0,1\}^{X\setminus J}}}\left[f(\boldsymbol{u}_X \circ \boldsymbol{w}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})\right] \geq \varepsilon/5.$$

Proof Let $\boldsymbol{u} \sim f^{-1}(1)$, $\boldsymbol{w} \sim \{0,1\}^{\overline{X}}$ and $\boldsymbol{y} \sim \{0,1\}^{X\setminus J}$ as above. Then the following inequality completes the proof (where the last line uses the assumption of the Lemma, and Lemma 26.

$$\begin{aligned}
\mathbf{Pr}\left[f(\boldsymbol{u}_X \circ \boldsymbol{w}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})\right] &\geq \mathbf{Pr}\left[f(\boldsymbol{u}) = f(\boldsymbol{u}_X \circ \boldsymbol{w}) \wedge f(\boldsymbol{u}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})\right] \\
&\geq \mathbf{Pr}\left[f(\boldsymbol{u}) = f(\boldsymbol{u}_X \circ \boldsymbol{w})\right] + \mathbf{Pr}\left[f(\boldsymbol{u}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})\right] - 1 \\
&\geq 1 - \varepsilon/20 + \varepsilon/4 - 1 \\
&= \varepsilon/5,
\end{aligned}$$

■

C.1. The tester and an overview

The junta testing algorithm, called JUNTATESTER, is presented in Algorithm 4. We bound its sample and query complexity in Appendix C.2. We show in Appendix C.3 that Algorithm 4 accepts with probability at least $2/3$ when f is a k-junta, and show in Appendix C.4 that it rejects with probability at least $2/3$ when f is ε-far from every k-junta in relative distance.

Before the formal proof, in this subsection we give an overview of the algorithm and its analysis. Since we are concerned with the entire class of k-juntas rather than a subclass, there is no need to do the kind of "implicit learning" that was used in Algorithm 1; this lets us achieve a much better query complexity than would follow from a naive application of Algorithm 1. Looking ahead, compared to Algorithm 1, the savings comes from the fact that no trimming is needed in Algorithm 4. Instead, at a high level the oranization of our Algorithm 4 is set up so as to follow the distribution-free junta testing algorithm and proof of correctness from Bshouty (2019) quite closely, but there are some differences which we describe at the end of this subsection.

Let $f : \{0,1\}^n \to \{0,1\}$ be the input function that is being tested. In Phase A, Algorithm 4 starts by drawing a partition of variables $[n]$ into $r = O(k^2)$ many blocks $\boldsymbol{X}_1, \dots, \boldsymbol{X}_r$ uniformly at random. In Phase B, Algorithm 4 tries to find as many *relevant* blocks $\boldsymbol{X}_\ell$ as possible under a query complexity budget. Here intuitively we say $\boldsymbol{X}_\ell$ is a relevant block of f if we have found a string $v^\ell \in \{0,1\}^n$ such that $f^\ell : \{0,1\}^{\boldsymbol{X}_\ell} \to \{0,1\}$, defined as

$$f^\ell(x) = f\left(x \circ v^\ell_{\overline{\boldsymbol{X}_\ell}}\right),$$

is not a constant function. Let $\{\boldsymbol{X}_\ell\}_{\ell \in I}$ be the set of relevant blocks found so far (with $I = \emptyset$ at the beginning), and let $\boldsymbol{X}$ be their union. Phase B draws a uniform satisfying assignment $\boldsymbol{u} \sim f^{-1}(1)$ and a random string $\boldsymbol{w}$ over $\{0,1\}^{\overline{\boldsymbol{X}}}$; if $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) = 0$, then a binary search over blocks can be run on $\boldsymbol{u}$ and $\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}$, using $O(\log r) = O(\log k)$ many queries, to find a new relevant block $\boldsymbol{X}^\ell$ together with an accompanying v^ℓ. If Phase B found too many relevant blocks (i.e., $|I| > k$) then Algorithm 4 rejects; on the other hand, if it fails to make progress by finding a new relevant block after $T = O(\log(k/\varepsilon))$ rounds, Algorithm 4 moves on to Phase C. The latter helps ensure that the number of queries used in Phase B stays within the budget of $\tilde{O}(k/\varepsilon)$.

In Phase C Algorithm 4 checks whether every function f^ℓ, $\ell \in I$, is close to a literal under the uniform distribution (i.e., $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ for some variable $\tau(\ell) \in \boldsymbol{X}_\ell$), and rejects if any of them is not.

