Electronic Colloquium on Computational Complexity, Report No. 312 (2026)

The Space Complexity of Property Testing

Clément Canonne* Xi Chen† William Pires‡ Toniann Pitassi§
Rocco Servedio¶

November 4, 2026

Abstract

We initiate the study of the space complexity of property testing. We focus on five properties of Boolean functions that have been intensively studied in the property testing literature: monotonicity, symmetry, linearity, being a junta, and being a linear threshold function.

On the upper bounds side, we show — perhaps surprisingly — that for most of these properties there exist new testing algorithms that use dramatically less space than the well-known testers for these properties, which have been designed to use as few queries as possible. For each of these five properties we also prove an unconditional space lower bound for any tester, regardless of its runtime or query complexity, that either matches or almost matches the space used by our new testers. Our results thus give an essentially complete characterization of the space complexity of testing these fundamental properties.

*The University of Sydney. Email: clement.canonne@sydney.edu.au.
†Columbia University. Email: xc2198@columbia.edu.
‡Columbia University. Email: wp2294@columbia.edu.
§Columbia University. Email: tp2684@columbia.edu.
¶Columbia University. Email: rocco@cs.columbia.edu.

ISSN 1433-8092

Contents

1 Introduction **1**
- 1.1 Some motivating examples and questions . . . 1
- 1.2 Our testing framework . . . 2
- 1.3 Main results . . . 2
- 1.4 Discussion and future work . . . 3
- 1.5 Related work . . . 5
- 1.6 AI Methodology . . . 6

2 Technical Overview **6**
- 2.1 Upper bounds . . . 6
- 2.2 Lower bounds . . . 8

3 Preliminaries **14**
- 3.1 The properties we study . . . 14

4 The Bounded-Memory Model **15**
- 4.1 Memory-bounded testers . . . 15
- 4.2 Markov chain results for lower bounds. . . . 16

5 Upper Bounds **17**
- 5.1 Space upper bound for monotonicity . . . 17
- 5.2 Space upper bound for symmetry . . . 27
- 5.3 Space upper bound for linearity . . . 27
- 5.4 Space upper bound for k-juntas . . . 27
- 5.5 Space upper bound for LTFs . . . 28

6 Lower Bounds **31**
- 6.1 A framework for symmetry and monotonicity lower bounds . . . 31
- 6.2 The lower bound for symmetric functions . . . 35
- 6.3 Space lower bound for monotonicity . . . 36
- 6.4 A space lower bound for linearity . . . 39
- 6.5 A space lower bound for k-juntas . . . 45
- 6.6 A space lower bound for LTFs . . . 46

A General tools **52**
- A.1 The Markov chain tree theorem . . . 52
- A.2 From relative error to close long-term probabilities . . . 53
- A.3 Normal form . . . 54

1 Introduction

The goal of a *property testing* algorithm is to determine whether an unknown object has some global property of interest, or is far from having that property. Over the past several decades the study of property testing has blossomed into a rich and active subfield of theoretical computer science, see e.g. [Gol10, Gol17, BY22]. One reason why property testing has flourished as a research topic is because it has many connections to other areas such as sublinear algorithms, graph theory, Boolean function analysis, complexity theory, and more. In particular, the main goal in the field of property testing is to understand the minimum *query complexity* that is required for testing various properties, which is very much in the spirit of complexity theory.

However, query complexity is only one part of the computational cost of a property tester. A number of papers have considered other computational resources for property testing algorithms, such as randomness complexity [GS10], number of rounds of adaptivity [CG18], and time complexity [FPPR26]; we give an overview of these and related prior works in Section 1.5. The purpose of this paper is to initiate the investigation of a computational resource which, curiously, has been overlooked in the study of property testing, namely *space complexity*. Our main question is:

> *How much memory is needed to test a property (when there is no restriction on the running time or the number of queries)?*

1.1 Some motivating examples and questions

In this paper we focus on testing properties of Boolean functions $f\colon \{0,1\}^n \to \{0,1\}$. Let us begin by recalling some of the best-known Boolean function property testing problems and algorithms.

- One of the earliest problems studied in property testing is testing whether f is a *linear* function, i.e., whether $f(x) = a \cdot x \mod 2$ for some unknown vector $a \in \{0,1\}^n$. The famous constant-query linearity tester given in the pioneering work of Blum, Luby and Rubinfeld [BLR93], which queries triples of strings $(x, y, x \oplus y)$, must store $\Theta(n)$ bits to construct the query string $x \oplus y$ after querying $f(x)$ and $f(y)$, so its space complexity is $\Omega(n)$. Is there a more space-efficient testing algorithm for linearity?

- Another long-studied property testing problem is *monotonicity testing*, in which the goal is to test whether $f\colon \{0,1\}^n \to \{0,1\}$ is a monotone Boolean function. A number of algorithms [GGL+00, CS13, CST14, KMS18] with better and better query complexities have been given over the years, and recent work [CCC+26, Min25] shows that the $\tilde{O}(\sqrt{n})$ query complexity of the [KMS18] algorithm is essentially the best possible. All of these algorithms work by repeatedly first querying some point x and then some point y which lies above x in the Boolean lattice $\{0,1\}^n$. Constructing such a y requires storing x, and hence known monotonicity testing algorithms also have space complexity $\Omega(n)$. Can this space bound be improved?

- A third flagship property testing problem that has been intensively studied [FKR+04, Bla08, Bla09] is testing whether $f\colon \{0,1\}^n \to \{0,1\}$ is a *k-junta*, i.e., whether f depends on (an unknown set of) at most k input coordinates. All of the above-mentioned junta-testing algorithms have $\mathrm{poly}(k)$ query complexity (independent of n), and all employ an initial stage of randomly partitioning the input coordinates $x_1, \dots, x_n$ into $\mathrm{poly}(k)$ different buckets, and hence use space $\Omega(n \cdot \log k)$ to store this partition across the execution of the algorithm. Can k-juntas be tested using less space?

1.2 Our testing framework

To study questions of the above sort, we introduce a bounded-memory model for property testing that places no limit on the running time or query complexity of testing algorithms. A testing algorithm in our model may choose its queries adaptively, but all information it retains between queries must fit within its memory budget. (This includes any record of how many queries it has made, i.e., the tester is not given access to an external clock; however, by storing the current time step $t \leq T$ using $\log T$ bits of the memory, it is possible to simulate having such a clock. We will return to this point later.) A memory lower bound for a property in our model means that a tester with too little memory cannot correctly test the property, even if it is allowed to have an arbitrary running time per query and to make an arbitrary number of queries to the function being tested.

In more detail, an M-state (or equivalently, a $\lceil \log M \rceil$-space) ε-testing algorithm $\mathcal{A}$ for a property $\mathcal{P}$ is a finite-state randomized query algorithm that has oracle access to its input $f\colon \{0,1\}^n \to \{0,1\}$. From its current state, the algorithm chooses a query point $x \in \{0,1\}^n$ according to a probability distribution which is determined by its current state; observes $f(x)$; and moves to a new state according to transition probabilities which depend on $x, f(x)$, and the current state. It is not hard to see that given a space-bounded testing algorithm $\mathcal{A}$ in our model, when the algorithm is run on a fixed function f the resulting transition diagram $\mathcal{A}_f$ is a Markov chain. Every state of $\mathcal{A}$ is labeled as either an accept or reject state, and there are no halting states; a valid tester $\mathcal{A}$ must have *long-run average acceptance probability*

$$\lim_{T\to\infty} \sum_{t=0}^{T-1} \frac{1}{T} \mathbf{Pr}[\mathcal{A}_f \text{ is in a state labeled accept at time } t]$$

close to 1 when f has property $\mathcal{P}$, and must have long-run average acceptance probability close to 0 for every f that is ε-far from property $\mathcal{P}$. (See Section 4.1 for detailed definitions.)

Our model occupies a different point in the landscape of memory-bounded computation than existing models. Probabilistic streaming algorithms and probabilistic finite automata give the algorithm no control over the order in which different positions in the input are accessed, and in the recent study of memory-bounded learning (see, e.g., [Raz17, Raz19, GRT18]) the focus is on time-space tradeoffs for learning algorithms which do not have query access to the function being learned. In contrast, in our model the testing algorithm has query access to the function being tested and can run for arbitrarily many steps. (We explain in Section 1.4 how time-space tradeoffs for a related property testing model can be derived from the lower bound results that we give in our model.) We give further discussion of related work in Section 1.5.

1.3 Main results

The goal of this work is to understand the space complexity of testing some of the most intensively studied classes of Boolean functions in the property testing literature.[1] In addition to the three properties of linearity, monotonicity and being a k-junta mentioned earlier, we also consider two other fundamental properties. The first is *linear separability*, i.e., being a Linear Threshold Function (LTF) $f(x) = \mathrm{sign}(w \cdot x - \theta)$ (a.k.a. a halfspace); this notion is of course fundamental to machine learning theory and has been studied from the property testing perspective in papers

[1] A forthcoming companion paper [CCP+26] builds more general theory about space-bounded property testing that does not deal with specific properties.

such as [MORS10, BBBY12, MN15, Har19]. The second property is *symmetry*, i.e., being a Boolean function which depends only on the Hamming weight $x_1 + \cdots + x_n$ of the input x. This property has been studied in a number of property testing works including [DSS11, BWY15, AHW16].

As our main results, we obtain near-matching upper and lower bounds on the space complexity of testing all five of these properties:

Main Theorem: For each of the properties of Linearity, Monotonicity, k-Juntas, Symmetry, and Linear Thresholds, the upper and lower bounds on space complexity of testing the property are as listed in Table 1.

Property	**Space Complexity**
Linearity	$\Theta(n)$ (Theorem 30, Theorem 42)
Monotonicity	$\tilde{\Theta}(\sqrt{n})$ (Theorem 14, Theorem 41)
k-Juntas	$\Theta(k \log n)$ (Theorem 32, Theorem 57)
Symmetry	$\Theta(\log n)$ (Theorem 29, Theorem 38)
Linear Threshold Functions	$\Theta(\log n)$ (Theorem 33, Corollary 59)

Table 1: Space complexity of testing parity, monotonicity, k-juntas, symmetry, and LTFs with a sufficiently small constant distance parameter ε. The bound for k-Juntas assumes $k \leq n^{0.99}$.

Returning to the three questions that were posed in Section 1.1, we see that there is an intriguing diversity across the three answers. For linearity our results show that the $\Theta(n)$-bit space complexity of the standard [BLR93] algorithm is best possible up to constant factors. For k-juntas the answer is very different: a (simple) new testing algorithm achieves space complexity $O(k \log n)$ rather than the $\Theta(n \log k)$ of the standard query-efficient testing algorithms in the literature, and we show that this $O(k \log n)$ space upper bound is best possible up to constant factors. And for monotonicity the answer lies in the middle between these two extremes of large (linear) space and small (logarithmic) space: we give an $\tilde{O}(\sqrt{n})$-space algorithm and also prove an almost-matching $\Omega(\sqrt{n})$ space lower bound. Taken together, our results show that space complexity provides a new axis along which property testing problems and property testers can be analyzed and understood.

1.4 Discussion and future work

Our model and results suggest a number of directions for further work.

Query/space tradeoffs for property testing? Recall that in our space-bounded testing framework as described earlier there are no halting states, and so in some sense our testing algorithms never halt and thus we cannot discuss their query complexity. (This paper adopts this framework because it is a particularly strong one within which to prove space lower bounds.) However, for each of the five (essentially) space-optimal testing algorithms that we present, there is a natural variant of the algorithm, with the same space complexity, that always enters either an accepting or rejecting sink state after finitely many queries. (Indeed, in describing our algorithms it will be more natural to present these variants, which easily translate into algorithms in our infinite-time framework.) For these variants it is perfectly reasonable to consider their query complexity. A natural question,

then, is how the query complexities of these space-optimal variants compare with the optimal query complexities for these properties. (Recall that for each of the five properties that we study, a fairly tight bound on the optimal query complexity is known in the literature, see Table 2.)

Property	**Query Complexity**
Parity	$\Theta(1)$ [BLR93]
Monotonicity	$\tilde{O}(\sqrt{n})$ [KMS18], $\Omega(n^{1/2-o(1)})$ [CCC+26, Min25]
k-Juntas	$\tilde{\Theta}(k)$ [Bla09, BBM12]
Symmetry	$\Theta(1)$ [DSS11, BWY15]
Linear Threshold Functions	$\Theta(1)$ [MORS10]

Table 2: Query complexity of testing parity, monotonicity, k-juntas, symmetry, and LTFs with a constant distance parameter ε (say, $\varepsilon = 1/100$). The bound for k-Juntas assumes $k \leq n^{0.99}$.

It turns out that for linearity, the original [BLR93] query-efficient algorithm also has optimal ($\Theta(n)$) space complexity, see Section 6.4. Similarly for symmetry, the standard tester [DSS11, BWY15] turns out to have the optimal $\Theta(\log n)$ space complexity, see Section 6.2. For the other three properties, though, the new space-optimal algorithms that we give have sub-optimal query complexities. Our $O(k \log n)$-space k-junta tester makes roughly n^k queries (see Section 5.4) which is worse than the $\tilde{\Theta}(k)$ queries of the query-optimal tester; our $O(\log n)$-space LTF tester makes $\mathrm{poly}(n)$ queries (see Section 5.5) which is worse than the $O(1)$ queries of the query-optimal tester; and our $\tilde{O}(\sqrt{n})$-space monotonicity tester makes $2^{\tilde{\Theta}(\sqrt{n})}$ queries which is worse than the roughly $\sqrt{n}$ queries of the query-optimal tester. It would be interesting to map the landscape of which properties do (and do not) have testers that are simultaneously query-optimal and space-optimal.

We remark that the forthcoming paper mentioned earlier [CCP+26] formally defines a finite-query variant of our space-bounded testing model in which the testing algorithm has a "clock" that records how many queries have been made so far. Since, as was noted earlier, in our current paper's framework it is possible to simulate a clock for T queries using $\log T$ bits of memory, the lower bounds we prove in this paper already imply some query/space tradeoffs for the finite-query model with a clock. For example, for a sufficiently small absolute constant $c > 0$, Theorem 57 implies that any $ck \log n$-space, clock-equipped k-junta tester must make $n^{\Omega(k)}$ queries; Corollary 59 implies that any $c \log n$-space, clock-equipped LTF tester must make $n^{\Omega(1)}$ queries; and Theorem 41 implies that any $c\sqrt{n}$-space, clock-equipped monotonicity tester must make $2^{\Omega(\sqrt{n})}$ queries.

A characterization of the space complexity of property testing? Is there a combinatorial or information-theoretic quantity, analogous in spirit to VC dimension, sample compression scheme size, or communication complexity, that determines the amount of memory that is needed to test a property of Boolean functions? Our lower-bound arguments (sketched in Section 2) suggest that one relevant parameter may be the extent to which a property admits "local corruptions" that preserve all weighted statistics that can be estimated by a small-state tester.

A general theory of space-bounded testing. Finally and more broadly, a natural goal for future work is to develop a body of general theoretical results about space-bounded property testing that apply to all Boolean function properties. As mentioned earlier, a forthcoming companion paper takes some initial steps along these lines [CCP+26].

1.5 Related work

At a big-picture level, as mentioned earlier the vast majority of research on property testing has focused almost exclusively on query complexity. There have been a number of works on property testing that have considered other computational resources besides query complexity (we briefly survey some of these below), but to the best of our knowledge our space-bounded model has not previously been considered for property testing.

- Starting with runtime, an early example is the work of [DLM+11] on testing s-sparse $\mathbb{F}_2$-polynomials, which gave a testing algorithm for this class with $\mathrm{poly}(s)$ runtime (improving on the $\exp(\Omega(s))$ runtime of prior work [DLM+07]). More recently, various papers on testing decision trees and juntas have studied the runtime of testing algorithms, either by giving runtime-efficient testers [BHZ22, BLT22] or runtime lower bounds under assumptions such as randomized ETH or NP-hardness [KST23b, KST23a]. Quite recently, [FPPR26] initiated a systematic study of the relation between runtime and query complexity of property testing, establishing time-query hierarchy theorems and studying the time complexity of tolerant LTF testing. None of these works considered space as a computational resource for property testers.

- The randomness complexity of property testing was studied (from a number of different perspectives) in works including [GS10, BSSVW03, Zim07]. The number of rounds of adaptivity used for property testing (which may also be viewed as a resource) was studied in [CG18], and starting with [CHFSV19] a number of papers have studied round complexity for property testing in distributed settings such as the CONGEST model.

- Finally, turning to space complexity, a number of papers starting with [FKSV02] have studied the streaming complexity of property testing problems [FMdRS16, MMPS17, CFPS20, BS21, ABC+26]. While the streaming model is related to space complexity, it is very different from our model because in our setting the testing algorithm can actively choose query points. We also mention [LNS08], which shows that for many space bounds $s(n)$, there exists a language with space complexity $s(n)$ for which any testing algorithm must have query complexity at least $2^{\Omega(s(n))}$. Since [LNS08] considers the space complexity of *language recognizers* rather than testing algorithms, it addresses a very different question than our paper.

Looking beyond the property testing literature, our model is closely related to the classical finite-memory model that Cover and Hellman studied for the binary hypothesis testing problem (see [CH70], which was influential in the subsequent development of a large literature on distribution learning and testing which is surveyed in [BOS22]). In their model an algorithm receives an unbounded sequence of i.i.d. observations from an unknown distribution while maintaining one of only M internal states, each of which is labeled as either "null hypothesis" or "alternative hypothesis." As in our model the next state is a (randomized) function only of the current state and the current observation and not of the current time step, and as in our model performance is measured by the long-run average probability of error as the number of time steps tends to infinity. There is, however, an important difference in how information is obtained. In the Hellman–Cover hypothesis-testing model observations arrive according to i.i.d. draws from the fixed distribution that is being tested, whereas in our model of Boolean function testing the tester actively chooses which input to query as a function of its current state.

1.6 AI Methodology

The authors used ChatGPT (5.6 and 6) and Claude (Fable 5.1) in this project.

The question, model, and subproblems studied in this work are solely due to the authors. AI tools were used for literature search in the early stage of the project. All the upper bounds except monotonicity were produced by the authors.

On monotonicity, the authors produced an $\Omega(\sqrt{n})$ lower bound but did not believe it was tight. Eventually, they obtained an algorithm that uses $n-\omega(1)$ bits of memory. After realizing that using fewer than n bits was possible, they asked ChatGPT to improve the upper bound, which produced the $\tilde{O}(\sqrt{n})$ space upper bound for monotonicity. They considerably simplified and rewrote the proof it produced, leading to the argument presented in Section 5.1.

AI tools were used to fine-tune the lower bounds. They were also used to proofread and assist with exposition. All text in this work was written by the authors. They of course take responsibility for the correctness of all content in this paper.

2 Technical Overview

In this section we give a high-level overview of our space-efficient testing algorithms and our space lower bounds. Throughout the discussion, for the sake of simplification we often set the distance parameter ε for the property tester to be an unspecified absolute constant such as $\varepsilon = 1/100$.

2.1 Upper bounds

We discuss the five algorithmic results for space-efficient testing in increasing order of how complicated and novel the algorithms and the analyses of their space complexity are. We remind the reader that our lower bounds (surveyed later, in Section 2.2) imply that all these algorithms are space-optimal up to constant factors (except for the final algorithm, for testing monotonicity, which is shown to be space-optimal up to a polylog(n) factor).

Linearity and symmetry. The first two algorithms are long-known results. For linearity, i.e. testing whether f is a parity function over some unknown subset of variables, the standard [BLR93] algorithm uses $O(n)$ bits of memory (see Theorem 30). Similarly, for symmetry (testing whether f is a symmetric function), the standard algorithm of [DSS11, BWY15] uses $O(\log n)$ bits of memory (see Theorem 29). It is interesting to note that while there are 2^n parities and 2^{n+1} symmetric functions, and both properties are testable using constantly many queries, the space complexities of these two properties are very different from each other.

Juntas and LTFs. The next two space-optimal algorithms are new but only require rather straightforward approaches of trading space for time. For juntas, recall that the standard query-efficient k-junta tester of [Bla09] maintains a random partition of the n variables into poly(k) blocks, which requires $\Omega(n \log k)$ bits to store in memory. Instead, we maintain and update a candidate set $S \subseteq [n]$ of k coordinates (the coordinates for which the tester is currently checking "is f a k-junta that depends only on these coordinates?"). By trying all sets S, it is straightforward to show (see Section 5.4) that k-juntas can be tested using only $O(k \log n)$ bits of memory. Turning to LTFs, our new testing algorithm builds on the original [MORS10] query-efficient tester in a simple way. Roughly speaking, the [MORS10] tester works by estimating various Fourier statistics that are based on degree-1 Fourier coefficients using correlated tuples of query points, and storing n-bit

query strings in order to generate their correlated partners incurs a memory cost of $\Omega(n)$ bits. We modify the original [MORS10] algorithm to obtain a (less query-efficient) variant of it which scans the n input coordinates one at a time, and for each coordinate estimates the relevant Fourier coefficient using an $O(\log n)$-bit counter. The estimates are accumulated into a constant number of running aggregates; including these aggregates and the control information, the total memory usage is $O(\log n)$ bits. Details are given in Section 5.5.

Monotonicity. The most complicated — and most surprising — of our space-efficient algorithms is a new $\tilde{O}(\sqrt{n})$-space algorithm for monotonicity testing. This new tester goes beyond the above simple approach of trading space for time, and is qualitatively quite different from prior monotonicity testing algorithms.

For simplicity we will describe our algorithm as a clock-equipped one, which makes a finite number $T = 2^{\tilde{\Theta}(\sqrt{n})}$ of queries. (As noted earlier, maintaining such a clock uses only $\log T = \tilde{O}(\sqrt{n})$ bits of memory, which is within our memory budget.) Our algorithm will focus on estimating the *discrete derivatives* $D_i f$ of the function f, defined as

$$D_i f(x) := \frac{f(x^{i\leftarrow 1}) - f(x^{i\leftarrow 0})}{2}$$

for $i \in [n]$ and $x \in \{0,1\}^n$. If f is monotone then clearly $D_i f(x) \geq 0$ for every x and i; and if f is ε-far from monotone, then the analysis of the standard edge tester [GGL+00] implies that f has an $\Omega(\varepsilon/n)$ fraction of violating edges, or, equivalently,

$$\Pr_{\boldsymbol{x},\boldsymbol{i}}\left[D_{\boldsymbol{i}} f(\boldsymbol{x}) < 0\right] = \Omega\big(\varepsilon/n\big)\,.$$

Thus, if we had a space-efficient way to estimate the quantity $\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}[D_{\boldsymbol{i}} f(\boldsymbol{x}) < 0]$, we would be able to distinguish between the two cases. Unfortunately, the naive approach — repeatedly draw an edge $\boldsymbol{e} = (\boldsymbol{x}, \boldsymbol{i})$ uniformly at random, and compute $D_{\boldsymbol{i}} f(\boldsymbol{x})$ using two queries to f — does not work, since storing $\boldsymbol{e}$ (as we must do in order to make queries to both its endpoints) takes $\Omega(n)$ space.

A first idea would be to hard-code the edges that will be inspected and use the clock to iterate through the hard-coded edges. Doing this naively again does not work: there are $\tilde{\Omega}(2^n)$ edges to cycle through, leading to an exponential number of time steps, while we can only afford $T = 2^{\tilde{O}(\sqrt{n})}$. However, a Fourier analysis view offers some hope, starting from the well-known fact that the quantity $D_i f$ can be expressed as

$$D_i f(x) = -\sum_{\substack{S \subseteq [n] \\ S \ni i}} \widehat{f}(S)\chi_{S\setminus\{i\}}(x), \qquad x \in \{0,1\}^n. \tag{1}$$

It follows immediately from (1) that if f has low Fourier degree, then $D_i f$ is a low-degree polynomial. Moreover, recalling that the Fourier concentration of monotone functions implies that every monotone function is close to an $O(\sqrt{n})$-degree polynomial [BT96], one might hope that each $D_i f$ should "behave like" an $O(\sqrt{n})$-degree polynomial when f is monotone, and hence we might hope to estimate $\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}[D_{\boldsymbol{i}} f(\boldsymbol{x}) < 0]$ using only $2^{O(\sqrt{n})}$ queries by "only looking at the low-degree (Fourier) truncation" $f^{\leq\sqrt{n}}$ of f. However, this approach faces two challenges. The first is that it is not clear how to effectively get access to the low-degree truncation $f^{\leq\sqrt{n}}$ of f, which in general is not a Boolean function or even a function that takes values in $[-1,1]$. The second challenge is that while monotone functions are well-concentrated on the first $O(\sqrt{n})$ Fourier levels, they are not

well-concentrated *enough*: even if we could do it, truncating a monotone f to obtain the low-degree (real-valued) function $f^{\leq\sqrt{n}}$ might create many non-monotone edges. So even if we could estimate $\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}[D_{\boldsymbol{i}}f^{\leq\sqrt{n}}(\boldsymbol{x}) < 0]$ using few queries and little memory, it is not clear that this quantity would usefully distinguish between the low-degree truncation of a monotone function and the low-degree truncation of a far-from-monotone function.

Happily, a natural approach to dealing with the first challenge also turns out to help with the second one. The approach is to *apply noise* (more precisely, the standard Boolean noise operator T_ρ) to the Boolean function f; drawing on standard notions in Fourier analysis of Boolean functions, it can be shown that a suitably "noised" version $\mathrm{T}_\rho f$ of the original function f will take values in $[-1,1]$ and will have exponentially better Fourier concentration, while preserving monotonicity if f is monotone and maintaining nontrivial distance to monotonicity if f is far from monotone. Importantly, as we show, this "noising" of f can be performed "on-the-fly," given oracle access to the original function f, in low memory.

This leads to the following rough plan: First, check that f is (at least as) well-concentrated on its low-degree Fourier coefficients as a monotone function must be. (For technical reasons, we will actually perform a slightly stronger check in this first step, which implies Fourier concentration, namely checking that f has low *noise sensitivity*.) Then, assuming the first check passes, estimate $\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}[D_{\boldsymbol{i}}((\mathrm{T}_\rho f)^{\leq\sqrt{n}})(\boldsymbol{x}) < 0]$ with $\tilde{O}\big(\sqrt{n}\big)$ memory; note that this estimate is computed using the low-degree truncation of the noisy version $\mathrm{T}_\rho f$ of f rather than f itself.

In more detail, our tester will do the following (in a space-efficient way): setting $L \coloneqq \tilde{\Theta}\big(\sqrt{n}\big)$ and $\rho \coloneqq 1 - \tilde{\Theta}\big(1/L\big)$,

1. Check that f has sufficiently small noise sensitivity (at noise parameter ρ) as any honest-to-goodness monotone function must have. If not, enter a reject state.
2. Estimate
$$\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}\Big[\, D_{\boldsymbol{i}}((\mathrm{T}_\rho f)^{\leq L})(\boldsymbol{x}) < -3/(8n) \,\Big] \tag{2}$$
to accuracy $\pm 1/\mathrm{poly}(n)$, using a hard-coded set H of $2^{\tilde{O}(L)}$ edges (x,i).
3. Enter a reject state if this estimate is above some $\tau = 1/\mathrm{poly}(n)$, and an accept state otherwise.

A crucial aspect of the correctness argument is that the hard-coded set H of only $2^{\tilde{O}(L)}$ edges that is used in Step 2 is "universal:" it does not depend on f, and hence can be hard-coded and iterated through (using the clock). The existence of such an H is established using uniform convergence results together with the fact that for any monotone f, the function $\mathrm{T}_\rho f$ has extremely little Fourier weight above level $\tilde{\Theta}(\sqrt{n})$ (Corollary 26). So when all the dust settles, all that is needed (ignoring for now a number of technical details) is to maintain a counter to keep track of the estimate of Equation (2). Since this estimate is obtained by iterating over the elements of H, this takes only $O(\log|H|) = \tilde{O}\big(\sqrt{n}\big)$ bits. The remaining technical work for the argument lies in showing that all the steps described above (simulating query access to the noisy version of f, its truncation, and $D_i f_{\rho}^{\leq L}$) can indeed be implemented using $\tilde{O}\big(\sqrt{n}\big)$ bits of memory. We give all details in Section 5.1.

2.2 Lower bounds

Our lower bounds draw upon the well-developed theory of Markov chains. Let $\mathcal{A}$ denote a fixed testing algorithm for some property $\mathcal{P}$. Given any fixed input function f that is to be tested, the

transition diagram of $\mathcal{A}$ when it is run on f, which we denote $\mathcal{A}_f$, is a Markov chain on the state space of $\mathcal{A}$, and the output of the tester is determined by the long-run mass of its accepting states. We use the *Markov chain tree theorem* [LR83] to show that if two input functions f, g induce transition probabilities that are multiplicatively close for every pair of states, then $\mathcal{A}$ will have (almost) the same long-run probability of accepting on f and on g (see Lemma 10).

Interestingly, this leads to a lower bound paradigm that is quite different from the usual one in property testing. Standard query lower bounds typically (i) construct a fixed "yes"-distribution over functions with the property and a fixed "no"-distribution over functions that are far from having the property, and (ii) show that every algorithm making few queries fails to distinguish them because the distributions of the query/reply history $(x_1, \boldsymbol{f}(x_1)), (x_2, \boldsymbol{f}(x_2)), \ldots, (x_k, \boldsymbol{f}(x_k))$ are close for $\boldsymbol{f}$ drawn from the yes-distribution versus the no-distribution. In contrast, our testers may make an arbitrary number of queries and indeed may query the entire truth table of the functions, so this notion of transcript indistinguishability is no longer useful. Instead, we (i) first fix a bounded-memory testing algorithm $\mathcal{A}$, and (ii) then argue that there exist two input functions f, g such that f has the property and g is far from having the property, but the transition probabilities between every pair of states in the induced Markov chains $\mathcal{A}_f$, $\mathcal{A}_g$ are nearly indistinguishable. Here, by "nearly indistinguishable" we mean that for any pair of states s and s', the probabilities of taking the transition from s to s' in $\mathcal{A}_f$ and in $\mathcal{A}_g$ are *multiplicatively* close to each other.[2] Thus, in our lower bound arguments the hard instances depend on the testing algorithm $\mathcal{A}$, and are tailored to what this particular tester is "remembering."

Equipped with this framework, all of our lower bounds have the same high-level structure. Given a property $\mathcal{P}$ to test, we start with a distribution $\mathcal{D}_{\text{yes}}$ over yes-inputs (functions in $\mathcal{P}$). Then we fix a particular testing algorithm $\mathcal{A}$ with bounded memory. Based on $\mathcal{A}$, we remove some inputs from $\mathcal{D}_{\text{yes}}$ that are easy for $\mathcal{A}$, and for each surviving yes-input f, we construct a "distorted" version, $\tilde{f}$, of f such that (i) $\tilde{f}$ is far from every function in $\mathcal{P}$, but (ii) the transition probabilities for $\mathcal{A}_f$ and $\mathcal{A}_{\tilde{f}}$ are nearly indistinguishable.

Implementing this strategy is nontrivial, and depends on the property we are testing. Our lower bound arguments naturally fall into two distinct groups. The first group consists of our arguments for symmetry and monotonicity, which are quite similar to each other; first we present the lower bound in the simpler context of symmetry, and then we explain how the same idea can be adapted to give a lower bound for monotonicity. The second group consists of the lower bound arguments for linearity, k-juntas, and linear threshold functions. In Section 2.2.2 we first explain the lower bound for testing linearity, and then sketch how it is adapted to yield lower bounds for testing k-juntas and testing LTFs.

2.2.1 Lower Bound for Symmetry and Monotonicity

We start by developing some intuition for why any tester for symmetry "ought to" require $\Omega(\log n)$ bits of memory. The natural way to detect a violation of symmetry for a function f is to find two input strings x, y with the same Hamming weight but which have $f(x) \neq f(y)$. To implement such a check, it seems necessary to use $\Omega(\log(n))$ bits of memory to "remember" the Hamming weight of the first string x so that another string y with the same weight can be constructed and queried.

Let us make this intuition more formal (though the following description simplifies some aspects of our actual construction). Let $\mathcal{A}$ be a clockless tester using $M = n^\alpha$ states of memory for some

[2] Note that additive closeness of transition probabilities does not suffice; since the time scale is arbitrarily large, small additive differences can accumulate and lead to dramatically different long-run state distributions.

small constant α. For simplicity, we will assume that $\mathcal{A}$ has the following property: for each state s' of $\mathcal{A}$, there exists a bit $b_{s'} \in \{0,1\}$ such that $\mathcal{A}$ can only transition to s' when the response $f(x)$ to its current query string x is $b_{s'}$.[3]. We will write $\delta_{s,s'}(x)$ for

$$\mathbf{Pr}[\mathcal{A} \text{ queries } x \text{ when in state } s] \cdot \mathbf{Pr}[\mathcal{A} \text{ moves to state } s' \text{ when querying } x \text{ and receiving response } f(x) = b_{s'} \text{ in state } s].$$

For each weight layer ℓ of the hypercube (i.e., each set of points $B_\ell := \{x \in \{0,1\}^n : |x| = \ell\}$) we fix a "corruptible" set C_ℓ of half the points of B_ℓ (rounding down if necessary).

The yes distribution D_{yes}: A random yes-function drawn from D_{yes} is obtained by drawing a random string $\boldsymbol{c} = (\boldsymbol{c}_0, \ldots, \boldsymbol{c}_n) \in \{0,1\}^{n+1}$, and considering the function $f_{\boldsymbol{c}} \colon \{0,1\}^n \to \{0,1\}$ defined as

$$f_{\mathbf{c}}(x) := \mathbf{c}_{|x|}.$$

It is clear that $f_{\mathbf{c}}$ is a symmetric function for every outcome of $\boldsymbol{c}$.

To build our adversarial inputs against $\mathcal{A}$, we will only want to hide violations to symmetry in a layer ℓ if for every transition (s, s') of $\mathcal{A}$, the algorithm "doesn't put too much weight" on points with $|x| = \ell$. Specifically, for each transition (s, s') and weight layer $\ell \in \{0, 1, \ldots, n\}$, we define the *weight* of the layer for the transition to be

$$W_{s,s'}(\ell) := \sum_{x : |x| = \ell} \delta_{s,s'}(x).$$

Slightly hiding some complexity for now, we will call a layer $\ell^\star$ *heavy for* (s, s') if $\ell^\star$ is among the layers with the $\approx M^2$ largest values of $W_{s,s'}(\ell)$. Importantly, if ℓ is not heavy for (s, s') then its weight $W_{s,s'}(\ell)$ is only a small fraction of the transition weight of the heavy layers. I.e., we have

$$\frac{W_{s,s'}(\ell)}{\sum_{\ell^\star : \ell^\star \text{ is heavy for } s,\, s'} W_{s,s'}(\ell^\star)} = O\left(\frac{1}{M^2}\right). \tag{3}$$

We let $\mathcal{H} \subset \{0, 1, \ldots, n\}$ denote the set of all layers that are heavy for some transition (s, s'), and we refer to $\mathcal{H}$ as the set of *heavy layers* and to $\mathcal{L} := \{0, 1, \ldots, n\} \setminus \mathcal{H}$ as the set of *light layers*. Since $M = n^\alpha$, by taking α small enough we can guarantee that only a small fraction of all layers are in $\mathcal{H}$, and hence that most points of the hypercube are in some light layer. Now we are ready to build the no distribution:

The no distribution D_{no}: A random no-function drawn from D_{no} is obtained as follows: Draw a random string $\mathbf{c} \in \{0,1\}^{n+1}$, and consider the function $g_{\mathbf{c},\mathcal{L}} : \{0,1\}^n \to \{0,1\}$ defined as

$$g_{\mathbf{c},\mathcal{L}}(x) := \begin{cases} \mathbf{c}_{|x|} & \text{if } |x| \in \mathcal{H}; \\ \mathbf{c}_{|x|} & \text{if } |x| \in \mathcal{L} \text{ and } x \notin C_{|x|}; \\ 1 - \mathbf{c}_{|x|} & \text{if } |x| \in \mathcal{L} \text{ and } x \in C_{|x|}. \end{cases}$$

For each layer $\ell \in \mathcal{L}$, recall that C_ℓ contains essentially half the points of that layer. Since most layers belong to $\mathcal{L}$, it is easy to see that $g_{\mathbf{c},\mathcal{L}}$ is far from symmetric.

[3]We will refer to such a tester as being in "normal form," see Definition 12 We can assume that our tester is in normal form because, as shown in Lemma 13, going from a tester $\mathcal{A}$ to one in normal form with the same performance guarantee only requires one additional bit of memory.

In keeping with the intuition sketched above, we view the function $g_{c,\mathcal{L}}$ as a "corrupted" version of f_c, where we have flipped the value of f_c on the points in C_ℓ for each light layer ℓ. Note that f_c and $g_{c,\mathcal{L}}$ agree on all points in all heavy layers, which intuitively are the ones that have a large impact on any transition.

The remaining task is to show there exists a choice of $c \in \{0,1\}^{n+1}$ such that the functions f_c and $g_{c,\mathcal{L}}$ are not distinguishable by $\mathcal{A}$. In particular, for every transition (s, s') we will have that

$$\text{either} \quad \frac{\left|\mathcal{A}_{f_c}(s,s') - \mathcal{A}_{g_{c,\mathcal{L}}}(s,s')\right|}{|\mathcal{A}_{f_c}(s,s')|} = O(1/M) \quad \text{or} \quad \mathcal{A}_{f_c}(s,s') = \mathcal{A}_{g_{c,\mathcal{L}}}(s,s') = 0.$$

As was suggested earlier, this multiplicative closeness of the transition probabilities will be enough to guarantee that for any state s, the long-run fractions of time that the Markov chains $\mathcal{A}_{f_c}$ and $\mathcal{A}_{g_{c,\mathcal{L}}}$ spend in state s are close to each other. Thus algorithm $\mathcal{A}$ cannot be in an accept state on input f_c most of the time and also be in a reject state on input $g_{c,\mathcal{L}}$ most of the time, which means it is not a valid tester for symmetry.

Without going into too much detail, the required c will be obtained roughly as follows. First, if $\ell^\star$ is heavy for some transition (s, s') then we will set $c_{\ell^\star} = b_{s'}$.[4] This will guarantee that for any transition (s, s') we have:

$$\mathcal{A}_{f_c}(s,s') \geq \sum_{\ell^\star : \ell^\star \text{ is heavy for } (s,s')} W_{s,s'}(\ell^\star)$$

In particular, by Equation (3) if ℓ is a light layer, then its weight $W_{s,s'}(\ell)$ is relatively small compared to the probability of going from s to s' in the Markov chain $\mathcal{A}_{f_c}$.

It remains to set the bits c_ℓ when ℓ is a light layer. To do this, first observe that f_c and $g_{c,\mathcal{L}}$ only disagree on points x in the light layers. Using this observation, via a discrepancy argument we show that there must exist an assignment to these remaining bits such that $|\mathcal{A}_{f_c}(s,s') - \mathcal{A}_{g_{c,\mathcal{L}}}(s,s')|$ is "relatively small" compared to $\mathcal{A}_{f_c}(s,s')$ simultaneously for all transitions s, s'. As described earlier, this enables us to apply the Markov chain tree theorem and obtain our lower bound.

The above sketch was about symmetric functions. But what did we really use about symmetric functions, and what can we abstract away from the argument to use in other lower bounds?

(1) We can view each layer ℓ of the hypercube as a "block" of inputs B_ℓ. As long as we label all points in the same block B_ℓ with the same output bit b_ℓ, the resulting function f is symmetric.

(2) For each block B_ℓ, we had a subset $C_\ell \subseteq B_\ell$ of "corruptible" inputs. By taking a symmetric function f and flipping its value on the C_ℓ subsets for a large enough fraction of blocks B_ℓ, we were able to build functions g that are far from symmetric.

The above two properties turn out to capture a general "block framework"; in Section 6.1 we show how our lower bounds for both symmetric and monotone functions can be obtained in this framework.

[4] A natural question at this point is what happens if $\ell^\star$ is heavy both for transition (s, s') and for transition (s_1, s'_1) where $b_{s'} \neq b_{s'_1}$. In the actual proof, for each transition we will define the set of heavy layers in such a way that any given ℓ can be heavy for at most one transition, and this will ensure that the above definition of $c_{\ell^\star}$ is indeed well defined. (This is the sense in which our early description of what it means for a layer $\ell^\star$ to be heavy for (s, s') was "slightly hiding some complexity.")

To build the blocks for our monotonicity testing lower bound, we will use the so-called "Talagrand Partition" of the hypercube, an idea that was used in [Kan13, BB16, CWX17, CCC+26]. Roughly speaking a Talagrand partition is defined by a collection of $2^{\sqrt{n}}$ many monotone terms $T_1, \ldots, T_{2^{\sqrt{n}}}$ each of width $\sqrt{n}$ (a useful intuition is to think of the terms $T_1, \ldots$ as being chosen independently and uniformly at random from all width-$\sqrt{n}$ monotone terms). We will have $2^{\sqrt{n}}$ blocks, where B_i is the set of points $x \in \{0,1\}^n$ that have $T_i(x) = 1$ but $T_j(x) = 0$ for all $j \neq i$. One of the key features of the "Talagrand Partition" is that if x, y are in different blocks, then x and y are incomparable with respect to the monotonicity partial ordering (i.e., $x \not\preceq y$ and $y \not\preceq x$), and hence the pair (x, y) can't form a violation to monotonicity. Hence we achieve property (1) above: as long as we label all points in the same block B_ℓ with the same output bit b_ℓ, the resulting function f will be monotone (provided that f additionally labels points that satisfy two or more distinct terms T_i, T_j as 1 and labels points satisfying no term T_i as 0). With a bit more work we can argue that the blocks B_ℓ have "corruptible" subsets C_ℓ, which gives us (2) and lets us apply our general block framework.

2.2.2 Lower Bound for Linearity, k-juntas and LTFs

We first give the high-level overview of the lower bound for linearity testing. Recall that for symmetry and monotonicity, we constructed yes-instances by independently assigning labels to disjoint blocks of the domain. This freedom is not available for linearity testing, since for any points x and y, any linear function (parity) f must satisfy $f(x) \oplus f(y) = f(x \oplus y)$. This makes the lower bound more complicated, but it still relies on some of the key ideas underlying the symmetry/monotonicity lower bounds.

Fix an M-state alleged tester $\mathcal{A}$ for linearity, where $M = 2^{\alpha n}$ for a small constant $0 < \alpha < 1$. The goal is to find a parity function, f_a, $a \in \{0,1\}^n$, together with another input function g_a such that g_a is far from every parity, and at the same time the transition probabilities between every pair of states in the induced Markov chains, $\mathcal{A}_{f_a}$ and $\mathcal{A}_{g_a}$, are nearly indistinguishable. As before, crucially we need all pairs of these transition probabilities to be *multiplicatively* close to each other, even for very unlikely transitions. As before, we will assume without loss of generality that in $\mathcal{A}$ for every state s, there is an associated bit $b_s \in \{0,1\}$ such that only points x with $f(x) = b_s$ have nonzero probability of transitioning into state s.

Step 0: Identifying the Heavy Points. We begin by identifying the query points x whose answers could individually make a large difference. For a fixed transition (s, s') we say that a query point's *weight* is the probability of querying x from s and upon seeing $f(x) = b_{s'}$ transitioning from s to s' (this was defined as $\delta_{s,s'}(x)$ in the previous section). We select a sufficiently large number of the highest-weight points for each transition, and call a point *heavy* if it is selected for at least one transition, and *light* otherwise. Note that this partition of all points into heavy and light depends only on the testing algorithm $\mathcal{A}$, and not on any particular input function.

Now we want to restrict attention to *good* parities, where a parity f_a is good if for every transition (s, s') we have that $f_a(x) = b_{s'}$ for a constant fraction of the heavy points x of the transition (s, s'). Pairwise independence of parity functions together with Chebyshev's inequality will guarantee that almost all a's are good (See Lemma 50). Goodness gives an essential starting point for multiplicative approximation: it guarantees that every transition has many heavy points contributing to it. Therefore, all remaining light points have only a tiny weight compared to the unchanged heavy contribution, and — crucially — this holds true even when the transition itself is very unlikely.

Step 1: Defining indistinguishable input pairs. Given a good a, we want to construct a function that is far from not only f_a, but that is far from all parity functions. Towards this goal we define a distribution $\mathbf{g}_a$ of functions where $\mathbf{g}_a$ agrees with f_a on all heavy points, and for each light point x, we randomly and independently select $\mathbf{g}_a(x)$ to be 0 or 1; thus $\mathbf{g}_a$ is a random variable, and it is not hard to show that almost all $g_a \sim \mathbf{g}_a$ are far from every parity function. It is left to show that a random $g_a \sim \mathbf{g}_a$ is fairly likely to be ε-indistinguishable from f_a. For this analysis we will consider an intermediate Markov chain, called the "mean-field chain," whose transition matrix is the entry-wise average

$$\mathcal{A}_{\mathrm{Avg}_a} := \mathbf{E}_{\boldsymbol{g}_a}[\mathcal{A}_{\boldsymbol{g}_a}].$$

This chain has the same state space and initial state as $\mathcal{A}$. The heavy-point contributions are the same in $\mathcal{A}_{f_a}$, $\mathcal{A}_{\boldsymbol{g}_a}$, and $\mathcal{A}_{\mathrm{Avg}_a}$; each light point contributes half its weight to the mean-field transition probability. Thus all differences between the three transition matrices come from the light points.