When Algorithm 4 reaches Phase D, one may assume that the relevant blocks found satisfy the following two conditions:

1. First, we have
$$\Pr_{\substack{\boldsymbol{u} \sim f^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}}} \left[f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) \neq f(\boldsymbol{u})\right] \leq \varepsilon/20; \tag{13}$$
since otherwise it is unlikely for Phase B to fail to find a new relevant block after T repetitions before moving to Phase C.

2. Second, every f^ℓ is close to a literal $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$ for some unique $\tau(\ell) \in \boldsymbol{X}_\ell$; since otherwise Phase C rejects with high probability.

Let J be the set of indices $\{\tau(\ell)\}_{\ell \in I}$, with $|J| \leq k$. Then by Lemma 27, we have

$$\Pr_{\substack{\boldsymbol{u} \sim f^{-1}(1) \\ \boldsymbol{w} \sim \{0,1\}^{\overline{X}}, \\ \boldsymbol{y} \sim \{0,1\}^{X \setminus J}}} \left[f(\boldsymbol{u}_X \circ \boldsymbol{w}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})\right] \geq \varepsilon/5 \tag{14}$$

when f is ε-far from every k-junta in relative distance. While $\boldsymbol{u}$ and $\boldsymbol{w}$ above are easy to sample, Phase D uses a delicate subroutine (lines 16–25) to sample $\boldsymbol{y}$. The challenge here is that Algorithm 4 does not know J (even though it is well defined given that every f^ℓ is close to a literal) and it would require too many queries to explicitly identify any $\tau(\ell)$ given that each block is likely to contain $\Theta(n/k^2)$ many variables (recall that we wants to avoid $O(\log n)$-type query complexities). We explain how the subroutine works in Appendix C.4; Algorithm 4 rejects in Phase D if

$$f(\boldsymbol{u}_X \circ \boldsymbol{w}) \neq f(\boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w}) \tag{15}$$

and it accepts if this does not happen after $M = O(1/\varepsilon)$ rounds.

For the case when f is a k-junta, as will become clear in the analysis in Appendix C.3, Algorithm 4 always accepts if the partition is such that every block $\boldsymbol{X}_\ell$ contains at most one relevant variable of f. Given that we draw $r = O(k^2)$ blocks, this happens with high probability by a standard birthday paradox analysis. For the case when every k-junta is ε-far from f in relative distance, on the other hand, it follows from Equation (14) (which follows from Lemma 27 and the analysis of Phase B and C) that Algorithm 4 rejects with high probability.

Finally, we briefly discuss how our algorithm and (mostly) its analysis differ from Bshouty (2019). Each of our algorithm's phases is quite similar to the corresponding phase of the Bshouty (2019) algorithm. However, in Phase B, Bshouty (2019) picks a restriction f' of f in which every variable outside the "relevant blocks" is set to 0. Since Bshouty (2019) works in the distribution free setting it's easy to check whether f and f' are close, by simply sampling $\boldsymbol{u} \sim \mathcal{D}$ and checking if $f(\boldsymbol{u}) = f'(\boldsymbol{u})$. Finally, in Phase D Bshouty (2019) checks if f' is close to a specific k-junta under the distribution $\mathcal{D}$. In the no case of the distribution-free setting, since f is far from every k-junta and f' is close to f, f' must fail this final check with high probability.

In contrast, in our relative-error setting it is not easy to check if f is relative-error close to a restriction like f'. In the relative-error setting we can only get samples from $f^{-1}(1)$, and it is not clear how to use such samples to detect that f is far from f'. This is because it could be the case that $\text{rel-dist}(f, f')$ is large, but for $\boldsymbol{u} \sim f^{-1}(1)$ we always have $f'(\boldsymbol{u}) = 1$.

To deal with this issue, we do not use a restriction of f, but rather we set variables in the "irrelevant blocks" uniformly at random in both Phase B and Phase D. We do not get that f is close to some function f' by the end of Phase B; instead, the correctness of our algorithm crucially relies on Lemma 27, which does not have an analogue in the analysis of Bshouty (2019).