Now we want to prove that for most good a's, with reasonable probability a random $g_a \sim \boldsymbol{g}_a$ will have the $\mathcal{A}_{g_a}$ transition probabilities multiplicatively close to those of $\mathcal{A}_{f_a}$. We prove this in the two steps below: Step 2 shows that the transition probabilities of $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ are multiplicatively close for most good a's; then Step 3 shows that with high probability over $g_a \sim \boldsymbol{g}_a$, the transition probabilities of $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{g_a}$ are multiplicatively close.

Step 2: Showing that $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ are multiplicatively close. For a fixed transition, each light point contributes an error of plus or minus half its weight. To control the magnitude of their sum, we average its square over a uniformly random choice of a. When we expand this square, the products of errors from distinct points have expectation zero, by orthogonality of parity functions. Consequently, the expected squared total error is controlled by the sum of the squared individual weights. We call a parity function *typical* if, for every transition, its squared error is at most $100M^2$ times this expectation. Markov's inequality and a union bound show that most parity functions are typical. Since most parities are also good, we can choose one that is both good and typical, which implies that the transition probabilities of $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ have small multiplicative error for every transition. See Lemma 46 for more details.

Step 3: Showing that $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{g_a}$ are multiplicatively close. Fix a good, typical parity a, and recall that $\boldsymbol{g}_a$ has each light answer independently set to 0 or 1 with equal probability. Each light point contributes half its weight in expectation, exactly its mean-field contribution. Moreover, independence of the random variables $\boldsymbol{g}_a(x)$ for light points x again makes the products of errors from distinct points average to zero. We therefore obtain the same bound on the squared error as before, and for the same reason as before, it is small relative to the transition probability. The difference between the two arguments is the source of cancellation: orthogonality of parities in the first, and independence of the random light answers in the second. Chebyshev's inequality and a union bound give a constant probability that all transitions are close simultaneously. We can therefore choose one g_a that is both far from all parities and multiplicatively close to f_a on every transition. See Lemma 47 for more details.

Our lower bounds for k-juntas will be very similar to the one for parities, using the simple observation that parities restricted to k variables are k-juntas. Similarly our lower bound for LTFs will follow from our lower bound for parities of 1 variable (i.e., literals).

3 Preliminaries

We write $[n]$ to denote the set $\{1, \ldots, n\}$. 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 a string $x \in \{0,1\}^n$, we use $|x|$ to denote the Hamming weight of x. We will use bold font to denote random variables, for instance $\boldsymbol{x}$ or $\mathbf{X}$. Given $x, y \in \{0,1\}^n$ we use $x \cdot y$ to denote the dot product of two vectors over $\mathbb{F}_2$.

A property $\mathcal{P}$ is any non-empty subset of functions mapping $\{0,1\}^n$ to $\{0,1\}$. The distance between two functions f, g is

$$d(f,g) = \frac{\left|\{\, x \in \{0,1\}^n : f(x) \neq g(x) \,\}\right|}{2^n}.$$

The distance between f and a property $\mathcal{P}$ is:

$$d(f,\mathcal{P}) := \min_{g \in \mathcal{P}} d(f,g).$$

If $d(f,\mathcal{P}) \geq \varepsilon$ then we say that f is ε-far from $\mathcal{P}$.

We also recall the standard definition of a property testing algorithm:

Definition 1. Fix a property $\mathcal{P}$. A (standard model) tester for $\mathcal{P}$ is a randomized algorithm $\mathcal{A}$ which takes as input parameters $\varepsilon, \delta > 0$, and black-box access to a function $f : \{0,1\}^n \to \{0,1\}$. The algorithm should accept f with probability at least $1-\delta$ when $f \in \mathcal{P}$, and reject with probability at least $1-\delta$ when f is ε-far from $\mathcal{P}$. The tester has query complexity $q : \mathbb{N} \times (0,1] \times (0,1] \to \mathbb{N}$ if on input ε, δ and query access to any function $f : \{0,1\}^n \to \{0,1\}$, the tester makes at most $q(n, \varepsilon, \delta)$ queries.

Background on Fourier analysis. Given a function $f\colon \{0,1\}^n \to \mathbb{R}$, the Fourier coefficients of f are defined as $\widehat{f}(S) := \mathbf{E}_{\boldsymbol{x} \sim \{0,1\}^n}[f(\boldsymbol{x})\chi_S(\boldsymbol{x})]$ where $\chi_S(x) = (-1)^{\sum_{i \in S} x_i}$.

Parseval's identity tells us that

$$\mathop{\mathbf{E}}_{\boldsymbol{x} \sim \{0,1\}^n}[f(\boldsymbol{x})^2] = \sum_{S \subseteq [n]} \widehat{f}(S)^2.$$

We also define

$$\|f\|_2 := \left(\sum_x f(x)^2\right)^{1/2}.$$

3.1 The properties we study

For completeness we give formal definitions of the properties we study in this work:

Definition 2 (Monotonicity)**.** A function $f\colon \{0,1\}^n \to \{0,1\}$ is *monotone* if and only if $f(x) \leq f(y)$ for all $x \preceq y$. Here $x \preceq y$ means $x_i \leq y_i$ for all $i \in [n]$.

Definition 3 (Symmetry)**.** A function $f\colon \{0,1\}^n \to \{0,1\}$ is *symmetric* if and only if $f(x) = f(y)$ whenever $|x| = |y|$.

Definition 4 (Linearity)**.** A function $f\colon \{0,1\}^n \to \{0,1\}$ is a *linear function* (also known as a *parity function*) if and only if there exists $a \in \{0,1\}^n$ such that $f(x) = a \cdot x$. Equivalently, a function f is a parity function if and only if $f(x) \oplus f(y) = f(x \oplus y)$ for all strings $x, y \in \{0,1\}^n$.

Definition 5 (k-Junta)**.** Let $k \in [n]$. A function $f : \{0,1\}^n \to \{0,1\}$ is a *k-junta* if and only if there exists a set $S \subseteq [n]$, $|S| \leq k$ such that $f(x)$ only depends on x_S. I.e., we have $f(x) = f(y)$ whenever $x_S = y_S$.

Definition 6 (Linear Threshold Function)**.** A function $f\colon \{0,1\}^n \to \{0,1\}$ is a *linear threshold function* (LTF) if and only if there exist real numbers $w_0, w_1, \ldots, w_n$ such that

$$f(x) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \ldots + w_n x_n \geq 0 \\ 0 & \text{otherwise} \end{cases}.$$

4 The Bounded-Memory Model

4.1 Memory-bounded testers

Definition 7. An *M-state tester* $\mathcal{A}$ consists of a set $[M]$ of M states (where we always view state 1 as the initial state) plus the following information for each state $s \in [M]$:

1. $\mathcal{Q}_s$, a distribution over $\{0,1\}^n$, which describes which input string to query when the algorithm is in state s.

2. $\mathcal{M}_s$, a function from $\{0,1\}^n \times \{0,1\} \times [M]$ to $[0,1]$. For a given current state s, the distribution $\mathcal{M}_s$ describes the probability of transitioning to state s' for each choice of $(x, f(x))$. In particular if $\mathcal{A}$ is run on a particular Boolean function f, then given that $\mathcal{A}$ is in state s and x is queried, $\mathcal{M}_s(x, f(x), s')$ is the probability that the algorithm enters state s'.

3. A value $v(s) \in \{\text{Acc}, \text{Rej}\}$.

On input f the testing algorithm updates its state as follows: When it is in state s, the algorithm chooses a point $\boldsymbol{x} \sim \mathcal{Q}_s$ to query, receives $f(\boldsymbol{x})$ as the response to its query, and then transitions to the next state according to the distribution $\mathcal{M}_s(\boldsymbol{x}, f(\boldsymbol{x}), \cdot)$. Thus on input $f : \{0,1\}^n \to \{0,1\}$, the algorithm $\mathcal{A}$ executes on f according to the evolution of a Markov chain that we denote by $\mathcal{A}_f$. This chain has transition probability from state s to state s' given by

$$\mathcal{A}_f(s, s') = \sum_{x \in \{0,1\}^n} \delta_{s,s'}(x, f(x)), \text{ where} \tag{4}$$

$$\delta_{s,s'}(x, b) = \mathcal{Q}_s(x) \cdot \mathcal{M}_s(x, b, s'), \qquad x \in \{0,1\}^n, b \in \{0,1\} \tag{5}$$

Lastly, we need to describe the conditions for $\mathcal{A}$ to be a successful tester for a Boolean function property $\mathcal{P} \subseteq \{0,1\}^{2^n}$. As stated above, on input $f\colon \{0,1\}^n \to \{0,1\}$, $\mathcal{A}_f$ is a Markov chain. For $s \in [M]$, let

$$\widehat{p}_{\mathcal{A}_f}(s) := \lim_{T \to \infty} \sum_{t=0}^{T-1} \frac{1}{T} \mathbf{Pr}[\mathcal{A}_f \text{ is in state } s \text{ at time } t].$$

Let $S_{\mathrm{Acc}} \subseteq [M]$ be the set of states with value Acc in $\mathcal{A}$, and let $\widehat{p}_{\mathcal{A}_f}(S_{\mathrm{Acc}})$ denote the probability (in the limit) of being in an Acc-state, i.e.,

$$\widehat{p}_{\mathcal{A}_f}(S_{\mathrm{Acc}}) := \sum_{s \in [M] : v(s) = \mathrm{Acc}} \widehat{p}_{\mathcal{A}_f}(s).$$

For arbitrary $\varepsilon, \delta \in (0, 1]$, we say that $\mathcal{A}$ is an (ε, δ)-*tester for* $\mathcal{P}$ if for every $f \in \mathcal{P}$ we have $\widehat{p}_{\mathcal{A}_f}(S_{\mathrm{Acc}}) \geq 1 - \delta$, and for every g that is ε-far from every $f \in \mathcal{P}$ we have $\widehat{p}_{\mathcal{A}_g}(S_{\mathrm{Acc}}) \leq \delta$.

For simplicity, throughout this paper we will focus on the setting where ε, δ are small fixed constants. We will accordingly often just refer to a $(1/100, 1/3)$-tester for $\mathcal{P}$ as a "tester for $\mathcal{P}$".

We sometimes refer to an M-state testing algorithm as an algorithm that uses m *bits of memory*, where $m = \lceil \log_2 M \rceil$.

4.2 Markov chain results for lower bounds.

All proofs for results of this section are in Appendix A.

From indistinguishability to close long-term probabilities. We show that approximating the long-term probabilities $\widehat{p}_{\mathcal{A}_f}(s)$ reduces to approximating the individual transition probabilities $\mathcal{A}_f(s, s')$. In particular this will allow us to show that if a tester $\mathcal{A}$ is such that all transition probabilities for $\mathcal{A}_f$ and $\mathcal{A}_g$ are extremely close to one another (multiplicatively), then the acceptance probabilities for f and g are very close to each other. This will be useful in proving lower bounds.

Definition 8. Let $\mathcal{C}_1, \mathcal{C}_2$ be M-state Markov chains. We say that $\mathcal{C}_1$ and $\mathcal{C}_2$ are *ε-indistinguishable* if the two chains have the same set of states and the same start state, and if, for all transitions (s, s'), we have

$$(1 + \varepsilon)^{-1} \cdot \mathcal{C}_1(s, s') \leq \mathcal{C}_2(s, s') \leq (1 + \varepsilon) \cdot \mathcal{C}_1(s, s').$$

Note that the definition of ε-indistinguishability is symmetric.

Note that ε-indistinguishability respects some form of composition:

Observation 9. Let $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$ be M-state Markov chains and $\varepsilon \leq 1$. If $\mathcal{C}_1, \mathcal{C}_2$ are ε-indistinguishable and $\mathcal{C}_2, \mathcal{C}_3$ are ε-indistinguishable then $\mathcal{C}_1, \mathcal{C}_3$ are 3ε-indistinguishable.

The following lemma states that if M-state Markov chains $\mathcal{C}_1$ and $\mathcal{C}_2$ are ε-indistinguishable for sufficiently small ε, then $\mathcal{C}_1$ and $\mathcal{C}_2$ have similar long-term behavior (and hence similar behavior with respect to testing).

Lemma 10. Let $\mathcal{C}_1, \mathcal{C}_2$ be two M-state Markov chains, with the same start state, such that they are $\frac{1}{10M}$-indistinguishable. Then for every state s, it holds that $\widehat{p}_{\mathcal{C}_1}(s) \leq 1.5\widehat{p}_{\mathcal{C}_2}(s)$.

From the above lemma, we immediately have the following, which we will use in our lower bounds:

Corollary 11. Let $\mathcal{A}$ be an M-state tester and $\mathcal{P}$ a property. Suppose that there exist functions f, g such that (1) $f \in \mathcal{P}$; (2) g is ε-far from every function in $\mathcal{P}$; and (3) $\mathcal{A}_f$ and $\mathcal{A}_g$ are $1/(10M)$-indistinguishable. Then $\mathcal{A}$ is not an $(\varepsilon, 1/3)$-tester for $\mathcal{P}$.

Normal Form for Transitions. To simplify the lower bounds it will be helpful to assume without loss of generality that $\mathcal{A}$ has the following normal form with respect to its transitions.

Definition 12 (Normal Form)**.** A tester $\mathcal{A}$ is in normal form if for every state s', there exists $b_{s'} \in \{0,1\}$ such that for all s, x: $\delta_{s,s'}(x, 1-b_{s'}) = 0$. That is, for each state s', all transitions into s' that have non-zero probability have the same value $b_{s'}$ for $f(x)$.

For $\mathcal{A}$ in normal form, since $b_{s'}$ is fixed for every state s', we can denote the transition probability $\delta_{s,s'}(x, b_{s'})$ by $\delta_{s,s'}(x)$.

We can turn $\mathcal{A}$ into a normal form algorithm by enforcing that $\mathcal{A}$ store in memory the bit b which is equal to the last value of $f(x)$ that it received. In particular, normal form can be achieved using just one more bit of memory:

Lemma 13. Let $\mathcal{A}$ be an (ε, δ)-tester for a property $\mathcal{P}$ using M states. Then there exists an (ε, δ)-tester $\mathcal{A}'$ in normal form using $2M$ states.

Since being in normal form only increases the number of bits of memory needed by 1, in our lower bounds we will always assume that our testers are in normal form.

5 Upper Bounds

5.1 Space upper bound for monotonicity

This section is dedicated to proving the following theorem:

Theorem 14. There is a $2^{\tilde{O}(\sqrt{n}/\varepsilon^2)}$-state tester for monotonicity.

As detailed in the introduction, we will describe our algorithm as a clock-equipped one, which makes a finite number $T = 2^{\tilde{O}(\sqrt{n}/\varepsilon^2)}$ of queries and uses $S = 2^{\tilde{O}(\sqrt{n}/\varepsilon^2)}$ states. By the generic conversion mentioned in the Introduction (using an additional $\log T$ bits of memory to simulate a clock that goes up to T), this will imply an $O(ST)$-state tester for monotonicity, establishing the above theorem. Thus, we will focus on proving:

Theorem 15. There is a $2^{\tilde{O}(\sqrt{n}/\varepsilon^2)}$-state, $2^{\tilde{O}(\sqrt{n}/\varepsilon^2)}$-query clock-equipped tester for monotonicity.

In what follows, we set $L \coloneqq O\big(\sqrt{n}\log(n/\varepsilon)/\varepsilon^2\big)$, and

$$\rho \coloneqq 1 - \frac{\varepsilon^2}{128\sqrt{n}} \tag{6}$$

such that $\rho \leq 1 - \frac{3\log(n/\varepsilon)}{L}$.

The algorithm We can now describe the algorithm behind Theorem 15. It relies on two successive tests, involving the *noise sensitivity* of the function f (Definition 21), and its *low-degree noisy version* $f_\rho^{\leq L}$ (Eq. (8)).

Algorithm 1 The clock-equipped algorithm for monotonicity testing

1: In the first $T_1 = \mathrm{poly}(2^{L\log n}, 1/\varepsilon)$ time steps, using $S_1 = 2^{O(L\log(n/\varepsilon))}$ states, estimate $\mathrm{NS}_{\frac{1-\rho}{2}}[f]$ to $\pm\frac{\varepsilon^2}{512}$ with probability of success 5/6, as described in Lemma 27. If the estimate exceeds $\frac{3\varepsilon^2}{512}$, enter a reject state.

2: In the next $T_2 = \mathrm{poly}(2^{L\log n}, 1/\varepsilon)$ time steps, using $S_2 = 2^{O(L\log(n/\varepsilon))}$ states, distinguish between

$$\mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}\Big[\, D_{\boldsymbol{i}} f_{\widetilde{\rho}}^{\leq L}(\boldsymbol{x}) \leq -1/(4n) \,\Big] \leq \varepsilon/(4n^3) \quad \text{and} \quad \mathbf{Pr}_{\boldsymbol{x},\boldsymbol{i}}\Big[\, D_{\boldsymbol{i}} f_{\widetilde{\rho}}^{\leq L}(\boldsymbol{x}) \leq -1/(2n) \,\Big] \geq \varepsilon/(2n^3)$$

using a hard-coded set of edges (x, i), as described in Lemma 28. In the latter case, enter a reject state.

3: Enter an accept state otherwise.

To establish the theorem, we need to prove two things: first, that the above algorithm is correct, i.e., will accept (with high probability) monotone functions, and reject (w.h.p.) far-from-monotone ones. Second, we need to show that both subtests can be implemented by a small-space clock-equipped tester. We start with the first aspect, correctness, through a series of structural lemmas (Lemma 22 for the first subtest, and Lemma 24 and Corollary 26 for the second); and defer the small-space implementation to the end of the section (Section 5.1.1).

Some more Fourier background. To simplify the Fourier-analysis-based arguments in this section, we will switch to the ± 1 representation of the domain and codomain of our functions: that is, we take our Boolean functions f to be mappings from $\{\pm1\}^n$ to $\{\pm1\}$. Our argument will rely on some standard concepts and facts from Boolean Fourier analysis, which we briefly recall below. The interested reader is referred to [O'D14, Chapter 2].

Definition 16 (Influence of Boolean functions)**.** The *influence* of the i-th coordinate of a Boolean function $f\colon \{\pm1\}^n \to \{\pm1\}$ is defined as $\mathbf{Inf}_i(f) \coloneqq \mathbf{Pr}_{\boldsymbol{x}\sim\{\pm1\}^n}\big[\, f(\boldsymbol{x}) \neq f(\boldsymbol{x}^{\oplus i}) \,\big]$. The *total influence* is then defined as $\mathbf{Inf}[f] \coloneqq \sum_{i=1}^n \mathbf{Inf}_i(f)$.

This can be generalized to real-valued functions $f\colon \{\pm1\}^n \to \mathbb{R}$ as follows. Letting, for $i \in [n]$, D_i denote the i-th discrete derivative operator,

$$D_i f(x) \coloneqq \frac{f(x^{i\mapsto 1}) - f(x^{i\mapsto -1})}{2}, \qquad x \in \{\pm1\}^n,$$

where $x^{i\mapsto b} \in \{\pm1\}^n$ is the string obtained from x by fixing x_i to b, we have:

Definition 17 (Influence of real-valued functions)**.** The *influence* of the i-th coordinate of a real-valued function $f\colon \{\pm1\}^n \to \mathbb{R}$ is defined as $\mathbf{Inf}_i(f) \coloneqq \mathbb{E}_{\boldsymbol{x}\sim\{\pm1\}^n}[D_i f(\boldsymbol{x})^2]$. The *total influence* is again defined as $\mathbf{Inf}[f] \coloneqq \sum_{i=1}^n \mathbf{Inf}_i(f)$.

The next two results are standard:

Fact 18 ([O'D14, Theorem 2.20])**.** Fix any $f\colon \{\pm1\}^n \to \mathbb{R}$. Then, for $i \in [n]$,

$$\mathbf{Inf}_i(f) = \sum_{\substack{S\subseteq[n]\\ S\ni i}} \widehat{f}(S)^2,$$

which implies $\mathbf{Inf}[f] = \sum_{S\subseteq[n]} |S|\widehat{f}(S)^2$.

Fact 19 ([O'D14, Theorem 2.33 and Proposition 3.2])**.** Fix any monotone $f\colon \{\pm1\}^n \to \{\pm1\}$. Then $\mathbf{Inf}[f] \le \sqrt{n}$, and, for every $\alpha \in (0,1]$,

$$\sum_{S:|S|\ge\sqrt{n}/\alpha} \widehat{f}(S)^2 \le \alpha\,,$$

that is, f has Fourier spectrum α-*concentrated* on levels up to $\sqrt{n}/\alpha$.

As discussed in Section 2, an intuitive interpretation of Fact 19 is that "monotone functions are essentially $2^{O(\sqrt{n})}$-dimensional objects, not 2^n."

Definition 20 (Noise operator)**.** The *noise operator* with parameter $\rho \in [-1,1]$ is the linear operator T_ρ defined, for any $f\colon \{\pm1\}^n \to \mathbb{R}$, by

$$\mathrm{T}_\rho f(x) = \mathbb{E}_{\boldsymbol{y}\sim N_\rho(x)}[f(\boldsymbol{y})], \qquad x \in \{\pm1\}^n,$$

where $\boldsymbol{y} \sim N_\rho(x)$ is obtained by flipping each coordinate of x independently with probability $\frac{1-\rho}{2}$.

Definition 21 (Noise sensitivity)**.** The *noise sensitivity* of a Boolean function $f\colon \{\pm1\}^n \to \{\pm1\}$ with parameter $\delta \in [0,1]$, denoted $\mathrm{NS}_\delta[f]$ is the probability that $f(\boldsymbol{x}) \ne f(\boldsymbol{y})$, when $\boldsymbol{x} \in \{\pm1\}^n$ is drawn uniformly at random, and $\boldsymbol{y} \sim N_{1-2\delta}(\boldsymbol{x})$, is obtained by flipping each coordinate of $\boldsymbol{x}$ independently at random with probability δ.

Lemma 22 (Monotone functions have low sensitivity)**.** Let $\rho = 1 - \Theta\Big(\frac{\log(n/\varepsilon)}{L}\Big)$ as in (6). Then, for any monotone $f\colon \{\pm1\}^n \to \{\pm1\}$, we have $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{\varepsilon^2}{256}$.

Proof. This follows from the fact that $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{1-\rho}{2}\mathbf{Inf}[f]$ for every $f\colon \{\pm1\}^n \to \{\pm1\}$ [O'D14, Exercise 2.42], combined with Fact 19. □

The noisy version of f. We first show that the noisy version $\mathrm{T}_\rho f$ of the input function has the desired properties: it preserves both monotonicity and violations of monotonicity, and has very good Fourier concentration on degree up to L.

Lemma 23. Let $\rho = 1 - \Theta\Big(\frac{\log(n/\varepsilon)}{L}\Big)$ as in (6). We have the following: for any $f\colon \{\pm1\}^n \to \{\pm1\}$, letting $f_\rho \coloneqq \mathrm{T}_\rho f\colon \{\pm1\}^n \to [-1,1]$,

1. If f is monotone, then so is f_ρ, and
$$\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\big[\,D_{\boldsymbol{i}} f_\rho(\boldsymbol{x}) < 0\,\big] = 0\,.$$
2. If f is ε-far from monotone with noise sensitivity $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{\varepsilon^2}{128}$, then
$$\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[\,D_{\boldsymbol{i}} f_\rho(\boldsymbol{x}) \le -\frac{3}{4n}\,\right] \ge \frac{\varepsilon}{n^3}\,.$$
3. f_ρ has Fourier spectrum α-concentrated on levels up to L, for $\alpha = O(\varepsilon^6/n^6)$.

Proof. The first bullet is straightforward, and actually holds for any value of $\rho \in [0,1]$: for any $x \preceq x'$, there is a natural coupling of $y \sim N_\rho(x)$ and $y' \sim N_\rho(x')$ such that $y \preceq y'$ and thus $f(y) \le f(y')$ (always), from which $\mathrm{T}_\rho f(x) \le \mathrm{T}_\rho f(x')$.

Turning to the second bullet, observe that, by Parseval and the Fourier expression for $\mathrm{T}_\rho f$,

$$\mathbb{E}\Big[(f(\boldsymbol{x}) - f_\rho(\boldsymbol{x}))^2\Big] = \sum_{S\subseteq[n]} (1-\rho^{|S|})^2 \widehat{f}(S)^2 \le \sum_{S\subseteq[n]} (1-\rho^{|S|}) \widehat{f}(S)^2 = 2\,\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{\varepsilon^2}{64}. \tag{7}$$

This implies

$$\mathbb{E}\big[|f(\boldsymbol{x}) - f_\rho(\boldsymbol{x})|\big] \le \sqrt{\mathbb{E}\big[(f(\boldsymbol{x}) - f_\rho(\boldsymbol{x}))^2\big]} \le \frac{\varepsilon}{8}\,.$$

Now, since f is ε-far from monotone, its violation graph[5] has a matching of size at least $\varepsilon 2^n$ [FLN+02, Lemma 4]: letting M be such a matching, we have that (1) $f(x) = 1$ and $f(y) = -1$ for every $(x,y) \in M$, and (2) each element $z \in \{\pm1\}^n$ appears at most once as either endpoint of an edge in M, from which

$$\frac{\varepsilon}{8}2^n \ge \sum_{z\in\{\pm1\}^n} |f(z) - f_\rho(z)| \ge \sum_{(x,y)\in M} (|f(x) - f_\rho(x)| + |f(y) - f_\rho(y)|) = \sum_{(x,y)\in M} \underbrace{(1 - f_\rho(x) + f_\rho(y) + 1)}_{\ge 0}.$$

Dividing by $|M| \ge \varepsilon 2^n$, and applying Markov's inequality we then have that there is a matching $M_\rho \subseteq M$ of size at least $\frac{1}{2}|M|$ such that, for all $(x,y) \in M_\rho$,

$$1 - f_\rho(x) + f_\rho(y) + 1 \le \frac{1}{4}\,,$$

or, equivalently, $f_\rho(x) - f_\rho(y) \ge \frac{7}{4}$. Recalling that $f_\rho(x), f_\rho(y) \in [-1,1]$, this implies that $f_\rho(x) \ge 3/4$ and $f_\rho(y) \le -3/4$ for every $(x,y) \in M_\rho$. So f_ρ, like f, has a large matching of "blatant violations of monotonicity." From this matching, we want to extract a large number of "noticeably violating edges" (i.e., edges witnessing a violation of monotonicity by at least an $\Omega(1/n)$ gap between the values at the two endpoints). With this goal in mind, since there are $\binom{n+1}{2} < n^2$ pairs of Hamming levels (denoted $L_0, \dots, L_n$) in the Boolean hypercube, we observe that there must exist some $0 \le i < j \le n$ with

$$|M_\rho \cap (L_i \times L_j)| \ge \frac{|M_\rho|}{n^2} \ge \frac{\varepsilon}{2n^2} \cdot 2^n.$$

That is, there are many pairs from M_ρ with one endpoint with Hamming weight i and the other j. We can now invoke the Lehman–Ron Theorem [LR01, Theorem 1] to obtain, from these many disjoint pairs starting in L_i and ending in L_j, as many *vertex-disjoint* (and in particular edge-disjoint) paths starting from one of these vertices in L_i and ending at one of the endpoints in L_j. Since each of these paths starts at a vertex x with $f_\rho(x) \ge 3/4$ and ends at a vertex y with $f_\rho(y) \le -3/4$ and has length $j - i \le n$, it must have at least one non-monotone edge with a gap of $3/(2n)$. Considering one such edge from each path, we get a collection E of at least $\frac{\varepsilon}{2n^2} \cdot 2^n$ distinct edges of the form (x,y), with $x \preceq y$ but $f_\rho(y) - f_\rho(x) \le -\frac{3}{2n}$. This establishes the second bullet, as

$$\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n \\ \boldsymbol{i}\sim[n]}}\left[\, D_{\boldsymbol{i}} f_\rho(\boldsymbol{x}) \le -\frac{3}{4n} \,\right] \ge \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n \\ \boldsymbol{i}\sim[n]}}\left[\, (\boldsymbol{x}^{\boldsymbol{i}\mapsto -1}, \boldsymbol{x}^{\boldsymbol{i}\mapsto 1}) \in E \,\right] = \frac{1}{n} \cdot \frac{2|E|}{2^n} \ge \frac{\varepsilon}{n^3}.$$

[5] Recall that the violation graph of a Boolean function f (with respect to monotonicity) is the graph $G_f = (V,E)$ where $V = \{\pm1\}^n$, and $(x,y) \in E$ whenever $x \preceq y$ but $f(x) > f(y)$.

Turning finally to the third bullet, note that by the Fourier expression for the noise operator, we have that $f_\rho(x) = \sum_{S\subseteq[n]} \rho^{|S|}\widehat{f}(S)\chi_S(x)$ and so

$$\sum_{S:|S|>L} \widehat{f_\rho}(S)^2 = \sum_{S:|S|>L} \rho^{2|S|}\widehat{f}(S)^2 \le \rho^{2L} \sum_{S:|S|>L} \widehat{f}(S)^2 \le \rho^{2L},$$

and the conclusion follows from the setting of $\rho = 1 - \Theta\Big(\frac{\log(n/\varepsilon)}{L}\Big)$. □

The truncated noisy version of f. Now that we have obtained, from $f\colon \{\pm1\}^n \to \{\pm1\}$, a noisy version $f_\rho\colon \{\pm1\}^n \to [-1,1]$ which preserves some key properties (as far as testing monotonicity is concerned) *and whose Fourier spectrum is tightly concentrated on levels at most L*, we can go one step further and only focus on these L layers, considering the low-degree truncation $f_\rho^{\le L}$ of f_ρ, defined by

$$f_\rho^{\le L}(x) := \sum_{S:|S|\le L} \widehat{f_\rho}(S)\chi_S(x) = \sum_{S:|S|\le L} \rho^{|S|}\widehat{f}(S)\chi_S(x), \qquad x \in \{\pm1\}^n. \tag{8}$$

While this truncation may no longer perfectly preserve monotonicity of the original function f, we show that it is "good enough", as it provides a quantitative analogue of Lemma 23:

Lemma 24. Let ρ, L be as before. For any $f\colon \{\pm1\}^n \to \{\pm1\}$, the function $f_\rho^{\le L}\colon \{\pm1\}^n \to \mathbb{R}$ satisfies the following.

1. If f is monotone, then
$$\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n \\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}} f_\rho^{\le L}(\boldsymbol{x}) \le -\frac{1}{4n} \right] \le \beta\,,$$
for some $\beta = O\big(\varepsilon^6/n^4\big)$.

2. If f is ε-far from monotone with noise sensitivity $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{\varepsilon^2}{128}$, then
$$\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n \\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}} f_\rho^{\le L}(\boldsymbol{x}) \le -\frac{1}{2n} \right] \ge \frac{\varepsilon}{2n^3}\,.$$

Proof. Consider the truncated part, $f_\rho^{>L} := f_\rho - f_\rho^{\le L}$. We then have

$$\mathbb{E}_{\boldsymbol{i},\boldsymbol{x}}[D_{\boldsymbol{i}} f_\rho^{>L}(\boldsymbol{x})^2] = \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{\boldsymbol{x}}[D_i f_\rho^{>L}(\boldsymbol{x})^2] = \frac{1}{n}\sum_{S\subseteq[n]} |S|\widehat{f_\rho^{>L}}(S)^2 \le \sum_{S\subseteq[n]} \widehat{f_\rho^{>L}}(S)^2 = \sum_{S:|S|>L} \widehat{f_\rho}(S)^2.$$

the second equality follows by Fact 18. This last quantity is at most $\alpha = O(\varepsilon^6/n^6)$ by Lemma 23 (Item 3). Now:

- If f is monotone, then by linearity of D_i, Lemma 23 (Item 1), and Markov's inequality,

$$\begin{aligned}
\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho^{\le L}(\boldsymbol{x}) \le -\frac{1}{4n}\right] &= \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho(\boldsymbol{x}) - D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x}) \le -\frac{1}{4n}\right]\\
&\le \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x}) \ge \frac{1}{4n}\right] && (D_if_\rho \ge 0)\\
&\le \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x})^2 \ge \frac{1}{16n^2}\right]\\
&\le 16n^2\alpha = O\left(\frac{\varepsilon^6}{n^4}\right), && (\text{Markov's inequality})
\end{aligned}$$

  showing the first claim.

- If f is ε-far from monotone with small noise sensitivity, then again by linearity of D_i,

$$\begin{aligned}
\mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho^{\le L}(\boldsymbol{x}) \le -\frac{1}{2n}\right] &= \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho(\boldsymbol{x}) - D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x}) \le -\frac{1}{2n}\right]\\
&\ge \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho(\boldsymbol{x}) \le -\frac{3}{4n} \text{ and } D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x}) \ge -\frac{1}{4n}\right]\\
&\ge \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho(\boldsymbol{x}) \le -\frac{3}{4n}\right] - \mathbf{Pr}_{\substack{\boldsymbol{x}\sim\{\pm1\}^n\\ \boldsymbol{i}\sim[n]}}\left[D_{\boldsymbol{i}}f_\rho^{>L}(\boldsymbol{x}) < -\frac{1}{4n}\right]\\
&\ge \frac{\varepsilon}{n^3} - 16n^2\alpha && (\text{Lemma 23 (Item 2) and Markov})\\
&= \frac{\varepsilon}{n^3} - O\left(\frac{\varepsilon^6}{n^4}\right),
\end{aligned}$$

  which is at least $\varepsilon/(2n^3)$ for n large enough. □

Estimating $\mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}f_\rho^{\le L}(\boldsymbol{x}) \le -\frac{3}{8n}\right]$. Having shown in Lemma 24 that it suffices to estimate this quantity, we now show how to deterministically carry out this estimation using a small, fixed (in the sense of having no dependence on f) set of edges, leveraging the fact that the truncated function $f_\rho^{\le L}$ has low degree.

Lemma 25. For $1 \le d \le n$, let $\mathcal{F}_{\le d} \subseteq \{\{\pm1\}^n \to \mathbb{R}\}$ be the set of all functions with Fourier degree at most d, and let $\eta \in (0,1]$. Then, there exists a (multi)set $H_{d,\eta} \subseteq \{(x,i) : i \in [n], x \in \{\pm1\}^n\}$ of edges such that the following holds:

(i) $|H_{d,\eta}| = \frac{n^{O(d)}}{\eta^2}$;

(ii) For every $h \in \mathcal{F}_{\le d}$, and every $t \in \mathbb{R}$,

$$\left|\frac{1}{|H_{d,\eta}|}\sum_{(x,i)\in H_{d,\eta}} \mathbf{1}[D_ih(x) \le t] - \mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}h(\boldsymbol{x}) \le t\right]\right| \le \eta\,.$$

That is, $H_{d,\eta}$ is a small approximating set for the probabilities $\mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}h(\boldsymbol{x}) \le t\right]$, uniformly good over all low-degree functions h and thresholds t.

Before proving this lemma, we show how this implies the desired result for our algorithm:

Corollary 26. There exists a (multi)set $H \subseteq \{(x,i) : i \in [n], x \in \{\pm 1\}^n\}$ of $\mathrm{poly}(n^L, 1/\varepsilon)$ edges such that, for every $f\colon \{\pm 1\}^n \to \{\pm 1\}$ and every $t \in \mathbb{R}$,

$$\left| \frac{1}{|H|} \sum_{(x,i)\in H} \mathbf{1}[D_i f_{\rho}^{\leq L}(x) \leq t] - \mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}} f_{\rho}^{\leq L}(\boldsymbol{x}) \leq t \right] \right| \leq \frac{\varepsilon}{16n^3}\,.$$

Proof. Setting $d = L$ and $\eta = \varepsilon/(16n^3)$, we apply Lemma 25 to the subclass of functions $\{f_{\rho}^{\leq L} : f\colon \{\pm 1\}^n \to \{\pm 1\}\}$, using the fact that, by definition, for any function f the low-degree truncation $f_{\rho}^{\leq L}$ has Fourier degree at most L. □

Proof of Lemma 25. Fix $1 \leq i \leq n$. For $S \subseteq [n]$ with $|S| \leq d$, we define the function $\Phi_S\colon \{\pm 1\}^n \times [n] \to \mathbb{R}$ by

$$\Phi_S(x,i) = \frac{\chi_S(x^{i\mapsto 1}) - \chi_S(x^{i\mapsto -1})}{2}, \qquad (x,i) \in \{\pm 1\}^n \times [n],$$

which is equal to 0 if $i \notin S$, and $\chi_{S\setminus\{i\}}(x)$ otherwise. For $h \in \mathcal{F}_{\leq d}$, we can then rewrite

$$D_i h(x) = \sum_{S\subseteq[n]:S\ni i} \widehat{h}(S)\chi_{S\setminus\{i\}}(x) = \sum_{S:|S|\leq d, S\ni i} \widehat{h}(S)\chi_{S\setminus\{i\}}(x) = \sum_{S:|S|\leq d} \widehat{h}(S)\Phi_S(x,i) = \sum_{S:|S|\leq d} \widehat{h}(S) y_S\,,$$

where $y = (\Phi_S(x,i))_{S\subseteq[n]:|S|\leq d} \in \mathbb{R}^D$, for $D \coloneqq \sum_{k=0}^{d} \binom{n}{k} = n^{O(d)}$. The uniform distribution on edges (x,i) induces a distribution μ on $y \in \mathbb{R}^D$, and, for every $t \in \mathbb{R}$, $D_i h(x) \leq t$ if, and only if, the LTF $h_t(y) \coloneqq \mathbf{1}[\sum_{S:|S|\leq d} \widehat{h}(S) y_S \leq t]$ in $\mathbb{R}^D$ evaluates to 1. Since the VC dimension of LTFs over $\mathbb{R}^D$ is $D+1$, the VC uniform convergence theorem (see e.g. Theorem 6.8 of [SSBD14]) guarantees that a (multi)set of

$$s = O\left(\frac{D}{\eta^2}\right) = \frac{n^{O(d)}}{\eta^2}$$

i.i.d. samples from μ suffices, with probability at least, say, 9/10, to approximate $\mathbb{E}_{y\sim\mu}[h_t(y)]$ by the empirical average to error η, simultaneously for every $h \in \mathcal{F}_{\leq d}$ and every $t \in \mathbb{R}$. Since this probability is strictly positive, in particular there *exists* a (multi)set of s samples with the same guarantees. For each sample y in this multiset, choose an edge (x,i) mapping to y; and let the resulting multiset of edges be the claimed $H_{d,\eta}$. □

Correctness of the algorithm We are now able to establish the first part of Theorem 15: namely, that the algorithm provided is correct with high probability.

Completeness. Suppose f is monotone. By Lemma 22, it has noise sensitivity (for our ρ) $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \leq \frac{\varepsilon^2}{256}$, and with probability at least 5/6 the estimate from the first subtest is at most

$$\frac{\varepsilon^2}{256} + \frac{\varepsilon^2}{512} = \frac{3\varepsilon^2}{512}$$

and hence the algorithm does not enter a reject state in Step 1. The second subtest, using the hard-coded multiset H of edges guaranteed to exist by Corollary 26, provides, as described in Lemma 28, an estimate $\widehat{\gamma}$ such that, with probability at least 5/6,

$$\widehat{\gamma} \leq \mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}} f_{\rho}^{\leq L}(\boldsymbol{x}) \leq -\frac{1}{4n} \right] + \frac{\varepsilon}{16n^3} \leq O\left(\frac{\varepsilon^6}{n^4}\right) + \frac{\varepsilon}{16n^3} < \frac{3\varepsilon}{8n^3}\,,$$

(here the second inequality is from Lemma 24 (Item 1)). Hence the algorithm does not enter a reject state here either: it then enters an accept state in the last step.

Soundness. Suppose f is ε-far from monotone. If $\mathrm{NS}_{\frac{1-\rho}{2}}[f] > \frac{\varepsilon^2}{128}$, then, with probability at least 5/6 the estimate from the first subtest exceeds

$$\frac{\varepsilon^2}{128} - \frac{\varepsilon^2}{512} = \frac{3\varepsilon^2}{512}$$

so the algorithm enters a reject state in Step 1, and we are done. Otherwise, if $\mathrm{NS}_{\frac{1-\rho}{2}}[f] \le \frac{\varepsilon^2}{128}$, then the second subtest provides an estimate $\widehat{\gamma}$ such that, with probability at least 5/6,

$$\widehat{\gamma} \ge \mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}} f_{\rho}^{\le L}(\boldsymbol{x}) \le -\frac{1}{2n} \right] - \frac{\varepsilon}{16n^3} \ge \frac{\varepsilon}{2n^3} - \frac{\varepsilon}{16n^3} > \frac{3\varepsilon}{8n^3},$$

(here the second inequality is from Lemma 24 (Item 2)). So the algorithm enters a reject state.

Overall, by a union bound the algorithm enters the correct state with probability at least 2/3, as claimed.

5.1.1 Implementing both subtests in small space.

It only remains to show that both subtests of the algorithm can indeed be implemented by a clock-equipped algorithm in small space. We do this in Lemmas 27 and 28, respectively.

Lemma 27 (Estimating the noise sensitivity)**.** There is a $2^{O(L\log(n/\varepsilon))}$-state clock-equipped algorithm which constructs an estimate $\widehat{\gamma}$, such that, after $2^{O(L\log n)}/\varepsilon^4$ queries,

$$|\widehat{\gamma} - \mathrm{NS}_{\frac{1-\rho}{2}}[f]| \le \frac{\varepsilon^2}{512}$$

with probability at least 5/6.

Proof. For $f\colon \{\pm1\}^n \to \{\pm1\}$, we can rewrite $\mathrm{NS}_{\frac{1-\rho}{2}}[f]$ (recall (7)) as

$$\mathrm{NS}_{\frac{1-\rho}{2}}[f] = \frac{1}{2}\sum_{S\subseteq[n]} (1-\rho^{|S|})\widehat{f}(S)^2 = \frac{1}{2}\left(1 - \sum_{S\subseteq[n]} \rho^{|S|}\widehat{f}(S)^2\right)$$

(see, e.g., [O'D14, Theorem 2.49]), and so it suffices to estimate $\sum_{S\subseteq[n]} \rho^{|S|}\widehat{f}(S)^2$ to $\pm\frac{\varepsilon^2}{256}$. Moreover, since (as in the proof of Lemma 23 (Item 3)) we have

$$\sum_{S:|S|>L} \rho^{|S|}\widehat{f}(S)^2 \le \rho^L \le \frac{\varepsilon^3}{n^3} < \frac{\varepsilon^2}{512}$$

(for large enough n), it is enough to estimate $\sum_{S:|S|\le L} \rho^{|S|}\widehat{f}(S)^2$ to $\pm\frac{\varepsilon^2}{512}$. To do so, we use the following 2-query process, which maintains a side counter $\widehat{C}$:

1. Draw a set $\boldsymbol{S} \subset [n]$ of size at most L, where S is drawn with probability proportional to $\rho^{|S|}$ (note that this distribution can be hard-coded in the clock-equipped algorithm, as it does not depend on f), and store the resulting set $\boldsymbol{S}$.
2. Draw $\boldsymbol{x} \in \{\pm 1\}^n$ uniformly at random, query $f(\boldsymbol{x})$, compute and store the bit $f(\boldsymbol{x})\chi_{\boldsymbol{S}}(\boldsymbol{x})$.
3. Draw $\boldsymbol{y} \in \{\pm 1\}^n$ uniformly at random, query $f(\boldsymbol{y})$, compute the bit $f(\boldsymbol{y})\chi_{\boldsymbol{S}}(\boldsymbol{y})$, increment $\widehat{C}$ with the value $f(\boldsymbol{x})\chi_{\boldsymbol{S}}(\boldsymbol{x})f(\boldsymbol{y})\chi_{\boldsymbol{S}}(\boldsymbol{y})$, and "forget" everything else.