C.2. Sample and query complexity of Algorithm 4

We first note that Phase A does not make any samples or queries. In Phase B, we repeat lines 4–8 until either $|I| > k$ or $t = T$. In one iteration of the loop, if the "if" condition on line 5 is true, then lines 6–7 are executed which makes $O(\log r) = O(\log k)$ queries, and $|I|$ increases by one and t is set to 0. And if the "if" condition on line 5 is false, then line 5 only makes two queries and $|I|$ does not increase, but t increases by one. Therefore, the query complexity of this phase is $O(k(T + \log k)) = O(k \log(k/\varepsilon))$ given that $T = O(\log(k/\varepsilon))$. Moreover, the number of samples from $f^{-1}(1)$ that are drawn across Phase B is at most $O(kT) = O(k \log(k/\varepsilon))$, since line 4 is executed at most T times between every two consecutive increments of $|I|$, and $|I|$ never grows larger than k. In Phase C, lines 11–12 are executed at most k times, and by Theorem 11, each call to UNIFORMJUNTA makes $O(1)$ queries (and no samples), so the query complexity of this phase is $O(k)$. In the final phase, lines 15–27 are repeated $M = O(1/\varepsilon)$ times, and each time the query complexity is $O(kh) = O(k \log(k/\varepsilon))$, so the query complexity of this phase is $O((k/\varepsilon) \log(k/\varepsilon))$ (and this phase makes M samples, corresponding to the M executions of line 26). Therefore, the total number of queries and samples made by Algorithm 4 is $O((k/\varepsilon) \log(k/\varepsilon))$.

Algorithm 4: The relative error junta testing algorithm, JUNTATESTER(f, k, ε)

Phase A: Randomly partition $[n]$ into blocks

Parameters used in the algorithm: $r = 2k^2$, $T = O(\log(k/\varepsilon))$, $M = O(1/\varepsilon)$, and $h = O(\log(k/\varepsilon))$;
Choose uniformly at random an r-way partition $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r$ of $[n]$;

Phase B: Find relevant blocks

Set $\boldsymbol{X} = \emptyset$, $I = \emptyset$ and $t = 0$;
while $t \neq T$ **do**
  Draw uniform random $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$, and set $t \leftarrow t + 1$;
  if $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) \neq f(\boldsymbol{u})$ **then**
    Binary Search over the elements of $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_r$ that are subsets of $\overline{\boldsymbol{X}}$ to find a new relevant set $\boldsymbol{X}_\ell$ and a string $v^\ell \in \{0,1\}^n$ such that

$$f(v^\ell) \neq f\Big(\boldsymbol{w}_{\boldsymbol{X}_\ell} \circ v^\ell_{\overline{\boldsymbol{X}_\ell}}\Big).$$

    Denote by $f^\ell : \{0,1\}^{\boldsymbol{X}_\ell} \to \{0,1\}$ the following function:

$$f^\ell(x) := f\Big(x \circ v^\ell_{\overline{\boldsymbol{X}_\ell}}\Big).$$

    Set $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup \boldsymbol{X}_\ell$, $I \leftarrow I \cup \{\ell\}$ and $t \leftarrow 0$. Reject if $|I| > k$;
  end
end

Phase C: Test if each f^ℓ, $\ell \in I$, is close to a literal

foreach $\ell \in I$ **do**
  Run UNIFORMJUNTA$(f^\ell, 1, 1/30, 1/15)$; Reject if it rejects;
  Draw $\boldsymbol{b} \sim \{0,1\}^{\boldsymbol{X}_\ell}$; Reject and halt if $f^\ell(\boldsymbol{b}) = f^\ell(\overline{\boldsymbol{b}})$;
end