Besides the counter $\widehat{C}$, this 3-step process only requires $O(L \log n)$ bits of space (to store the set $\boldsymbol{S}$ and the bit $f(\boldsymbol{x})\chi_{\boldsymbol{S}}(\boldsymbol{x})$). Letting $W \coloneqq \sum_{S:|S|\le L} \rho^{|S|}$, it is easy to check that the value $f(\boldsymbol{x})\chi_{\boldsymbol{S}}(\boldsymbol{x})f(\boldsymbol{y})\chi_{\boldsymbol{S}}(\boldsymbol{y}) \in \{\pm 1\}$ added to the counter has expectation

$$\mathbb{E}_{\boldsymbol{S},\boldsymbol{x},\boldsymbol{y}}[f(\boldsymbol{x})\chi_{\boldsymbol{S}}(\boldsymbol{x})f(\boldsymbol{y})\chi_{\boldsymbol{S}}(\boldsymbol{y})] = \mathbb{E}_{\boldsymbol{S}}[\widehat{f}(\boldsymbol{S})^2] = \frac{1}{W}\sum_{S:|S|\le L} \rho^{|S|}\widehat{f}(S)^2.$$

Hence by a standard Hoeffding bound, after $T = O(W^2/\varepsilon^4)$ repetitions of this 3-step process, the quantity $\frac{W}{T}\widehat{C}$ will be within $\pm\frac{\varepsilon^2}{512}$ of $\sum_{S:|S|\le L} \rho^{|S|}\widehat{f}(S)^2$ with probability at least $5/6$, in which case $\widehat{\gamma} \coloneqq \frac{1}{2}(1 - \frac{W}{T}\widehat{C})$ is, by the foregoing discussion, a good estimate of $\mathrm{NS}_{\frac{1-\rho}{2}}[f]$. Since $|\widehat{C}| \le T$ and $W \le n^{O(L)}$, the total space required is

$$O(\log T) + O(L \log n) = O(L \log n + \log(1/\varepsilon))$$

which gives the claimed bound on the number of states. As for the number of queries, the above bound on W gives $2T \le 2^{O(L \log n)}/\varepsilon^4$. □

Lemma 28 (Estimating $\mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}f_{\widetilde{\rho}}^{\le L}(\boldsymbol{x}) \le t\right]$)**.** There is a $2^{O(L\log(n/\varepsilon))}$-state clock-equipped algorithm which constructs an estimate $\widehat{\gamma}$, such that, after $\mathrm{poly}(n^L, 1/\varepsilon)$ queries,

$$\mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}f_{\widetilde{\rho}}^{\le L}(\boldsymbol{x}) \le -\frac{1}{2n}\right] - \frac{\varepsilon}{16n^3} \le \widehat{\gamma} \le \mathbf{Pr}_{\boldsymbol{i},\boldsymbol{x}}\left[D_{\boldsymbol{i}}f_{\widetilde{\rho}}^{\le L}(\boldsymbol{x}) \le -\frac{1}{4n}\right] + \frac{\varepsilon}{16n^3}$$

with probability at least $5/6$.

Proof. Fix the multiset H of edges guaranteed to exist by Corollary 26: setting $h \coloneqq f_{\widetilde{\rho}}^{\le L}$ for convenience, it is enough, in view of Corollary 26, to compute an estimate $\widehat{\gamma}$ such that, with probability at least $5/6$,

$$\frac{1}{|H|}\sum_{(x,i)\in H} \mathbf{1}[D_i h(x) \le -\tfrac{1}{2n}] \le \widehat{\gamma} \le \frac{1}{|H|}\sum_{(x,i)\in H} \mathbf{1}[D_i h(x) \le -\tfrac{1}{4n}].$$

In turn, by a union bound, it is enough to show how to obtain, for a fixed (x,i), a value $\boldsymbol{v}_{x,i}$ such that

$$\mathbf{Pr}\left[\mathbf{1}[D_i h(x) \le -\tfrac{1}{2n}] \le \boldsymbol{v}_{x,i} \le \mathbf{1}[D_i h(x) \le -\tfrac{1}{4n}]\right] \ge 1 - \frac{1}{6|H|}$$

which is what we will focus on now. Fix any $(x,i) \in H$: unrolling the definition of $D_i h(x)$, we have

$$\begin{aligned} D_i h(x) &= \sum_{S : S \ni i} \widehat{h}(S) \chi_{S \setminus \{i\}}(x) = \sum_{S : |S| \le L, S \ni i} \rho^{|S|} \widehat{f}(S) \chi_{S \setminus \{i\}}(x) \\ &= \sum_{S : |S| \le L, S \ni i} \rho^{|S|} \left(\frac{1}{2^n} \sum_{y \in \{\pm 1\}^n} f(y) \chi_S(y) \right) \chi_{S \setminus \{i\}}(x) \\ &= \sum_{y \in \{\pm 1\}^n} a_y f(y) \,, \end{aligned}$$

where

$$a_y \coloneqq \frac{1}{2^n} \sum_{S : |S| \le L, S \ni i} \rho^{|S|} \chi_S(y) \chi_{S \setminus \{i\}}(x)$$

does not depend on f. Letting $A \coloneqq \sum_{y \in \{\pm 1\}^n} |a_y|$, consider the following 1-query process, which maintains a side counter $\widehat{C}$:

- Draw an element $\boldsymbol{y} \in \{\pm 1\}^n$, where y is drawn with probability proportional to $|a_y|$ (this distribution can be hard-coded in the clock-equipped algorithm, as it does not depend on f).
- Query $f(\boldsymbol{y})$, and compute the bit

$$b_{\boldsymbol{y}} \coloneqq \begin{cases} f(\boldsymbol{y}) & \text{if } a_{\boldsymbol{y}} \ge 0 \\ -f(\boldsymbol{y}) & \text{otherwise.} \end{cases}$$

- Increment $\widehat{C}$ with the value $b_{\boldsymbol{y}}$.

It is not hard to check that the bit $b_{\boldsymbol{y}} \in \{\pm 1\}$ satisfies $\mathbb{E}_{\boldsymbol{y}}[b_{\boldsymbol{y}}] = \frac{1}{A} \sum_{y \in \{\pm 1\}^n} f(y) a_y$. Moreover, since, by the triangle inequality,

$$A \le \frac{1}{2^n} \sum_{y \in \{\pm 1\}^n} \sum_{S : |S| \le L, S \ni i} \rho^{|S|} |\chi_S(y) \chi_{S \setminus \{i\}}(x)| = \sum_{S : |S| \le L, S \ni i} \rho^{|S|} \le n^{O(L)} \,,$$

a standard Hoeffding bound shows that $R \coloneqq O\big(A^2 n^2 \log |H|\big) = n^{O(L)} \log(1/\varepsilon)$ repetitions are enough to obtain an estimate of $\sum_{y \in \{\pm 1\}^n} a_y f(y)$ (i.e., $D_i h(x)$) accurate to $\pm 1/(8n)$, with probability $1 - 1/(6|H|)$. For this, the space required is

$$O(L \log n + \log \log(1/\varepsilon))$$

to maintain the counter $\widehat{C}$. Once this estimate has been obtained, the bit $\boldsymbol{v}_{x,i} \coloneqq \mathbf{1}\left[\frac{A}{R} \widehat{C} \le -3/(8n)\right]$ is computed, and the value of the counter $\widehat{C}$ can be discarded.

After going through all edges $(x,i) \in H$, the algorithm computes $\widehat{\gamma} \coloneqq \frac{1}{|H|} \sum_{(x,i) \in H} \boldsymbol{v}_{x,i}$. Overall, the total space necessary is thus $O(L \log n + \log \log(1/\varepsilon)) + O(\log |H|) = O(L \log(n/\varepsilon))$ (where the first term is for the computation of the current $\boldsymbol{v}_{x,i}$, and the second is for the running total $\sum_{(x,i) \in H} \boldsymbol{v}_{x,i}$). □

Having established both correctness and the space and query usage of our clock-equipped tester, this concludes the proof of Theorem 15. □

5.2 Space upper bound for symmetry

The rest of our space upper bounds are much simpler than the monotonicity upper bound. In order to reduce the number of parameters and keep the exposition simple, we present all of our remaining testing upper bounds with distance parameter ε fixed to be 0.01 and confidence parameter δ fixed to be a small constant such as 1/3.

The upper bound follows from the known symmetry tester discussed in [DSS11, BWY15].

Theorem 29. There is a tester for symmetry that uses $O(\log n)$ bits of memory and makes $O(1)$ queries.

Proof. Recall that the algorithm of [DSS11, BWY15] works by repeating the following routine $O(1)$ times: First draw a uniformly random point $\boldsymbol{x} \in \{0,1\}^n$ and query $f(\boldsymbol{x})$; then draw a uniformly random point $\boldsymbol{y}$ conditioned on $|\boldsymbol{y}| = |\boldsymbol{x}|$ and query $f(\boldsymbol{y})$, rejecting if $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$. It is easy to see that in order to implement each two-query iteration of the routine we only need to store $f(\boldsymbol{x})$ and the Hamming weight $|\boldsymbol{x}|$ of $\boldsymbol{x}$, which takes only $O(\log n)$ bits; moreover, the counter for how many repetitions have been performed uses only $O(1)$ additional bits. The overall algorithm accepts if all $O(1)$ many repetitions of this two-query routine pass. □

5.3 Space upper bound for linearity

This upper bound follows from the standard BLR linearity tester [BLR93].

Theorem 30. There is a tester for linearity that uses $O(n)$ bits of memory and makes $O(1)$ queries.

Proof. Recall that the algorithm of [BLR93] repeats the following test independently $O(1)$ times: sample independent uniformly random strings $\boldsymbol{x}, \boldsymbol{y} \in \{0,1\}^n$ and check that $f(\boldsymbol{x}) \oplus f(\boldsymbol{y}) = f(\boldsymbol{x} \oplus \boldsymbol{y})$. If all the tests pass, the algorithm accepts.

This test can easily be implemented using $O(n)$ bits of memory. We query $\boldsymbol{x}$ and store the $(n+1)$-bit string $(\boldsymbol{x}, f(\boldsymbol{x}))$. We then query $\boldsymbol{y}$ and replace the stored string by $(\boldsymbol{x} \oplus \boldsymbol{y}, f(\boldsymbol{x}) \oplus f(\boldsymbol{y}))$. Finally, we query $\boldsymbol{x} \oplus \boldsymbol{y}$ and compare the response $f(\boldsymbol{x} \oplus \boldsymbol{y})$ with the stored bit $f(\boldsymbol{x}) \oplus f(\boldsymbol{y})$, moving to a reject state if they differ (and staying there). A tester using a clock repeats the above routine, which makes three queries, $O(1)$ times. Such a clock can be implemented using $O(1)$ bits of memory. □

5.4 Space upper bound for k-juntas

We first recall the following result of [Bla16] (Proposition 2.3):

Lemma 31. Let $f : \{0,1\}^n \to \{0,1\}$ be ε-far from being a k-junta, and let $S \subseteq [n]$ satisfy $|S| \leq k$. If $\boldsymbol{x}$ is uniformly distributed over $\{0,1\}^n$ and the string $\boldsymbol{y}$ is uniformly distributed in $\{y \in \{0,1\}^n : y_S = \boldsymbol{x}_S\}$, then

$$\mathbf{Pr}[f(\boldsymbol{x}) \neq f(\boldsymbol{y})] \geq \varepsilon.$$

The idea is to build an algorithm that iterates over all k-element subsets $S_1, \ldots, S_\ell$ of $[n]$, one at a time. For each S_i, the algorithm looks for points $x, y \in \{0,1\}^n$ with $x_{S_i} = y_{S_i}$ but $f(x) \neq f(y)$. Such a pair certifies that f does not depend only on the variables in S_i, so once such a pair is found the algorithm moves to the next candidate subset. If no such pair is found after sufficiently many trials, the algorithm accepts.

Theorem 32. Let $k \geq 1$. There exists a tester for the property of being a k-junta that uses $O(k \log n)$ bits of memory and makes $n^{O(k)}$ queries.

Proof. Let $\ell = \binom{n}{k}$, and let $S_1, \ldots, S_\ell$ be a fixed enumeration of all subsets of $[n]$ of size k (say in lexicographic order). Set

$$L := \lceil 100 \ln(3\ell) \rceil.$$

The tester stores an index $i \in \{1, \ldots, \ell\}$ and a trial counter $t \in \{0, \ldots, L\}$. Starting with $i = 1$ and $t = 0$, repeat the following two-query trial until the tester accepts or rejects:

- Draw $\boldsymbol{x}$ uniformly from $\{0,1\}^n$, query $f(\boldsymbol{x})$, and store $(\boldsymbol{x}_{S_i}, f(\boldsymbol{x}))$. Then draw $\boldsymbol{y}$ uniformly from $\{y \in \{0,1\}^n : y_{S_i} = \boldsymbol{x}_{S_i}\}$ and query $f(\boldsymbol{y})$.
- If $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$, set $i \leftarrow i + 1$ and $t \leftarrow 0$. If $i = \ell + 1$, go to a reject state.
- If $f(\boldsymbol{x}) = f(\boldsymbol{y})$, set $t \leftarrow t + 1$. If $t = L$, go to an accept state.

If f is a k-junta, there is a set S_j such that f only depends on the variables in S_j. The algorithm either accepts earlier or reaches S_j, where every trial gives equal answers and it accepts after L trials. Now suppose that f is 0.01-far from being a k-junta. For each set S_i, Lemma 31 gives

$$\mathbf{Pr}[f(\boldsymbol{x}) \neq f(\boldsymbol{y})] \geq 0.01.$$

Conditional on reaching S_i, the probability of passing all the L trials is at most $(0.99)^L$. A union bound over all ℓ candidate sets gives

$$\mathbf{Pr}[\text{the tester accepts}] \leq \ell(0.99)^L \leq \ell e^{-0.01L} \leq \frac{1}{3}.$$

The index i and the counter t require $O(\log(\ell+1))$ bits, while $(\boldsymbol{x}_{S_i}, f(\boldsymbol{x}))$ requires $k+1$ bits. The two-query phase uses $O(1)$ additional bits. Thus the total memory is $O(k + \log(\ell+1)) = O(k \log n)$ bits. There are at most ℓL trials, hence at most $2\ell L = O(\ell \log(\ell+1))$ queries. □

5.5 Space upper bound for LTFs

Theorem 33. There is a tester for the property of being an LTF that uses $O(\log(n))$ bits of memory and makes $\mathrm{poly}(n)$ queries.

Since going through all details would be onerous, we only provide a proof sketch. Our algorithm is a modification of the algorithm of [MORS10] so we begin by recalling the main elements of that algorithm. Throughout this section, we use the $\{\pm 1\}$ representation of both the domain and codomain, so $f : \{\pm 1\}^n \to \{\pm 1\}$.

Background: The [MORS10] approach. We write $\tau = \tau(\varepsilon) > 0$ for the "regularity" parameter used in [MORS10], recalling that this parameter depends only on ε and not on n. (Here a function $f : \{\pm 1\}^n \to \{\pm 1\}$ is said to be τ-regular if $|\widehat{f}(i)| \leq \tau$ for all $i \in [n]$.) Let $\gamma = \tau^2$. The core of the [MORS10] tester revolves around a $\mathrm{poly}(1/\varepsilon)$-size set $J \subset [n]$, which is meant to capture the set $\{i : |\widehat{f}(i)| \geq \gamma\}$ of "influential" coordinates whose degree-1 Fourier coefficients are at least γ in magnitude. For each restriction $\pi \in \{\pm 1\}^J$ the algorithm uses estimates

$$\text{of } \mu_\pi := \mathbf{E}[f_\pi], \qquad \text{of } \sigma_\pi^2 := \sum_{k \notin J} \widehat{f_\pi}(k)^2, \qquad \text{of } \kappa_\pi := \sum_{k \notin J} \widehat{f_\pi}(k)^4,$$

and, for pairs of restrictions,

$$\text{of } \rho_{\pi,\pi'} := \sum_{k \notin J} \widehat{f_\pi}(k)\widehat{f_{\pi'}}(k).$$

Here f_π denotes the subfunction obtained by fixing the coordinates in J according to π.

The [MORS10] algorithm constructs a set J, and given J it performs the following checks:

- If almost all restrictions are "nearly constant", then enumerate all $2^{O(|J|^2)}$ possible LTFs on the variables in J. Otherwise,
- Choose a restriction π^* whose mean is bounded away from ± 1 (such a restriction must exist because many restrictions are not "nearly constant"), and
  - Check regularity by checking that κ_{π^*} is "small";
  - Check that $\sigma^2_{\pi^*} \approx W(\mu_{\pi^*})$ (where W is the [MORS10] Gaussian-isoperimetric function, defined by $W(v) = \left(2\varphi(\mu^{-1}(v))\right)^2$ where φ is the standard Gaussian density and $\mu(\theta)$ is defined as $\mathbf{E}_{\boldsymbol{G}\sim N(0,1)}[\mathrm{sign}(\boldsymbol{G} - \theta)]$);
  - Check "cross-consistency", i.e., that $\rho^2_{\pi^*,\pi} \approx W(\mu_{\pi^*})W(\mu_\pi)$ and $\widetilde{\rho}_{\pi^*,\pi} \geq -\eta$ for every tested restriction π, where $\widetilde{\rho}_{\pi^*,\pi}$ estimates $\rho_{\pi^*,\pi}$ to the prescribed additive accuracy η;
  - And finally enumerate a finite grid of weight vectors on the variables in J and thresholds, of size depending only on ε, to check that the restricted thresholds are globally consistent.

Sections 6.3-6.6 of the [MORS10] analysis prove completeness and soundness of this approach provided that (i) the set J contains every coordinate with $|\widehat{f}(i)| \geq \gamma$ and contains only coordinates with $|\widehat{f}(i)| \geq \gamma/2$, and (ii) the above parameters are estimated to certain required (τ-dependent) accuracies. In our setting where ε is a constant, the required accuracy levels are constant.

Our small-space variant of [MORS10]: Working "one coordinate at a time." The [MORS10] algorithm estimates sums of powers of degree-1 Fourier coefficients by querying correlated tuples of strings. This is query-efficient, but as mentioned earlier a direct implementation would use at least n bits of memory to store a query string in order to generate its correlated partners. Our approach is instead to estimate the same sums of powers of degree-1 Fourier coefficients by scanning all n coordinates in sequence.

The basic primitive is the following. Suppose a constant-size set $R \subset [n]$ and a restriction $\pi \in \{\pm 1\}^R$ have been stored (note that for constant-size R, this takes $O(\log n)$ bits of memory). For a coordinate $k \notin R$, we have that

$$\widehat{f_\pi}(k) = \mathbf{E}_{\boldsymbol{z}}[\boldsymbol{z}_k f_\pi(\boldsymbol{z})],$$

where $\boldsymbol{z}$ is uniform over the unrestricted coordinates. This coefficient can be estimated by making m independent queries to uniformly random strings $\boldsymbol{z}$ extending π and keeping the running sum of the ± 1 random variables $\boldsymbol{z}_k f_\pi(\boldsymbol{z})$ in a counter. With $m = O(\lambda^{-2}\log(n/\delta_0))$, Hoeffding's inequality gives additive error λ with failure probability $\delta_0/n^{O(1)}$. The counter has range $[-m, m]$, hence uses $O_\varepsilon(\log m) = O_\varepsilon(\log n)$ bits when $\lambda = n^{-O_\varepsilon(1)}$.

The key point is that no query string is ever stored in this approach. With a clock-equipped tester, the current block of time specifies which coordinate k and which sample number are being used. The transition rule of the clock-equipped tester "sees" the queried string $\boldsymbol{z}$ and the oracle

answer, so it can add $\boldsymbol{z}_k f(\boldsymbol{z})$ to the counter and then forget $\boldsymbol{z}$. The same approach is used to estimate $\widehat{f}(i)$ in the initial scan which is performed to obtain the set J, and to estimate all restricted coefficients $\widehat{f}_\pi(k)$ later.

The space-efficient clock-equipped testing algorithm. In more detail, our algorithm proceeds as follows. Set $\lambda \ll \tau/n$ with $\lambda \le \gamma/4$.

1. *Find the set J of influential coordinates.* For $i = 1, \dots, n$, estimate $\widehat{f}(i) = \mathbf{E}[\boldsymbol{z}_i f(\boldsymbol{z})]$ as described above. Store every index whose estimate has magnitude at least $3\gamma/4$. If more than $4/\gamma^2$ such indices are found, reject. When all estimates are accurate, which happens with high probability, the choice $\lambda \le \gamma/4$ ensures that

$$\{i : |\widehat{f}(i)| \ge \gamma\} \subseteq J \subseteq \{i : |\widehat{f}(i)| \ge \gamma/2\}.$$

   Thus $|J| \le 4/\gamma^2$ by Parseval, so J has constant size for constant ε.

2. *Estimate the [MORS10] statistics.* For each of the $O_\varepsilon(1)$ restrictions $\pi \in \{\pm 1\}^J$ needed by the [MORS10] decision stage, estimate μ_π by sampling. (Since $|J|$ is constant, the number of restrictions and pairs is constant.) Given fixed restrictions π and π', let $\tilde{a}_k$ and $\tilde{b}_k$ denote the empirical estimates of $\widehat{f}_\pi(k)$ and $\widehat{f}_{\pi'}(k)$ obtained in a space-efficient way as described above. Scan over all $k \notin J$ and accumulate rounded values of $\widetilde{a}_k^2$, $\widetilde{a}_k^4$, or $\widetilde{a}_k \widetilde{b}_k$ to obtain additive $O(\tau)$-accurate estimates of σ_π^2, κ_π, and $\rho_{\pi,\pi'}$, respectively, by the union bound and the inequalities $|a_k| \le 1$, $\sum_k a_k^2 \le 1$. The accumulated totals each have $O_\varepsilon(\log n)$ bits of precision, so each total again takes $O_\varepsilon(\log n)$ bits of memory.

3. *Run the deterministic [MORS10] checks.* Use the estimated values in the known-J [MORS10] decision procedure described above, including the sign check $\widetilde{\rho}_{\pi^*,\pi} \ge -\eta$ for every tested restriction π. All enumerations are over a number of objects depending only on ε which is constant for constant ε.

Time and memory. For constant ε there are constantly many scans, each scan has n coordinates, and each coordinate uses $m = \mathrm{poly}(n)$ samples. Thus the query complexity of the algorithm is $T = \mathrm{poly}(n)$. At every moment the algorithm stores only: the constant-size list J of indices, a constant number of counters and rounded estimates with polynomial range, and finitely many control bits. This is $O(\log n)$ bits, or $S = \mathrm{poly}(n)$ states. In the clock-equipped implementation, the clock supplies the current phase, coordinate, restriction, and sample number. Simulating this clock in our model requires another $O(\log T) = O(\log n)$ bits of memory, preserving the claimed $O(\log n)$ total space bound.

Correctness. With probability at least $1 - o(1)$, all coordinate estimates and all derived [MORS10] statistics are within the prescribed constant accuracies; this is just a Chernoff bound and a union bound over polynomially many estimates. Conditioned on this event, the tester has supplied the same information used in the [MORS10] Test-LTF decision stage. The [MORS10] completeness argument shows that every LTF is accepted with constant probability, and the [MORS10] soundness argument implies that any function accepted with probability larger than the constant threshold is ε-close to an LTF. This concludes the proof sketch of Theorem 33.

6 Lower Bounds

Throughout all our lower bounds, we will assume the algorithms $\mathcal{A}$ are in normal form. This is without loss of generality since, as mentioned in Lemma 13, we can turn any algorithm into one in normal form (with the same correctness guarantees) using only one extra bit of memory.

6.1 A framework for symmetry and monotonicity lower bounds

As mentioned in the introduction, we will prove our lower bounds for symmetric and monotone functions using a general framework.

Definition 34 (Block system)**.** Let I be a nonempty finite index set and let $0 < \beta \leq 1$. An (I, β)-*block system* consists of the following:

(a) A set of pairwise disjoint blocks $\{B_i \mid i \in I\}$ with $B_i \subseteq \{0,1\}^n$, such that

$$\sum_{i \in I} |B_i| \geq \beta \cdot 2^n, \qquad |B_i| \leq \frac{2^n}{|I|} \quad \forall i \in I.$$

(b) For each $i \in I$, a sub-block $C_i \subseteq B_i$.

(c) A "background map" $b : \{0,1\}^n \setminus \bigcup_{i \in I} B_i \to \{0,1\}$.

We also need to make sure that a block system is "corruptible," meaning that it can give us both functions in $\mathcal{P}$ and also functions far from $\mathcal{P}$.

Definition 35. Fix a property $\mathcal{P} \subseteq \{0,1\}^{2^n}$. An (I, β)-block system *corrupts property* $\mathcal{P}$ if it respects the following two conditions:

(1.) **Any extension of the background map that labels each block with a fixed bit is consistent with $\mathcal{P}$:** For every $c \in \{0,1\}^I$, the function f_c belongs to $\mathcal{P}$, where $f_c : \{0,1\}^n \to \{0,1\}$ is defined as follows:

$$f_c(x) = \begin{cases} c_i & \text{if there is some } i \in I \text{ with } x \in B_i, \\ b(x) & \text{otherwise.} \end{cases}$$

(2.) **Blocks are corruptible:** For any nonempty $J \subseteq I$ and any $c \in \{0,1\}^I$, consider the function $g_{c,J}$ obtained from f_c by flipping its value on $\bigcup_{i \in J} C_i$. That is,

$$g_{c,J}(x) = \begin{cases} 1 - f_c(x) & \text{if } x \in \bigcup_{i \in J} C_i, \\ f_c(x) & \text{otherwise.} \end{cases}$$

Then

$$d(g_{c,J}, \mathcal{P}) \geq 0.2 \sum_{i \in J} \frac{|B_i|}{2^n}.$$

We remark that there is nothing special about the constant 0.2, it is simply chosen for concreteness; any other positive absolute constant would work just as well, with minor corresponding changes to the statements and proofs below. The idea of condition (2.) is that when the value of f_c is flipped on a collection of sub-blocks C_i, this incurs distance from $\mathcal{P}$ that is linear in the total size of the associated blocks B_i.

The following theorem will allow us to prove a space lower bound for any tester for a property $\mathcal{P}$ whenever there is a block system that corrupts $\mathcal{P}$.

Theorem 36. Let $\mathcal{P}$ be a property of Boolean functions such that there exists an (I, β)-block system that corrupts $\mathcal{P}$. If $\mathcal{A}$ is an M-state tester in normal form, and

$$M^3\sqrt{\log(M)} < \frac{\beta}{400}\sqrt{|I|},$$

then $\mathcal{A}$ is not a $(\beta/10, 1/3)$-tester for $\mathcal{P}$.

Before diving into the proof, we give an overview. Given a proposed M-state tester $\mathcal{A}$ in normal form, our plan is to use Lemma 10 to show that $\mathcal{A}$ cannot successfully test $\mathcal{P}$ if M is too small. We will find $c \in \{0,1\}^I$ and $\mathcal{L} \subseteq I$ such that the Markov chains $\mathcal{A}_{f_c}$ and $\mathcal{A}_{g_{c,\mathcal{L}}}$ are $\frac{1}{10M}$-indistinguishable, while $g_{c,\mathcal{L}}$ is far from $\mathcal{P}$. To achieve this, we will separate blocks into two groups: "heavy blocks," which are the blocks that play an important role for some transition, and "light blocks", which do not. We will fix c_i to be a specific bit when B_i is heavy. For all transitions (s, s'), this will guarantee that the transition probability from s to s' is much larger than the weight of any light block. We will set $\mathcal{L}$ to be the set of indices of all the light blocks, and we will set the bits c_i for these blocks randomly, to argue that some good choice of c exists.

Proof of Theorem 36. Fix a property $\mathcal{P}$ and an (I, β)-block system that corrupts $\mathcal{P}$. Let $\mathcal{A}$ be an M-state tester in normal form, and assume that $M^3\sqrt{\log(M)} < \frac{\beta}{400}\sqrt{|I|}$. Let

$$\Lambda = 4\sqrt{|I|\log(M)}, \qquad \rho = \lceil 25M\Lambda \rceil .$$

For each transition (s, s') and each block B_i, define its *weight* by

$$W_{s,s'}(i) := \sum_{x \in B_i} \delta_{s,s'}(x, b_{s'}).$$

Process the transitions (s, s') in a fixed order, say lexicographic. For each transition, let $\mathcal{H}_{s,s'} \subseteq I$ contain the ρ indices i with the largest values of $W_{s,s'}(i)$ *which have not been previously included in an earlier* $\mathcal{H}_{s_1,s_1'}$, breaking ties arbitrarily. Thus the sets $\mathcal{H}_{s,s'}$ are pairwise disjoint and each has size ρ.[6] We define

$$\mathcal{H} := \bigcup_{s,s'} \mathcal{H}_{s,s'}$$

to be the size-$M^2\rho$ set of all *heavy block indices*. Let

$$\mathcal{L} := I \setminus \mathcal{H}$$

[6] This is possible because our assumption on M gives $M^2\rho \leq |I|$.

be the set of remaining *light block indices*. Our assumption on M gives

$$|\mathcal{H}| = \left|\bigcup_{s,s'} \mathcal{H}_{s,s'}\right| = M^2\rho \le 200M^3\sqrt{\log(M)|I|} \le \frac{\beta|I|}{2} \le \frac{|I|}{2}. \tag{9}$$

The following claim is the key step of the proof:

Claim 37. There exists a string $c \in \{0,1\}^I$ such that the Markov chains $\mathcal{A}_{f_c}$ and $\mathcal{A}_{g_{c,\mathcal{L}}}$ are $\frac{1}{10M}$-indistinguishable.

Before proving the claim, we show how it lets us complete the proof of Theorem 36. Consider the string c which is guaranteed to exist by Claim 37. By item (1.) of Definition 35, we have $f_c \in \mathcal{P}$. Furthermore, by Definition 34 and (9), we have that

$$\sum_{i\in\mathcal{L}} |B_i| \ge \beta\cdot 2^n - \frac{M^2\rho\cdot 2^n}{|I|} \ge 2^n(\beta - \beta/2) \ge 2^n\beta/2.$$

Combining this with item (2.) of Definition 35, we have $d(g_{c,\mathcal{L}}, \mathcal{P}) \ge \beta/10$, and the theorem follows by applying Corollary 11. □

We now turn to the proof of Claim 37.

Proof of Claim 37. First, recall that $\mathcal{A}$ is in normal form: for every state s' there is a bit $b_{s'}$ such that $\delta_{s,s'}(x, 1-b_{s'}) = 0$ for every state s and every string x.

We will use the probabilistic method to show that the desired string c exists. We construct a random string $\mathbf{c} \in \{0,1\}^I$ as follows. For each light block index $i \in \mathcal{L}$, choose a sign $\boldsymbol{z}_i \in \{-1,1\}$ independently and uniformly, and set

$$\mathbf{c}_i = (1+\boldsymbol{z}_i)/2.$$

For each transition (s,s') and each $i \in \mathcal{H}_{s,s'}$, set $\mathbf{c}_i = b_{s'}$. (Recalling that the $\mathcal{H}_{s,s'}$ sets are pairwise disjoint, the string $\mathbf{c}$ is well-defined.) This choice of the coordinates $\mathbf{c}_i$ for $i \in \mathcal{H}$ guarantees that, for every transition (s,s'), we have

$$\mathcal{A}_{f_{\mathbf{c}}}(s,s') \ge \sum_{i\in\mathcal{H}_{s,s'}} W_{s,s'}(i). \tag{10}$$

Now fix a transition (s,s'). For $i \in \mathcal{L}$, let

$$\Delta_{s,s'}(i) := \sum_{x\in C_i} \delta_{s,s'}(x, b_{s'}).$$

Note that $\Delta_{s,s'}(i)$ is the contribution to the transition probability when points in C_i are labeled $b_{s'}$. Furthermore, when points in C_i are labeled $1-b_{s'}$, they contribute 0 to the transition probability. Because $f_{\mathbf{c}}$ and $g_{\mathbf{c},\mathcal{L}}$ differ exactly on $\bigcup_{i\in\mathcal{L}} C_i$, and because $\mathbf{c}_i = (1+\boldsymbol{z}_i)/2$, we have

$$\left|\mathcal{A}_{f_{\mathbf{c}}}(s,s') - \mathcal{A}_{g_{\mathbf{c},\mathcal{L}}}(s,s')\right| = \left|\sum_{\substack{i\in\mathcal{L},\\ \mathbf{c}_i = b_{s'}}} \Delta_{s,s'}(i) - \sum_{\substack{i\in\mathcal{L},\\ \mathbf{c}_i \ne b_{s'}}} \Delta_{s,s'}(i)\right| = \left|\sum_{i\in\mathcal{L}} \boldsymbol{z}_i\Delta_{s,s'}(i)\right|. \tag{11}$$

Let $\gamma_{s,s'} := \max_{i\in\mathcal{L}} W_{s,s'}(i)$. We know that for any $i \in \mathcal{H}_{s,s'}$ we must have $W_{s,s'}(i) \geq \gamma_{s,s'}$, because when $\mathcal{H}_{s,s'}$ was created, we picked the ρ indices maximizing $W_{s,s'}(i)$ that hadn't already been picked for some earlier $\mathcal{H}_{s_1,s_1'}$. Since each $|\mathcal{H}_{s,s'}| = \rho$, Equation (10) gives

$$\mathcal{A}_{f_{\mathbf{c}}}(s,s') \geq \rho \cdot \gamma_{s,s'}. \tag{12}$$

Finally, since $C_i \subseteq B_i$, recalling the definition of $W_{s,s'}(i)$ we see that for any $i \in \mathcal{L}$, we have

$$|\Delta_{s,s'}(i)| = \sum_{x\in C_i} \delta_{s,s'}(x, b_{s'}) \leq \sum_{x\in B_i} \delta_{s,s'}(x, b_{s'}) = W_{s,s'}(i) \leq \gamma_{s,s'}.$$

Now we are ready to show that a good choice of c exists. As above, continue to consider a fixed transition (s,s'). If $\sum_{i\in\mathcal{L}} \Delta_{s,s'}(i)^2 = 0$ (in particular, if $\gamma_{s,s'} = 0$), then $\mathcal{A}_{f_{\mathbf{c}}}(s,s') = \mathcal{A}_{g_{\mathbf{c},\mathcal{L}}}(s,s')$ for every choice of $\mathbf{c}$. Otherwise, Hoeffding's inequality gives

$$\begin{aligned}
\mathbf{Pr}\left[\left|\sum_{i\in\mathcal{L}} \boldsymbol{z}_i \Delta_{s,s'}(i)\right| > \gamma_{s,s'}\Lambda\right] &\leq 2\exp\left(-\frac{\gamma_{s,s'}^2\Lambda^2}{2\sum_{i\in\mathcal{L}}\Delta_{s,s'}(i)^2}\right) \\
&\leq 2\exp\left(-\frac{\gamma_{s,s'}^2\Lambda^2}{2\sum_{i\in\mathcal{L}}\gamma_{s,s'}^2}\right) \\
&\leq 2\exp\left(-\frac{\Lambda^2}{2|I|}\right) && \text{(using } |\mathcal{L}| \leq |I|\text{)} \\
&\leq \frac{1}{4M^2}.
\end{aligned}$$

Combining this inequality with (11), by a union bound over the M^2 transitions (s,s'), there exists a choice of $z^* = (z_i)_{i\in\mathcal{L}}$ such that the corresponding $c^* \in \{0,1\}^I$ satisfies

$$|\mathcal{A}_{f_{c^*}}(s,s') - \mathcal{A}_{g_{c^*,\mathcal{L}}}(s,s')| \leq \gamma_{s,s'}\Lambda \qquad \text{for every transition } (s,s'). \tag{13}$$

We now consider two cases. First, if $\gamma_{s,s'} > 0$, then by (12) and (13) we have that

$$\frac{|\mathcal{A}_{f_{c^*}}(s,s') - \mathcal{A}_{g_{c^*,\mathcal{L}}}(s,s')|}{\mathcal{A}_{f_{c^*}}(s,s')} \leq \frac{\Lambda}{\rho} \leq \frac{1}{25M}.$$

Otherwise, $\gamma_{s,s'} = 0$, and in this case we must have $\mathcal{A}_{f_{c^*}}(s,s') = \mathcal{A}_{g_{c^*,\mathcal{L}}}(s,s')$. Hence we have that for every transition (s,s'),

$$\mathcal{A}_{g_{c^*,\mathcal{L}}}(s,s') \leq \left(1 + \frac{1}{25M}\right)\mathcal{A}_{f_{c^*}}(s,s') \leq \left(1 + \frac{1}{10M}\right)\mathcal{A}_{f_{c^*}}(s,s')$$

and

$$\mathcal{A}_{g_{c^*,\mathcal{L}}}(s,s') \geq \left(1 - \frac{1}{25M}\right)\mathcal{A}_{f_{c^*}}(s,s') \geq \left(1 + \frac{1}{10M}\right)^{-1}\mathcal{A}_{f_{c^*}}(s,s').$$

Hence $\mathcal{A}_{f_{c^*}}$ and $\mathcal{A}_{g_{c^*,\mathcal{L}}}$ are $\frac{1}{10M}$-indistinguishable. This completes the proof of Claim 37. □

6.2 The lower bound for symmetric functions

Let $\mathsf{Sym} \subseteq \{0,1\}^{2^n}$ denote the set of symmetric functions. We use the framework from the previous subsection to prove the following:

Theorem 38. Any $(1/100, 1/3)$-tester for symmetric functions must use at least $\Omega(\log n)$ space.

It is fairly easy to build a corruptible block system for symmetric functions: each block is a weight level of the hypercube centered around Hamming weight $n/2$. Let

$$I = \left\{ \ell \in \{0,1,\ldots,n\} : |\ell - n/2| < \frac{\sqrt{n}}{2} \right\}.$$

For each $\ell \in I$, consider the block

$$B_\ell = \{x \in \{0,1\}^n : |x| = \ell\}.$$

For each block, choose an arbitrary subset of essentially half of the points in the block:

$$C_\ell \subseteq B_\ell, \qquad |C_\ell| = \left\lfloor \frac{|B_\ell|}{2} \right\rfloor.$$

We set the background function to be the constant-0 function, that is, $b(x) := 0$ for all $x \in \{0,1\}^n \setminus \cup_{i \in I} B_i$. We now verify that this is an $(I, 0.6)$-block system that corrupts symmetric functions.

Claim 39. For sufficiently large n and $\beta = 0.6$, we have $\sum_{i \in I} |B_i| \geq \beta\, 2^n$. That is, at least 60% of the points in $\{0,1\}^n$ have Hamming weight in I.

Proof. Let $\boldsymbol{x}$ be a uniformly random string in $\{0,1\}^n$, and let $\boldsymbol{w}$ be its Hamming weight. Then $\boldsymbol{w} \sim \mathrm{Bin}(n, 1/2)$. By the central limit theorem, for sufficiently large n the event $|\boldsymbol{w} - n/2| < \sqrt{n}/2$ has probability at least 0.6. When this happens, $\boldsymbol{x}$ belongs to a block B_i for which $i \in I$. □

Claim 40. For all $i \in I$, we have $|B_i| \leq 2^n/|I|$.

Proof. Every i satisfies $|B_i| \leq \binom{n}{\lfloor n/2 \rfloor}$. It is known that

$$\binom{n}{\lfloor n/2 \rfloor} \leq \sqrt{\frac{2}{\pi}} \cdot \frac{2^n}{\sqrt{n}}.$$

The claim follows because $|I| \leq \sqrt{n} + 1 \leq \sqrt{\pi/2}\,\sqrt{n}$ for sufficiently large n. □

(1.) Any extension of the background map that labels each block with a fixed bit is consistent with $\mathcal{P}$: For any $c \in \{0,1\}^I$, the function f_c is symmetric, since the function is 0 outside the middle layers, and a constant across each middle layer.

(2.) Blocks are corruptible: Let $\emptyset \neq J \subseteq I$ and $c \in \{0,1\}^I$. On each layer $\ell \in J$, the function $g_{c,J}$ labels points in C_ℓ by $1 - c_\ell$ and points in $B_\ell \setminus C_\ell$ by c_ℓ. To make $g_{c,J}$ symmetric, we must therefore change at least $\min\{|C_\ell|, |B_\ell \setminus C_\ell|\}$ points on each such layer. For sufficiently large n, we have

$$|B_\ell \setminus C_\ell| \geq |C_\ell| \geq \frac{1}{3}|B_\ell|.$$

Consequently,

$$d(g_{c,J}, \mathsf{Sym}) \geq \frac{1}{3} \sum_{\ell \in J} \frac{|B_\ell|}{2^n}.$$

By Theorem 36, every $(0.06, 1/3)$-tester for symmetric functions must satisfy

$$M^3 \sqrt{\log(M)} = \Omega(\sqrt{|I|}).$$

Since $|I| = \Theta(\sqrt{n})$, this gives a lower bound of $M = \tilde{\Omega}(n^{1/12})$ states and hence $\Omega(\log n)$ bits of space. This concludes the proof of Theorem 38. □

6.3 Space lower bound for monotonicity

Let $\mathsf{Mon} \subseteq \{0,1\}^{2^n}$ denote the set of monotone functions. Now we use the framework from Section 6.1 to prove a space lower bound for testing monotonicity:

Theorem 41. There exists a constant $\varepsilon > 0$ such that any $(\varepsilon, 1/3)$-tester for monotonicity must use $\Omega(\sqrt{n})$ bits of space.

Building a corruptible block system for monotonicity is more complicated than it was for symmetry. To do this, we will use a partition of the hypercube inspired by "Talagrand partitions" (used, for instance, in [Kan13, CWX17, CCC+26]). Roughly speaking, such a partition is defined by a large collection of monotone terms $T_1, \ldots, T_N$. For each term T_i, the corresponding block consists of the points that satisfy T_i and no other term T_j.

Let us proceed to the detailed construction. We will use the first $n-2$ coordinates to define sets $U_1, \ldots, U_N$, and coordinates $n-1$ and n to define the sub-blocks C_i inside the blocks $B_i = U_i \times \{0,1\}^2$. We set

$$d := n-2, \qquad k := \lfloor \sqrt{d} \rfloor, \qquad N := 2^k, \qquad I := [N].$$

We will use the probabilistic method to show that a corruptible block system exists. Draw a sequence of N i.i.d. terms $\boldsymbol{T}_1, \ldots, \boldsymbol{T}_N$, where each term is obtained by choosing a uniformly random k-element subset $\boldsymbol{S}_i \subseteq [d]$ and setting $\boldsymbol{T}_i(x) := \bigwedge_{j \in \boldsymbol{S}_i} x_j$.

For $y \in \{0,1\}^d$, let

$$\boldsymbol{r}(y) = |\{i : \boldsymbol{T}_i(y) = 1\}|,$$

and, for each $i \in [N]$, define its *unique-satisfaction block* by

$$\boldsymbol{U}_i = \{y : \boldsymbol{T}_i(y) = 1 \text{ and } \boldsymbol{r}(y) = 1\}.$$

Fix $y \in \{0,1\}^d$ with $w := |y|$ satisfying $|w - d/2| \leq \sqrt{d}$. We first consider the probability that y satisfies a random term $\boldsymbol{T}_i$. Since the k coordinates are sampled without replacement, for sufficiently large d we have

$$p_y := \mathbf{Pr}[\boldsymbol{T}_i(y) = 1] = \frac{\binom{w}{k}}{\binom{d}{k}} = \prod_{j=0}^{k-1} \frac{w-j}{d-j},$$

from which it is easy to verify that

$$\left(\frac{1}{2}-\frac{2}{\sqrt{d}}\right)^k \le p_y \le \left(\frac{1}{2}+\frac{1}{\sqrt{d}}\right)^k.$$

Independence of the terms gives

$$\begin{aligned}
\mathbf{Pr}_{\boldsymbol{T}_1,\ldots,\boldsymbol{T}_N}[\boldsymbol{r}(y)=1] &= Np_y(1-p_y)^{N-1} \\
&\ge N\left(\frac{1}{2}-\frac{2}{\sqrt{d}}\right)^k\left(1-\left(\frac{1}{2}+\frac{1}{\sqrt{d}}\right)^k\right)^{N-1} \\
&= \left(1-\frac{4}{\sqrt{d}}\right)^k\left(1-2^{-k}\left(1+\frac{2}{\sqrt{d}}\right)^k\right)^{2^k-1} \\
&\ge e^{-8-2e^2} > 0.
\end{aligned}$$

For the last inequality, we used that for large enough d, $(1-4/\sqrt{d})^k \ge e^{-8}$ and $(1+2/\sqrt{d})^k \le e^2$, together with $\ln(1-u) \ge -2u$ for $0 \le u \le 1/2$.