Phase D: The final test

repeat M **times**
  Draw $\boldsymbol{s} \sim \{0,1\}^{\boldsymbol{X}}$ and set $\boldsymbol{z}$ to be the all-zero string in $\{0,1\}^{\boldsymbol{X}}$;
  foreach $\ell \in I$ **do**
    Set $\boldsymbol{Y}_{\ell,\zeta} = \{j \in \boldsymbol{X}_\ell \mid \boldsymbol{s}_j = \zeta\}$ and $\boldsymbol{G}_{\ell,\zeta} = 0$ for both $\zeta \in \{0,1\}$;
    repeat h **times**
      Draw $\boldsymbol{b}^\zeta \sim \{0,1\}^{\boldsymbol{Y}_{\ell,\zeta}}$ for both $\zeta \in \{0,1\}$;
      If $f^\ell(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq f^\ell(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1)$ then $\boldsymbol{G}_{\ell,0} \leftarrow \boldsymbol{G}_{\ell,0} + 1$;
      If $f^\ell(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq f^\ell(\boldsymbol{b}^0 \circ \overline{\boldsymbol{b}^1})$ then $\boldsymbol{G}_{\ell,1} \leftarrow \boldsymbol{G}_{\ell,1} + 1$;
    end
    If $\{\boldsymbol{G}_{\ell,0}, \boldsymbol{G}_{\ell,1}\} \neq \{0, h\}$ then reject;
    If $\boldsymbol{G}_{\ell,0} = h$ then $\boldsymbol{z}_{\boldsymbol{X}_\ell} \leftarrow \boldsymbol{s}_{\boldsymbol{X}_\ell}$ else $\boldsymbol{z}_{\boldsymbol{X}_\ell} \leftarrow \overline{\boldsymbol{s}_{\boldsymbol{X}_\ell}}$;
  end
  Draw $\boldsymbol{u} \sim f^{-1}(1)$, $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$;
  If $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) \neq f((\boldsymbol{u}_{\boldsymbol{X}} \oplus \boldsymbol{z}_{\boldsymbol{X}}) \circ \boldsymbol{w})$ then reject;
end
Accept;

C.3. The analysis when f is a k-junta

Throughout this subsection we assume that $f : \{0,1\}^n \to \{0,1\}$ is a k-junta, and let J be its set of relevant variables with $|J| \leq k$. The following lemma implies that after Phase A of Algorithm 4, with probability at least $2/3$, the partition satisfies $|\boldsymbol{X}_i \cap J| \leq 1$ for all $i \in [r]$.

Lemma 28 *With probability at least $2/3$, we have that $|\boldsymbol{X}_i \cap J| \leq 1$ for all $i \in [r]$.*

Proof Fix any $j_1, j_2 \in J$. The probability that they lie in the same block is $1/r$. By a union bound and the assumption that $|J| \leq k$, the probability of $|\boldsymbol{X}_i \cap J| > 1$ for some i is at most

$$\binom{k}{2} \cdot \frac{1}{r} \leq \frac{1}{3}.$$

■

Next we consider Phases B and C of the algorithm assuming that$|\boldsymbol{X}_i \cap J| \leq 1$ for all $i \in [r]$.

Lemma 29 *Assume that $|\boldsymbol{X}_i \cap J| \leq 1$ for all $i \in [r]$. Then Algorithm 4 always reaches Phase D and when this happens, we have $|\boldsymbol{X}_i \cap J| = 1$ for all $i \in I$.*

Proof On line 7, for ℓ to be added to I, we found v^ℓ such that

$$f(v^\ell) \neq f\left(\boldsymbol{w}_{\boldsymbol{X}_\ell} \circ v^\ell_{\overline{\boldsymbol{X}_\ell}}\right)$$

and thus, $|\boldsymbol{X}_\ell \cap J|$ must be exactly 1. As such, for every $\ell \in I$, we have that the function f^ℓ is a literal.

If Algorithm 4 fails to reach Phase D, then it must have halted on line 7, 11 or 12. If it halts on line 7, then $|I| > k$, but this implies $|J| > k$ which is a contradiction. If it halts on line 11, then since UNIFORMJUNTA has one-sided error it must be the case that f^ℓ is not a 1-junta, but we know f^ℓ is a literal (so it is a 1-junta). And if halts on line 12, then $f^\ell(\boldsymbol{b}) = f^\ell(\overline{\boldsymbol{b}})$ which contradicts the fact that f^ℓ is a literal. So in all cases, we reach a contradiction. ■

Lemma 30 *Assume that $|\boldsymbol{X}_i \cap J| \leq 1$ for all $i \in [r]$. Then the algorithm always reaches line 29 and outputs "accept."*

Proof By Lemma 29, Algorithm 4 reaches Phase D. To show that it reaches the last line, we need to show that it does not halt on line 23 or 27.