Using the central limit theorem, as we did in Claim 39, we know that for large enough d, at least $3/4$ of the points in $\{0,1\}^d$ have Hamming weight in $d/2 \pm \sqrt{d}$. Therefore,

$$\begin{aligned}
\mathbf{E}_{\boldsymbol{T}_1,\ldots,\boldsymbol{T}_N}\left[\sum_{i=1}^N |\boldsymbol{U}_i|\right] &= \sum_{y\in\{0,1\}^d} \mathbf{Pr}_{\boldsymbol{T}_1,\ldots,\boldsymbol{T}_N}[\boldsymbol{r}(y)=1] \\
&\ge \sum_{\substack{y\in\{0,1\}^d \\ ||y|-d/2|\le\sqrt{d}}} \mathbf{Pr}_{\boldsymbol{T}_1,\ldots,\boldsymbol{T}_N}[\boldsymbol{r}(y)=1] \\
&\ge \frac{3}{4}\cdot e^{-8-2e^2}\cdot 2^d.
\end{aligned}$$

We now fix $\beta := \frac{3}{4}\cdot e^{-8-2e^2} > 0$. Hence there is a fixed choice $T_1,\ldots,T_N$ of the terms for which

$$\sum_{i=1}^N \frac{|U_i|}{2^d} \ge \beta. \tag{14}$$

Furthermore, U_i is contained in the subcube satisfying T_i, so

$$\frac{|U_i|}{2^d} \le 2^{-k} = \frac{1}{N}. \tag{15}$$

Fix such an outcome of $T_1,\ldots,T_N$ satisfying Equation (14). In the subsequent discussion, when we refer to r and the sets $U_1,\ldots,U_N$, they are defined with respect to this fixed choice of $T_1,\ldots,T_N$. We now build the corruptible block system. Define

$$U^- := \{y\in\{0,1\}^d : r(y)=0\}, \qquad U^+ := \{y\in\{0,1\}^d : r(y)\ge 2\},$$

and define the blocks $B_1,\ldots,B_N \subset \{0,1\}^n$ by

$$B_i = U_i\times\{0,1\}^2, \qquad i\in I.$$

The complement $\{0,1\}^n \setminus \bigcup_{i\in I} B_i$ is partitioned into $B^+ = U^+ \times \{0,1\}^2$ and $B^- = U^- \times \{0,1\}^2$. Define the background map by $b(x) = 0$ on B^- and $b(x) = 1$ on B^+. Finally, set

$$C_i = U_i \times \{(0,0),(1,1)\}.$$

For $I = [N]$ and β as defined above, we now prove that the above is an (I,β)-block system that corrupts monotone functions.

We first show that this is an (I,β)-block system. For each $i \in I$, we have:

$$|B_i|/2^n = \frac{4|U_i|}{2^{d+2}} = \frac{|U_i|}{2^d}.$$

Thus $\sum_{i\in I} |B_i|/2^n \geq \beta$, and $|B_i| \leq 2^n/N$ for every $i \in I$.

(1.) Any extension of the background map that labels each block with a fixed bit is consistent with $\mathcal{P}$: Fix any $c \in \{0,1\}^N$. We show that f_c is monotone. The terms $T_1, \ldots, T_N$ are monotone, and each point in U_i satisfies T_i and no other term. Thus B^- is downward closed, B^+ is upward closed, and the following properties hold:

- if $x \in B_i$, $y \in B_j$, and $i \neq j$, then x, y are incomparable,
- if $x \in B_i$ and $y \preceq x$, then $y \in B^- \cup B_i$,
- if $x \in B_i$ and $x \preceq y$, then $y \in B_i \cup B^+$.

Hence, given $x \preceq y$, either $x \in B^-$, so $f_c(x) = 0 \leq f_c(y)$; or $y \in B^+$, so $f_c(y) = 1 \geq f_c(x)$; or $x, y \in B_i$ for some i, so $f_c(x) = f_c(y)$. In every case, $f_c(x) \leq f_c(y)$.

(2.) Blocks are corruptible: Fix a nonempty set $J \subseteq [N]$ and a string $c \in \{0,1\}^N$, and consider $g_{c,J}$. Fix $i \in J$. We consider two cases:

- If $c_i = 0$, then for every $u \in U_i$ the pair

  $$(u,0,0) \preceq (u,0,1)$$

  has $g_{c,J}(u,0,0) = 1$ but $g_{c,J}(u,0,1) = 0$.

- If $c_i = 1$, then for every $u \in U_i$ the pair

  $$(u,0,1) \preceq (u,1,1)$$

  has $g_{c,J}(u,0,1) = 1$ but $g_{c,J}(u,1,1) = 0$.

For each $i \in J$ and each $u \in U_i$, we have a pair in B_i that violates monotonicity. These pairs are disjoint, so $g_{c,J}$ has at least

$$\sum_{i\in J} |U_i| = \frac{1}{4} \sum_{i\in J} |B_i|$$

disjoint violating pairs. Making $g_{c,J}$ monotone requires changing its value on at least one point of each pair. Therefore,

$$d(g_{c,J}, \mathsf{Mon}) \geq \sum_{i \in J} \frac{|U_i|}{2^n} = \frac{1}{4} \sum_{i \in J} \frac{|B_i|}{2^n}.$$

Thus we have indeed built an (I, β)-block system that corrupts monotone functions, with $|I| = N = 2^k$. By Theorem 36, for $\varepsilon := \beta/10 > 0$, every $(\varepsilon, 1/3)$-tester with M states satisfies

$$M^3 \sqrt{\log(M)} = \Omega(\sqrt{|I|}) = \Omega(2^{k/2}).$$

Since $k = \lfloor \sqrt{n-2} \rfloor = \Theta(\sqrt{n})$, this implies $\log M = \Omega(\sqrt{n})$. Hence the tester must use $\Omega(\sqrt{n})$ bits of space, and Theorem 41 is proved. □

6.4 A space lower bound for linearity

Our main result in this section is the following lower bound for testing parities.

Theorem 42. Any $(1/100, 1/3)$-tester for linearity must use $\Omega(n)$ bits of space.

Throughout this section, we identify a vector $a \in \{0,1\}^n$ with the linear function $f_a(x) := a \cdot x$, where the inner product is over $\mathbb{F}_2$.

It suffices to rule out normal-form testers $\mathcal{A}$ using $m < n/7$ bits of memory for all sufficiently large n. Recall that a tester being in normal form means that for each state s' there exists $b_{s'} \in \{0,1\}$ such that for all s, x: $\delta_{s,s'}(x, 1 - b_{s'}) = 0$. In particular, we can denote the transition probability $\delta_{s,s'}(x, b_{s'})$ by $\delta_{s,s'}(x)$.

We'll use

$$N := 2^n \text{ and } M := 2^m.$$

For a Boolean function f, normal form gives

$$\mathcal{A}_f(s, s') = \sum_{x \in \{0,1\}^n} \mathbb{1}[f(x) = b_{s'}] \delta_{s,s'}(x).$$

We will construct a linear function f_a and a function g_a that is $1/10$-far from every linear function such that the Markov chains $\mathcal{A}_{f_a}$ and $\mathcal{A}_{g_a}$ are $1/(10M)$-indistinguishable.

Definition 43 (Heavy and light inputs)**.** Fix an integer $1 \leq \rho \leq N-1$. For each pair of states (s, s'), order the nonzero inputs $x \in \{0,1\}^n \setminus \{0^n\}$ by decreasing weight $\delta_{s,s'}(x)$, breaking ties arbitrarily.

- Let $\mathcal{H}_{s,s'}$ consist of the first ρ inputs in this ordering, called the *heavy inputs* for (s, s'), and let $\mathcal{L}_{s,s'} := \{0,1\}^n \setminus \mathcal{H}_{s,s'}$ be its complement.
- Define the global heavy and light sets by

$$\mathcal{H} := \{0^n\} \cup \bigcup_{(s,s')} \mathcal{H}_{s,s'}, \qquad \mathcal{L} := \{0,1\}^n \setminus \mathcal{H}.$$

  The sets $\mathcal{H}$ and $\mathcal{L}$ are disjoint, but the sets $\mathcal{H}_{s,s'}$ for different transitions may overlap. In particular, $|\mathcal{H}| \leq \rho M^2 + 1$. Including 0^n in $\mathcal{H}$ ensures that our sampled functions agree with every parity at the origin.

- Define the total, heavy, and light weights of a transition by

$$W_{s,s'} := \sum_{x\in\{0,1\}^n} \delta_{s,s'}(x), \qquad W_{s,s'}(\mathcal{H}) := \sum_{x\in\mathcal{H}} \delta_{s,s'}(x), \qquad W_{s,s'}(\mathcal{L}) := \sum_{x\in\mathcal{L}} \delta_{s,s'}(x).$$

Parameter choices. For the linearity lower bound, set $\rho := \lceil CM^4 \rceil$, where C is a sufficiently large absolute constant. Since $m < n/7$, for sufficiently large n we have

$$\rho M^2 + 1 \le (C+1)M^6 + 1 \le N/2.$$

In particular, the heavy sets are well defined and $|\mathcal{L}| \ge N/2$.

We first define a distribution of functions that agree with a given parity on $\mathcal{H}$.

Definition 44 (A distribution associated with each parity)**.** For a parity f_a, let $\mathcal{D}(f_a)$ be the distribution of Boolean functions $\boldsymbol{g}_a$ obtained as follows:

1. Set $\boldsymbol{g}_a(x) := f_a(x)$ for every $x \in \mathcal{H}$.
2. Independently for each $x \in \mathcal{L}$, choose $\boldsymbol{g}_a(x)$ uniformly from $\{0,1\}$.

Thus $\boldsymbol{g}_a \sim \mathcal{D}(f_a)$ agrees with f_a on the union of the heavy sets and at 0^n, and has independent, uniformly random values elsewhere.

We will compare $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\boldsymbol{g}_a}$ through their mean-field transition matrix.

Definition 45 (Mean-field chain)**.** For each parity f_a, define $\mathcal{A}_{\mathrm{Avg}_a}$ to be the Markov chain with the same state space and initial state as $\mathcal{A}$, and transition matrix

$$\mathcal{A}_{\mathrm{Avg}_a}(s,s') := \mathbf{E}_{\boldsymbol{g}_a\sim\mathcal{D}(f_a)}[\mathcal{A}_{\boldsymbol{g}_a}(s,s')] = \sum_{x\in\mathcal{H}} \mathbb{1}[f_a(x) = b_{s'}]\delta_{s,s'}(x) + \frac{1}{2}\sum_{x\in\mathcal{L}} \delta_{s,s'}(x).$$

This is the entry-wise average of the transition matrices induced by $\mathcal{D}(f_a)$.

We will define *good* parities below and show that at least a $9/10$ fraction of all 2^n parities are good. The following two lemmas then yield the theorem.

Lemma 46. For at least half of all good parities f_a, the Markov chains $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{f_a}$ are ε-indistinguishable, where $\varepsilon := 1/(30M)$.

Lemma 47. For every good parity f_a, there exists g_a in the support of $\mathcal{D}(f_a)$ such that

1. $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{g_a}$ are $1/(30M)$-indistinguishable; and
2. g_a is $1/10$-far from every parity.

Choose a good parity f_a satisfying Lemma 46, and then choose g_a using Lemma 47. By Observation 9, the chains $\mathcal{A}_{f_a}$ and $\mathcal{A}_{g_a}$ are indistinguishable with parameter $1/(10M)$. Since f_a is a parity and g_a is also $1/100$-far from every parity, Corollary 11 rules out a $(1/100, 1/3)$-tester for linearity. This proves Theorem 42, assuming the two lemmas.

6.4.1 Proof of Lemma 46

Definition 48 (Constraint sets)**.** For a parity f_a and a pair of states (s, s'), let $\mathcal{C}_{f_a}(s, s')$ be the family of constraints

$$\mathcal{C}_{f_a}(s, s') := \big(a \cdot x = b_{s'}\big)_{x \in \mathcal{H}_{s,s'}},$$

with one constraint for each heavy input in $\mathcal{H}_{s,s'}$.

Definition 49 (Good parities)**.** A parity f_a is (s, s')-*good* if it satisfies at least $1/4$ of the constraints in $\mathcal{C}_{f_a}(s, s')$. It is *good* if it is (s, s')-good for every pair of states (s, s').

To reuse the argument for juntas and LTFs, we state some of our results and definitions for arbitrary sets of parities. These estimates hold for any integer $1 \le \rho \le N - 1$ for which $|\mathcal{L}| \ge N/2$, with the heavy sets and $\mathcal{D}(f_a)$ defined using that specific value of ρ.

Lemma 50 (Good parities for a set S)**.** Let $S \subseteq \{0,1\}^n$ be nonempty. Then

$$\mathbf{Pr}_{\boldsymbol{a} \sim S}[f_{\boldsymbol{a}} \text{ is not good}] \le \frac{4NM^2}{\rho |S|}.$$

In particular, for $S = \{0,1\}^n$, we get that at least a $1 - 4M^2/\rho$ fraction of all parities are good.

Proof. Fix (s, s') and set $h(x) := (N/\rho)\mathbb{1}[x \in \mathcal{H}_{s,s'}]$. For every $a \in S$,

$$\widehat{h}(a) = \frac{1}{\rho} \sum_{x \in \mathcal{H}_{s,s'}} (-1)^{a \cdot x}.$$

If $|\widehat{h}(a)| \le 1/2$, each Boolean value occurs on at least one quarter of $\mathcal{H}_{s,s'}$, so f_a is (s, s')-good. By Parseval's identity we have:

$$\mathbf{E}_{\boldsymbol{a} \sim S}[\widehat{h}(\boldsymbol{a})^2] \le \frac{1}{|S|} \sum_{a \in \{0,1\}^n} \widehat{h}(a)^2 = \frac{1}{|S|} \mathbf{E}_{\boldsymbol{x} \sim \{0,1\}^n}[h(\boldsymbol{x})^2] = \frac{N}{\rho |S|}.$$

By Markov's inequality the probability of not being (s, s')-good is at most $4N/(\rho|S|)$. A union bound over the M^2 pairs proves the claim. □

Definition 51. For a Boolean function f, define

$$\mathcal{A}_f^{\mathcal{H}}(s, s') := \sum_{x \in \mathcal{H}} \mathbb{1}[f(x) = b_{s'}]\delta_{s,s'}(x), \qquad \mathcal{A}_f^{\mathcal{L}}(s, s') := \sum_{x \in \mathcal{L}} \mathbb{1}[f(x) = b_{s'}]\delta_{s,s'}(x).$$

These are the contributions of the heavy and light inputs, respectively, to the transition probability $\mathcal{A}_f(s, s')$.

For a parity f_a, the transition probabilities in the two chains are

$$\mathcal{A}_{f_a}(s, s') = \mathcal{A}_{f_a}^{\mathcal{H}}(s, s') + \sum_{x \in \mathcal{L}} \mathbb{1}[a \cdot x = b_{s'}]\delta_{s,s'}(x) \tag{16}$$

and

$$\mathcal{A}_{\mathrm{Avg}_a}(s,s') = \mathcal{A}^{\mathcal{H}}_{f_a}(s,s') + \frac{1}{2}\sum_{x\in\mathcal{L}} \delta_{s,s'}(x). \tag{17}$$

We define the difference between the two transition probabilities to be:

$$\mathrm{Loss}_{s,s'}(f_a, \mathrm{Avg}_a) := \left|\mathcal{A}_{\mathrm{Avg}_a}(s,s') - \mathcal{A}_{f_a}(s,s')\right|.$$

Note that this difference depends only on the light inputs.

Definition 52. We define the masked transition weights by

$$\delta'_{s,s'}(x) := \delta_{s,s'}(x)\mathbb{1}[x \in \mathcal{L}].$$

In particular, $\delta'_{s,s'}$ agrees with $\delta_{s,s'}$ on $\mathcal{L}$ and is zero on $\mathcal{H}$.

We will also use the following bounds:

Lemma 53. Let $S \subseteq \{0,1\}^n$ be nonempty. For at least 99/100 of the parities f_a, $a \in S$, every pair (s,s') satisfies

$$\mathrm{Loss}_{s,s'}(f_a, \mathrm{Avg}_a) \le 5M\sqrt{\frac{N}{|S|}}\,\|\delta'_{s,s'}\|_2.$$

In particular, taking $S = \{0,1\}^n$ gives the bound $5M\|\delta'_{s,s'}\|_2$.

Lemma 54. For every good parity f_a and every pair (s,s'),

$$\mathcal{A}_{\mathrm{Avg}_a}(s,s') \ge \frac{\sqrt{\rho}}{\sqrt{8}}\|\delta'_{s,s'}\|_2.$$

Proof of Lemma 46. Apply Lemmas 50 and 53 with $S = \{0,1\}^n$. At least 9/10 of all parities are good, and at most 1/100 of all parities fail the additive bound. Hence at least half of the good parities satisfy that bound for every transition. Fix such a parity f_a. Combining the additive bound with Lemma 54 gives, for every pair (s,s'),

$$\mathrm{Loss}_{s,s'}(f_a, \mathrm{Avg}_a) \le 5M\|\delta'_{s,s'}\|_2 \le \frac{5\sqrt{8}M}{\sqrt{\rho}}\mathcal{A}_{\mathrm{Avg}_a}(s,s') \le \frac{\mathcal{A}_{\mathrm{Avg}_a}(s,s')}{50M},$$

where the last inequality follows from $\rho \ge CM^4$ and C being a large enough constant. For $\varepsilon := 1/(30M)$, we have

$$(1+\varepsilon)^{-1} \le 1 - \frac{1}{50M} \quad \text{and} \quad 1 + \frac{1}{50M} \le 1+\varepsilon.$$

Thus the relative additive bound implies

$$(1+\varepsilon)^{-1}\mathcal{A}_{\mathrm{Avg}_a}(s,s') \le \mathcal{A}_{f_a}(s,s') \le (1+\varepsilon)\mathcal{A}_{\mathrm{Avg}_a}(s,s'),$$

including when $\mathcal{A}_{\mathrm{Avg}_a}(s,s') = 0$. This proves the claimed indistinguishability. □

6.4.2 Proof of Lemma 53

Fix (s, s'). Since $\mathbb{1}[a \cdot x = b_{s'}] = (1 + (-1)^{b_{s'}} \chi_a(x))/2$, we have

$$\begin{aligned}
\mathrm{Loss}_{s,s'}(f_a, \mathrm{Avg}_a) &= \left| \sum_{x \in \mathcal{L}} \left(\frac{1}{2} - \mathbb{1}[a \cdot x = b_{s'}] \right) \delta_{s,s'}(x) \right| \\
&= \frac{1}{2} \left| \sum_{x \in \mathcal{L}} \left((-1)^{b_{s'}} \chi_a(x) \right) \delta_{s,s'}(x) \right| \\
&= \frac{1}{2} \left| \sum_{x \in \{0,1\}^n} \chi_a(x) \delta'_{s,s'}(x) \right| \\
&= \frac{N}{2} \left| \widehat{\delta'_{s,s'}}(a) \right|. \qquad (18)
\end{aligned}$$

Thus the difference is the absolute value of a Fourier coefficient, scaled by $N/2$.

Proof of Lemma 53. Fix a pair (s, s'). By Parseval's identity:

$$\mathbf{E}_{\boldsymbol{a} \sim S} \left[\widehat{\delta'_{s,s'}}(\boldsymbol{a})^2 \right] = \frac{1}{|S|} \sum_{a \in S} \widehat{\delta'_{s,s'}}(a)^2 \leq \frac{1}{N|S|} \|\delta'_{s,s'}\|_2^2.$$

By Markov's inequality, we have

$$\left| \widehat{\delta'_{s,s'}}(\boldsymbol{a}) \right| \leq \frac{10M}{\sqrt{N|S|}} \|\delta'_{s,s'}\|_2, \qquad (19)$$

with probability at least $1 - 1/(100M^2)$ over the draw of $\boldsymbol{a}$. By a union bound over all M^2 pairs (s, s'), we have that with probability at least $99/100$ a random $\boldsymbol{a} \sim S$ satisfies Equation (19) for every pair (s, s'). Multiplying by $N/2$ and using Equation (18) proves the lemma. □

6.4.3 Proof of Lemma 54

We first bound the largest light transition weight using the heavy contribution.

Claim 55. For every good parity f_a and every pair (s, s'),

$$\max_{x \in \mathcal{L}} \delta_{s,s'}(x) \leq \frac{4}{\rho} \mathcal{A}^{\mathcal{H}}_{f_a}(s, s').$$

Proof. Because f_a is good, at least $\rho/4$ inputs $x \in \mathcal{H}_{s,s'}$ satisfy $f_a(x) = b_{s'}$. Each such input has weight at least that of any input in $\mathcal{L}$, by the definition of $\mathcal{H}_{s,s'}$. Their total contribution is at most $\mathcal{A}^{\mathcal{H}}_{f_a}(s, s')$, which proves the claim. □

Proof of Lemma 54. By Equation (17), $\|\delta'_{s,s'}\|_1 \leq 2\mathcal{A}_{\mathrm{Avg}_a}(s, s')$. Now using Claim 55 we get:

$$\begin{aligned}
\|\delta'_{s,s'}\|_2^2 &\leq \|\delta'_{s,s'}\|_1 \|\delta'_{s,s'}\|_\infty \\
&\leq 2\mathcal{A}_{\mathrm{Avg}_a}(s, s') \cdot \frac{4}{\rho} \mathcal{A}^{\mathcal{H}}_{f_a}(s, s') \\
&\leq \frac{8}{\rho} \mathcal{A}_{\mathrm{Avg}_a}(s, s')^2.
\end{aligned}$$

Hence we have $\|\delta'_{s,s'}\|_2 \leq \sqrt{\frac{8}{\rho}}\mathcal{A}_{\mathrm{Avg}_a}(s,s')$. □

6.4.4 Proof of Lemma 47

We first give a more general statement, which will also be of use for the k-junta and LTF lower bounds.