For every $\ell \in I$ we have that $\boldsymbol{X}_\ell$ contains exactly one relevant variable in J, which we denote by $x_{\tau(\ell)}$. So we have that f^ℓ is a literal; in particular, either $f^\ell(x) = x_{\tau(\ell)}$ or $f^\ell(x) = \overline{x_{\tau(\ell)}}$. Since $\boldsymbol{Y}_{\ell,0}, \boldsymbol{Y}_{\ell,1}$ is a partition of $\boldsymbol{X}_\ell$, $\tau(\ell)$ is in exactly one of $\boldsymbol{Y}_{\ell,0}$ or $\boldsymbol{Y}_{\ell,1}$. Suppose w.l.o.g. that $\tau(\ell) \in \boldsymbol{Y}_{\ell,0}$, then for any $b = b^0 \circ b^1$ with $b^i \in \{0,1\}^{\boldsymbol{Y}_{\ell,i}}$ we have

$$f^\ell(b^0 \circ b^1) \neq f^\ell(\overline{b^0} \circ b^1) \quad \text{and} \quad f^\ell(b^0 \circ b^1) = f^\ell(b^0 \circ \overline{b^1}).$$

Therefore, we always have $\boldsymbol{G}_{\ell,0} = h$ and $\boldsymbol{G}_{\ell,1} = 0$. Thus, the algorithm reaches line 26.

Next, observe that if $\tau(\ell) \in \boldsymbol{Y}_{\ell,0}$, then we set $\boldsymbol{z}_{\boldsymbol{X}_\ell} = \boldsymbol{w}_\ell$ and thus, $\boldsymbol{z}_{\tau(\ell)} = \boldsymbol{w}_{\tau(\ell)} = 0$ (and if we had $\tau(\ell) \in \boldsymbol{Y}_{\ell,1}$, then we would set $\boldsymbol{z}_{\boldsymbol{X}_\ell} = \overline{\boldsymbol{w}_\ell}$ and thus, $\boldsymbol{z}_{\tau(\ell)} = \overline{\boldsymbol{w}_{\tau(\ell)}} = 0$).

So, for every $\ell \in I$ we have $\boldsymbol{z}_{\tau(\ell)} = 0$ and $\tau(\ell)$ is the only relevant variable in $\boldsymbol{X}_\ell$. So $\boldsymbol{u}_{\boldsymbol{X}}$ and $\boldsymbol{u}_{\boldsymbol{X}} + \boldsymbol{z}_{\boldsymbol{X}}$ have the same value on all the relevant variables J in $\boldsymbol{X}$. As such we must always have $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{v}) = f((\boldsymbol{u}_{\boldsymbol{X}} \oplus \boldsymbol{z}_{\boldsymbol{X}}) \circ \boldsymbol{v})$ so the algorithm reaches the last line and accepts. ■

Lemma 31 *If f is a k-junta then Algorithm 4 outputs "accept" with probability at least $2/3$.*

Proof The result follows from Lemma 28, Lemma 29, and Lemma 30. ■

C.4. The analysis when f is ε-far in relative distance from every k-junta

Throughout this subsection we assume that $\text{rel-dist}(f, g) > \varepsilon$ for every k-junta $g : \{0,1\}^n \to \{0,1\}$. Our goal is to show that in this case, Algorithm 4 rejects with high probability.

For Phase A of the algorithm, we just fix any r-way partition $X_1, \ldots, X_r$ of $[n]$.Let us consider Phase B of the algorithm. It is clear that if Algorithm 4 passes Phase B (instead of being rejected on line 7), then at the moment when it reaches Phase C we must have $t = T = O(\log(k/\varepsilon))$ and $|I| \leq k$. Furthermore, we now show that the following condition is unlikely to happen:

Lemma 32 *The probability of Algorithm 4 reaching Phase C with*

$$\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{\boldsymbol{X}}}}}\left[f(\boldsymbol{u}_{\boldsymbol{X}}\circ\boldsymbol{w}) \neq f(\boldsymbol{u})\right] \geq \varepsilon/20$$

is at most $1/15$.

Proof For Algorithm 4 to reach Phase C, it must be the case that it has drawn $\boldsymbol{u} \sim f^{-1}(1)$ and $\boldsymbol{w} \sim \{0,1\}^{\overline{\boldsymbol{X}}}$ for T times and they satisfy $f(\boldsymbol{u}_{\boldsymbol{X}} \circ \boldsymbol{w}) = f(\boldsymbol{u})$ every time. But given the condition above, the probability that this happens is at most $(1 - \varepsilon/20)^T$, which is at most $\frac{1}{15k}$ by setting the hidden constant in $T = O(\log(k/\varepsilon))$ sufficiently large. By a union bound (over the at most k many different $\boldsymbol{X}$'s that are in play across all the executions of line 5), the probability of Algorithm 4 reaching Phase C with the condition above is at most $1/15$. ■

Now assume that Algorithm 4 reaches Phase C with an X that satisfies

$$\Pr_{\substack{\boldsymbol{u}\sim f^{-1}(1)\\ \boldsymbol{w}\sim\{0,1\}^{\overline{X}}}}\left[f(\boldsymbol{u}_X\circ\boldsymbol{w}) \neq f(\boldsymbol{u})\right] \leq \varepsilon/20. \tag{16}$$

Lemma 33 *If for some $\ell \in I$, $f^\ell : \{0,1\}^{X_\ell} \to \{0,1\}$ is $(1/30)$-far from every literal with respect to the uniform distribution, then Algorithm 4 rejects in Phase C with probability at least* $14/15$.

Proof If f^ℓ is $(1/30)$-far from every literal with respect to the uniform distribution then it is either

1. $(1/30)$-far from every 1-junta,
2. or $(1/30)$-far from every literal but $(1/30)$-close to a 0-junta (i.e., a constant-0 or 1 function).

In case 1, by Theorem 11, with probability at least $14/15$, UniformJunta($f^\ell, 1, 1/30, 1/15$) rejects, in which case Algorithm 4 rejects. In case 2, f^ℓ is $(1/30)$-close to a constant function; say w.l.o.g. that it is close to the constant-0 function. The probability we fail to reject on line 12 is :

$$\Pr_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\boldsymbol{b}) \neq f^\ell(\overline{\boldsymbol{b}})\right] \leq \Pr_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\overline{\boldsymbol{b}}) = 1\right] + \Pr_{\boldsymbol{b}\sim\{0,1\}^{X_\ell}}\left[f^\ell(\boldsymbol{b}) = 1\right] \leq 1/15,$$

where the last inequality follows from the fact f^ℓ is $(1/30)$-close to the constant-0 function under the uniform distribution and $\boldsymbol{b}$, $\overline{\boldsymbol{b}}$ are both sampled uniformly at random. ■

Next assume that Algorithm 4 reaches Phase D with X satisfying Equation (16) and I satisfying that f^ℓ is $(1/30)$-close to a literal with respect to the uniform distribution for every $\ell \in I$. For each $\ell \in I$, let $\tau(\ell) \in X_\ell$ be the (unique) variable such that f^ℓ is $(1/30)$-close to either $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$.

Lemma 34 *For each execution of the inner loop on ℓ in Phase D (i.e., lines 17–24), the probability that $\boldsymbol{z}_{\tau(\ell)}$ is set to 1 on line 24 (in which case the algorithm does not reject on line 23) is at most $(1/15)^h$.*

Proof Fix some $\ell \in I$, and assume without loss of generality that f^ℓ is $(1/30)$-close to $x_{\tau(\ell)}$ under the uniform distribution. The case where f^ℓ is close to $\overline{x_{\tau(\ell)}}$ is similar. Let $\boldsymbol{Y}_{\ell,0}, \boldsymbol{Y}_{\ell,1}$ be the partition of X_ℓ drawn in the loop. Then $\tau(\ell)$ is in exactly one of $\boldsymbol{Y}_{\ell,0}$ or $\boldsymbol{Y}_{\ell,1}$. Assume without loss of generality that $\boldsymbol{s}_{\tau(\ell)} = 1$ so $\tau(\ell) \in \boldsymbol{Y}_{\ell,1}$; the case when $\tau(\ell) \in \boldsymbol{Y}_{\ell,0}$ is similar. We have that $\boldsymbol{z}_{\tau(\ell)} = 1$ only if we set $\boldsymbol{z}_{\boldsymbol{X}_\ell} \leftarrow \boldsymbol{s}_{\boldsymbol{X}_\ell}$, meaning $\boldsymbol{G}_{\ell,0} = h$ is a necessary condition for $\boldsymbol{z}_{\tau(\ell)} = 1$. So, it suffices to show that $\mathbf{Pr}[\boldsymbol{G}_{\ell,0} = h] \le (1/15)^h$.

Focusing on a single execution of the loop spanning lines 19–21, we have

$$\mathbf{Pr}\left[f^\ell(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq f^\ell(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1)\right] \le \mathbf{Pr}\left[f^\ell(\boldsymbol{b}^0 \circ \boldsymbol{b}^1) \neq \boldsymbol{b}^1_{\tau(\ell)}\right] + \mathbf{Pr}\left[f^\ell(\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1) \neq \boldsymbol{b}^1_{\tau(\ell)}\right] \le 1/15,$$

where the second inequality follows from the assumption that f^ℓ is $(1/30)$-close to $x_{\tau(\ell)}$ under the uniform distribution and the marginal distribution of both $\boldsymbol{b}^0 \circ \boldsymbol{b}^1$ and $\overline{\boldsymbol{b}^0} \circ \boldsymbol{b}^1$ is uniform. It follows that $\mathbf{Pr}[\boldsymbol{G}_{\ell,0} = h] \le (1/15)^h$. ■

We are now ready to show that the algorithm accepts with probability at most $1/3$.

Lemma 35 *If $\text{rel-dist}(f, g) > \varepsilon$ for every k-junta g, then Algorithm 4 rejects with probability at least $2/3$.*

Proof By Lemma 32 and Lemma 33, we have that with probability at least $13/15$, Algorithm 4 either already rejects in Phases A–C or reaches Phase D with X and I satisfying Equation (16) and f^ℓ being $(1/30)$-close to a literal (either $x_{\tau(\ell)}$ or $\overline{x_{\tau(\ell)}}$) with respect to the uniform distribution for every $\ell \in I$. Assume that the latter happens. We show below that Phase D of Algorithm 4 rejects with probability at least $13/15$. It follows from a union bound that overall Algorithm 4 accepts with probability at most $2/15 + 2/15 < 1/3$.

To this end, by Lemma 34 and a union bound, with probability at least $1 - M|I|(1/15)^h$, either Algorithm 4 rejects on some execution of line 23 or $\boldsymbol{z}_{X_\ell}$ satisfies $\boldsymbol{z}_{\tau(\ell)} = 0$ for every ℓ and in every one of the M loops. By making the hidden constant in h sufficiently larger than the hidden constant in M, we can make

$$1 - M|I|(1/15)^h \ge 1 - O(k/\varepsilon) \cdot (1/15)^h \ge 14/15.$$

Assuming that this happens and Algorithm 4 does not reject on line 23, then for each of the M loops, we claim that the joint distribution of $(\boldsymbol{u}_X \circ \boldsymbol{w}, (\boldsymbol{u}_X \oplus \boldsymbol{z}_X) \circ \boldsymbol{w})$ is exactly the same as the joint distribution of $(\boldsymbol{u}_X \circ \boldsymbol{w}, \boldsymbol{u}_J \circ \boldsymbol{y} \circ \boldsymbol{w})$ in Lemma 27 with $J = \{\tau(\ell) : \ell \in I\}$. To see this, first observe that the joint distributions of the $\overline{X}$ coordinates (corresponding to $\boldsymbol{w}$) are clearly the same in both cases, so it suffices to consider coordinates in X. We observe that for any $\tau(\ell) \in J$, we have $\left((\boldsymbol{u}_X \oplus \boldsymbol{z}_X) \circ \boldsymbol{w}\right)_{\tau(\ell)} = \boldsymbol{u}_{\tau(\ell)} \oplus \boldsymbol{z}_{\tau(\ell)} = \boldsymbol{u}_{\tau(\ell)}$, since $\boldsymbol{z}_{\tau(\ell)} = 0$ by assumption. Turning to the coordinates in $X \setminus J$, fix any $i \in X_\ell, \ell \in I$ with $i \neq \tau(\ell)$. We have $\left((\boldsymbol{u}_X \oplus \boldsymbol{z}_X) \circ \boldsymbol{w}\right)_i = \boldsymbol{u}_i \oplus \boldsymbol{z}_i$, but $\boldsymbol{z}_i = \boldsymbol{s}_i$ if $\boldsymbol{s}_{\tau(\ell)} = 0$ and $\boldsymbol{z}_i = \overline{\boldsymbol{s}_i}$ otherwise. Since $\boldsymbol{s}_i \sim \{0,1\}$ and is independent of $\boldsymbol{s}_{\tau(\ell)}$, we thus have that $\boldsymbol{z}_i \sim \{0,1\}$ as well. So $\left((\boldsymbol{u}_X \oplus \boldsymbol{z}_X) \circ \boldsymbol{w}\right)_i$ is uniformly random, which matches the distribution of $(\boldsymbol{u}_J \circ \boldsymbol{y} \circ w)_i$ as desired.

As a result, Algorithm 4 rejects on line 27 with probability at least $1 - (1 - \varepsilon/5)^M \ge 14/15$ by making the constant hidden in M sufficiently large. Therefore, Algorithm 4 rejects in Phase D with probability at least $13/15$. This finishes the proof of the lemma. ■