Lemma 56. Fix $1 \leq \rho \leq N-1$ such that $|\mathcal{L}| \geq N/2$ and consider the heavy sets for that specific value of ρ. Let f_a be a good parity and let $\mathcal{P}$ be a property. A draw $\boldsymbol{g}_a \sim \mathcal{D}(f_a)$ satisfies both

1. $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{\boldsymbol{g}_a}$ are $1/(30M)$-indistinguishable and
2. $d(\boldsymbol{g}_a, \mathcal{P}) \geq 1/10$

with probability at least

$$1 - \frac{2 \cdot 50^2 \cdot M^4}{\rho} - |\mathcal{P}| \cdot e^{-\Omega(N)}.$$

Proof. For the first point, fix a transition (s,s') and let $\mu := \mathcal{A}_{\mathrm{Avg}_a}(s,s')$. For a function $\boldsymbol{g}_a \sim \mathcal{D}(f_a)$, we have that $\boldsymbol{g}_a(x) = f_a(x)$ if $x \in \mathcal{H}$ and otherwise $\mathbf{g}_a(x)$ is an independent random bit. Therefore, by Lemma 54,

$$\mathbf{E}_{\boldsymbol{g}_a}[\mathcal{A}_{\boldsymbol{g}_a}(s,s')] = \mu, \qquad \mathbf{Var}_{\boldsymbol{g}_a}[\mathcal{A}_{\boldsymbol{g}_a}(s,s')] = \frac{1}{4}\|\delta'_{s,s'}\|_2^2 \leq \frac{2\mu^2}{\rho}.$$

If $\mu = 0$, we have $\mathcal{A}_{\boldsymbol{g}_a}(s,s') = \mu$ with probability 1. Otherwise, by Chebyshev's inequality, we have:

$$\mathbf{Pr}_{\boldsymbol{g}_a}\left[|\mathcal{A}_{\boldsymbol{g}_a}(s,s') - \mu| > \frac{\mu}{50M}\right] \leq \frac{2 \cdot 50^2 \cdot M^2}{\rho}.$$

Hence, by a union bound over all M^2 transitions, we have with probability $1 - 2 \cdot 50^2 \cdot M^4/\rho$ that for all (s,s'):

$$\frac{|\mathcal{A}_{\boldsymbol{g}_a}(s,s') - \mathcal{A}_{\mathrm{Avg}_a}(s,s')|}{\mathcal{A}_{\mathrm{Avg}_a}(s,s')} \leq 1/(50M) \quad \text{or} \quad \mathcal{A}_{\mathrm{Avg}_a}(s,s') = \mathcal{A}_{\boldsymbol{g}_a}(s,s') = 0$$

which implies

$$\left(1 + \frac{1}{30M}\right)^{-1} \mathcal{A}_{\mathrm{Avg}_a}(s,s') \leq \mathcal{A}_{\boldsymbol{g}_a}(s,s') \leq \left(1 + \frac{1}{30M}\right) \mathcal{A}_{\mathrm{Avg}_a}(s,s').$$

Hence, with probability $1 - 2 \cdot 50^2 \cdot M^4/\rho$ we have that $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{\boldsymbol{g}_a}$ are $1/(30M)$-indistinguishable.

We now turn to proving the second point. Fix a Boolean function $h \in \mathcal{P}$. The number of points $x \in \mathcal{L}$ such that $h(x) \neq \boldsymbol{g}_a(x)$ follows a binomial distribution with parameters $|\mathcal{L}|$ and $1/2$, since for each $x \in \mathcal{L}$, $\boldsymbol{g}_a(x)$ is an independent random bit. By a Chernoff bound we have

$$\mathbf{Pr}[d(\boldsymbol{g}_a, h) \leq 1/10] \leq e^{-\Omega(N)}.$$

By a union bound over all $h \in \mathcal{P}$, the probability that $d(\boldsymbol{g}_a, \mathcal{P}) \leq 1/10$ is at most $|\mathcal{P}|e^{-\Omega(N)}$.

The lemma now follows by a union bound over events. □

Proof of Lemma 47. Apply Lemma 56 with $\mathcal{P}$ equal to the set of all N parities. Since $\rho \geq CM^4$ and C is a large constant, for large enough n we have:

$$\mathbf{Pr}_{\boldsymbol{g}_a}[\mathcal{A}_{\mathrm{Avg}_a} \text{ and } \mathcal{A}_{\boldsymbol{g}_a} \text{ are } 1/(30M)\text{-indistinguishable and } d(\boldsymbol{g}_a, \mathcal{P}) \geq 1/10] > 0.$$

So there exists a function g_a in the support of $\mathcal{D}(f_a)$ that satisfies the conditions of Lemma 47. □

6.5 A space lower bound for k-juntas

In this section, we show that testing whether a function is a k-junta requires $\Omega(k \log n)$ bits of memory, whenever $k \leq n^{0.99}$ (Note that the choice of 0.99 is arbitrary, the proof would work for any constant $c < 1$). To prove the lower bound, our yes instances will be *k-parities* (parities on exactly k variables). This will allow us to reuse results from Section 6.4 that we proved for arbitrary subsets of parity functions.

Theorem 57. Let $k : \mathbb{N} \to \mathbb{N}$ satisfy $1 \leq k(n) \leq n^{0.99}$ for all sufficiently large n. Every $(1/10, 1/3)$-tester for k-juntas must use at least $\Omega(k \log n)$ bits of space.

We let

$$\mathrm{Par}(k) := \{a \in \{0,1\}^n : |a| = k\}, \qquad B := |\mathrm{Par}(k)| = \binom{n}{k}, \qquad N := 2^n,$$

and $C > 0$ be a large enough constant. As before, a denotes a vector in $\{0,1\}^n$ and f_a denotes the parity function $f_a(x) := a \cdot x$. Since

$$\log B \geq k \log(n/k) \geq 0.01k \log n,$$

and turning a tester into one in normal form costs at most one bit of memory, it suffices to rule out normal-form testers with M states satisfying $M^{12} \leq B$. Fix such a tester $\mathcal{A}$ and set

$$\rho := \left\lfloor \frac{N}{4M^2} \right\rfloor.$$

We use the heavy sets, the notion of a parity being good, the distributions $\mathcal{D}(f_a)$, and the mean-field chains from Section 6.4, with this value of ρ. For sufficiently large n and using $M^{12} \leq B$ we get:

$$\rho B \geq \frac{NB}{8M^2} \geq \frac{NM^4\sqrt{B}}{8} \geq CNM^4.$$

Furthermore, since $B \leq N$, this also gives $\rho \geq CM^4$. Finally, we have the following bound on the size of the heavy and light sets:

$$|\mathcal{H}| \leq 1 + \rho M^2 \leq 1 + N/4 \leq N/2, \qquad |\mathcal{L}| \geq N/2.$$

Lemma 58. With the parameters above, the following two points hold:

1. At least $9/10$ of the parities in $\mathrm{Par}(k)$ are good,
2. for at least half of these good parities f_a, the chains $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ are $1/(30M)$-indistinguishable.

Proof. Apply Lemma 50 to $S = \mathrm{Par}(k)$. The probability that for a uniformly chosen $a \in S$, f_a is not good is at most

$$\frac{4NM^2}{\rho B} \le \frac{4}{CM^2} \le \frac{1}{10}.$$

By Lemma 53, at least 99/100 of parities in S satisfy:

$$\left|\mathcal{A}_{f_a}(s,s') - \mathcal{A}_{\mathrm{Avg}_a}(s,s')\right| \le 5M\sqrt{\frac{N}{B}}\,\|\delta'_{s,s'}\|_2 \quad \text{for all } (s,s').$$

So, at least half of the good parities satisfy the above. For any such f_a, Lemma 54 gives, for every pair (s,s'),

$$\begin{aligned}\left|\mathcal{A}_{f_a}(s,s') - \mathcal{A}_{\mathrm{Avg}_a}(s,s')\right| &\le 5M\sqrt{N/B}\,\|\delta'_{s,s'}\|_2 \\ &\le \frac{5\sqrt{8}\,M\sqrt{N}}{\sqrt{\rho B}}\,\mathcal{A}_{\mathrm{Avg}_a}(s,s') \\ &\le \frac{\mathcal{A}_{\mathrm{Avg}_a}(s,s')}{50M}.\end{aligned}$$

This implies:

$$\left(1+\frac{1}{30M}\right)^{-1}\mathcal{A}_{\mathrm{Avg}_a}(s,s') \le \mathcal{A}_{f_a}(s,s') \le \left(1+\frac{1}{30M}\right)\mathcal{A}_{\mathrm{Avg}_a}(s,s'),$$

proving that $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ are $1/(30M)$-indistinguishable.

□

Proof of Theorem 57. Let $\mathcal{J}_{n,k}$ be the class of k-juntas. We have that $|\mathcal{J}_{n,k}| \le B\,2^{2^k}$. Because $k \le n^{0.99}$ we have $\log(B) + 2^k = o(N)$, and thus $|\mathcal{J}_{n,k}|e^{-\Omega(N)} = o(1)$. Choose a good $a \in \mathrm{Par}(k)$ satisfying Lemma 58. By Lemma 56, with probability at least

$$1 - \frac{2\cdot 50^2\cdot M^4}{\rho} - |\mathcal{J}_{n,k}|e^{-\Omega(N)} \ge 1 - \frac{2\cdot 50^2}{C} - o(1) > 0,$$

a draw $\boldsymbol{g}_a \sim \mathcal{D}(f_a)$ is 1/10-far from $\mathcal{J}_{n,k}$ and also $\mathcal{A}_{\boldsymbol{g}_a}$ is $1/(30M)$-indistinguishable from $\mathcal{A}_{\mathrm{Avg}_a}$. So fix such a choice of g_a. By Observation 9, $\mathcal{A}_{f_a}$ and $\mathcal{A}_{g_a}$ are indistinguishable with parameter $1/(10M)$. Since f_a is a k-junta and g_a is also 1/10-far from every k-junta, we can conclude from Corollary 11 that $\mathcal{A}$ isn't a $(1/10, 1/3)$-tester for k-juntas. □

6.6 A space lower bound for LTFs

To prove our lower bound for LTFs, we reuse our lower bound for k-juntas. As such, we will focus on the specific case of parities of size 1: the functions $f(x) := x_i$, which are LTFs.

Corollary 59. Every $(1/10, 1/3)$-tester for LTFs requires $\Omega(\log n)$ bits of memory.

Proof. Let $\mathcal{A}$ be a tester in normal form that has $M \le n^{1/12}$ states, we will show $\mathcal{A}$ can't be a $(1/10, 1/3)$-tester for LTFs. We use the same parameters as the k-junta lower bound from Section 6.5 with $k = 1$:

$$N = 2^n, \quad \rho = \lfloor N/(4M^2)\rfloor, \quad \rho \ge CM^4 \text{ where } C \text{ is a sufficiently large constant.}$$

For large enough n, we have $|\mathcal{L}| \geq N/2$. By Lemma 58, there is a good $a \in \mathrm{Par}(1)$ such that $\mathcal{A}_{f_a}$ and $\mathcal{A}_{\mathrm{Avg}_a}$ are $1/(30M)$-indistinguishable. Let $\mathcal{T}$ denote the class of LTFs. By Chow's theorem [Cho61], an LTF is determined by its Fourier coefficients of degrees zero and one. Each of these $n+1$ coefficients is an integer multiple of $1/N$ in $[-1,1]$, so $|\mathcal{T}| = 2^{O(n^2)}$. Apply Lemma 56 to f_a and $\mathcal{T}$. The probability that $\boldsymbol{g}_a \sim \mathcal{D}(f_a)$ is $1/10$-far from every LTF and that $\mathcal{A}_{\mathrm{Avg}_a}$ and $\mathcal{A}_{\boldsymbol{g}_a}$ are $1/(30M)$-indistinguishable is at least

$$1 - \frac{2 \cdot 50^2 \cdot M^4}{\rho} - |\mathcal{T}| e^{-\Omega(N)} > 0$$

for sufficiently large n and by setting C to be large enough. We now fix a function g_a in the support of $\mathcal{D}(f_a)$ that satisfies these two conditions. By Observation 9, $\mathcal{A}_{f_a}$ and $\mathcal{A}_{g_a}$ are $1/(10M)$-indistinguishable. Since f_a is a 1-junta and thus an LTF, but g_a is also $1/10$-far from every LTF, we have from Corollary 11 that $\mathcal{A}$ isn't a $(1/10, 1/3)$-tester for LTFs.

Thus every tester for LTFs in normal form has more than $n^{1/12}$ states. Since putting a tester in normal form only doubles the number of states, we obtain the $\Omega(\log n)$ lower bound on the number of bits of memory needed to test LTFs. □

References

[ABC+26] Amir Azarmehr, Soheil Behnezhad, Lily Chung, Alma Ghafari, Jane Lange, and Ronitt Rubinfeld. Streaming algorithms for monotonicity testing. *arXiv preprint arXiv:2608.07073*, 2026. 5

[AHW16] Noga Alon, Rani Hod, and Amit Weinstein. On active and passive testing. *Combinatorics, Probability and Computing*, 25(1):1–20, 2016. 3

[BB16] A. Belovs and E. Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th ACM Symposium on Theory of Computing*, 2016. 12

[BBBY12] 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. 3

[BBM12] Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. *Computational Complexity*, 21(2):311–358, 2012. 4

[BHZ22] Nader H. Bshouty and Catherine A. Haddad-Zaknoon. On testing decision tree. In *39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 17:1–17:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. 5

[Bla08] Eric Blais. Improved bounds for testing juntas. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (RANDOM 2008)*, pages 317–330, 2008. 1

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 151–158, 2009. 1, 4, 6

[Bla16] Eric Blais. Testing juntas nearly optimally. *SIAM Journal on Computing*, 45(5):1558–1571, 2016. 27

[BLR93] 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. 1, 3, 4, 6, 27

[BLT22] Guy Blanc, Jane Lange, and Li-Yang Tan. Reconstructing decision trees. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 24:1–24:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. 5

[BOS22] Tomer Berg, Or Ordentlich, and Ofer Shayevitz. On the memory complexity of uniformity testing. In *Proceedings of the 35th Conference on Learning Theory*, volume 178 of *Proceedings of Machine Learning Research*, pages 3506–3523. PMLR, 2022. 5

[BS21] Gabriel Bathie and Tatiana Starikovskaya. Property testing of regular languages with applications to streaming property testing of visibly pushdown languages. In *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*,

volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 119:1–119:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. 5

[BSSVW03] Eli Ben-Sasson, Madhu Sudan, Salil P. Vadhan, and Avi Wigderson. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing (STOC)*, pages 612–621. ACM, 2003. 5

[BT96] N. Bshouty and C. Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, 1996. 7

[BWY15] Eric Blais, Amit Weinstein, and Yuichi Yoshida. Partially symmetric functions are efficiently isomorphism-testable. *SIAM Journal on Computing*, 44(2):411–432, 2015. 3, 4, 6, 27

[BY22] Arnab Bhattacharyya and Yuichi Yoshida. *Property Testing - Problems and Techniques*. Springer, 2022. 1

[CCC+26] Mark Chen, Xi Chen, Hao Cui, William Pires, and Jonah Stockwell. Boolean function monotonicity testing requires (almost) $n^{1/2}$ queries. In *Proceedings of the 58th Annual ACM Symposium on Theory of Computing, STOC 2026, Salt Lake City, UT, USA, June 22-26, 2026*, pages 584–594. ACM, 2026. 1, 4, 12, 36

[CCP+26] Xi Chen, Clement Canonne, William Pires, Toni Pitassi, and Rocco A. Servedio. More on the space complexity of property testing. Manuscript, 2026. 2, 4

[CFPS20] Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler. Testable properties in general graphs and random order streaming. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. 5

[CG18] Clément L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational Complexity*, 27(4):671–716, 2018. 1, 5

[CH70] Thomas M. Cover and Martin E. Hellman. Learning with finite memory. *Annals of Mathematical Statistics*, 41(3):765–782, 1970. 5

[CHFSV19] Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Computing*, 32(1):41–57, 2019. 5

[Cho61] C.K. Chow. On the characterization of threshold functions. In *Proceedings of the Symposium on Switching Circuit Theory and Logical Design (FOCS)*, pages 34–38, 1961. 47

[CS13] Deeparnab Chakrabarty and C. Seshadhri. A $o(n)$ monotonicity tester for boolean functions over the hypercube. In *Proceedings of the 45th ACM Symposium on Theory of Computing*, pages 411–418, 2013. 1

[CST14] Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for testing monotonicity. In *Proceedings of the 55th IEEE Symposium on Foundations of Computer Science*, pages 286–295, 2014. 1

[CWX17] 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. 12, 36

[DLM+07] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *48th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2007, Providence, RI, USA, October 20-23, 2007, Proceedings*, pages 549–558. IEEE Computer Society, 2007. 5

[DLM+11] Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Rocco A. Servedio, and Andrew Wan. Efficiently testing sparse $GF(2)$ polynomials. *Algorithmica*, 61(3):580–605, 2011. 5

[DSS11] Dana Dachman-Soled and Rocco A. Servedio. A canonical form for testing boolean function properties. In *RANDOM*, pages 460–471, 2011. 3, 4, 6, 27

[FKR+04] Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. *Journal of Computer and System Sciences*, 68(4):753–787, 2004. 1

[FKSV02] Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. Testing and spot-checking of data streams. *Algorithmica*, 34(1):67–80, 2002. 5

[FLN+02] Eldar Fischer, Eric P. Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *STOC*, pages 474–483. ACM, 2002. 20

[FMdRS16] Nathanaël François, Frédéric Magniez, Michel de Rougemont, and Olivier Serre. Streaming property testing of visibly pushdown languages. In *24th Annual European Symposium on Algorithms (ESA 2016)*, volume 57 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. 5

[FPPR26] Renato Ferreira Pinto, Jr., Diptaksho Palit, and Sofya Raskhodnikova. Computational complexity in property testing. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 408–440. SIAM, 2026. 1, 5

[GGL+00] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. 1, 7

[Gol10] O. Goldreich, editor. *Property Testing: Current Research and Surveys*. Springer, 2010. LNCS 6390. 1

[Gol17] O. Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. 1

[GRT18] Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC 2018)*, pages 297–310. ACM, 2018. 2

[GS10] Oded Goldreich and Or Sheffet. On the randomness complexity of property testing. *Computational Complexity*, 19(1):99–133, 2010. 1, 5

[Har19] Nathaniel Harms. Testing halfspaces over rotation-invariant distributions. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 694–713. SIAM, 2019. 3

[Kan13] Daniel M. Kane. A monotone function given by a low-depth decision tree that is not an approximate junta. *Theory Comput.*, 9:587–592, 2013. 12, 36

[KMS18] Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric-type theorems. *SIAM J. Comput.*, 47(6):2238–2276, 2018. 1, 4

[KST23a] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Properly learning decision trees with queries is NP-hard. In *64th IEEE Annual Symposium on Foundations of Computer Science (FOCS 2023)*, pages 2383–2407. IEEE, 2023. 5

[KST23b] Caleb Koch, Carmen Strassle, and Li-Yang Tan. Superpolynomial lower bounds for decision tree learning and testing. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1962–1994. SIAM, 2023. 5

[LNS08] Oded Lachish, Ilan Newman, and Asaf Shapira. Space complexity vs. query complexity. *Computational Complexity*, 17:70–93, 2008. 5

[LR83] Frank Thomson Leighton and Ronald L Rivest. The Markov Chain Tree Theorem. Technical report, MIT Laboratory for Computer Science, 1983. 9, 52

[LR01] Eric P. Lehman and Dana Ron. On disjoint chains of subsets. *J. Comb. Theory A*, 94(2):399–404, 2001. 20

[Min25] Dor Minzer. A distance amplification lemma for monotonicity. *arXiv preprint arXiv:2512.13566*, 2025. 1, 4

[MMPS17] Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, volume 80 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 131:1–131:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017. 5

[MN15] Elchanan Mossel and Joe Neeman. Robust optimality of gaussian noise stability. *Journal of the European Mathematical Society*, 17:433–482, 2015. 3

[MORS10] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing halfspaces. *SIAM Journal on Computing*, 39(5):2004–2047, 2010. 3, 4, 6, 7, 28, 29, 30

[O'D14] Ryan O'Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014. 18, 19, 24

[Raz17] Ran Raz. A time-space lower bound for a large class of learning problems. In *58th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2017)*, pages 732–742. IEEE Computer Society, 2017. 2

[Raz19] Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. *Journal of the ACM*, 66(1):3:1–3:18, 2019. 2

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014. 23

[Zim07] Marius Zimand. On derandomizing probabilistic sublinear-time algorithms. In *22nd Annual IEEE Conference on Computational Complexity (CCC 2007)*, pages 1–9. IEEE Computer Society, 2007. 5

A General tools

A.1 The Markov chain tree theorem

In this section, we recall the Markov chain tree theorem of [LR83], which will allow us to express the long-run state probabilities of the Markov chains induced by an algorithm.

We recall some basic notions from the theory of discrete-time Markov chains. In this paper, every Markov chain $\mathcal{C}$ will be over a finite state space. We say that states s and s' in a given Markov chain *communicate* with each other if s is reachable from s' by a sequence of transitions that have positive probability and vice versa. A Markov chain is *irreducible* if every state communicates with every other state. A state s of a Markov chain is *aperiodic* if the GCD of the positive integers t for which a return from s to itself in t steps has positive probability is 1; a Markov chain is said to be aperiodic if all of its states are aperiodic. Finally, a Markov chain is said to be *ergodic* if there is some k such that all entries of M^k are positive, where M is its transition matrix. In particular, a Markov chain is ergodic if and only if it is irreducible and aperiodic.

Definition 60. Given a Markov chain $\mathcal{C}$ with state space S, a *closed communicating class* of $\mathcal{C}$ is a maximal set of vertices V such that for all $u, v \in V$ we have that u and v communicate and there are no transitions from any $u \in V$ to a vertex $v \notin V$.

Let $\mathcal{C}$ be a finite Markov chain with state space S and initial state $s^\star$. We view $\mathcal{C}$ as a directed graph on S, with an edge from s to s' if and only if $\mathcal{C}(s, s') > 0$. Under this view, a closed communicating class is a set V of vertices, such that for any $u, v \in V$ there is a directed path from u to v, and there are no edges leaving V.

Definition 61 (Arborescence of a chain)**.** Given a Markov chain $\mathcal{C}$ with d closed communicating classes, an *arborescence* of $\mathcal{C}$ is a spanning directed forest in the directed graph of $\mathcal{C}$ with exactly d components, whose edges are directed toward the root of each tree. Equivalently, every non-root vertex has outdegree one, every root has outdegree zero, and the forest contains $|S| - d$ edges.

With the above definition in mind, we let $\mathcal{F}(\mathcal{C}) := \{\, \text{all arborescences of } \mathcal{C} \,\}$, and for a state s, let

$$\mathcal{F}_s(\mathcal{C}) := \{\, F \in \mathcal{F}(\mathcal{C}) : s^\star \text{ belongs to the tree rooted at } s \,\}.$$

Note that the above might be empty. For such an arborescence F of $\mathcal{C}$, define

$$W(\mathcal{C}, F) := \prod_{(u,v)\in F} \mathcal{C}(u, v).$$

Finally, we define

$$||W(\mathcal{C})|| = \sum_{F\in\mathcal{F}(\mathcal{C})} W(\mathcal{C}, F) \quad \text{and} \quad ||W_s(\mathcal{C})|| = \sum_{F\in\mathcal{F}_s(\mathcal{C})} W(\mathcal{C}, F).$$

If $\mathcal{F}_s(\mathcal{C}) = \emptyset$ we define $||W_s(\mathcal{C})|| = 0$.

Theorem 62 (Markov chain tree theorem)**.** Let $\mathcal{C}$ be a Markov chain over state space S with start state $s^\star$. For a state s, recall that

$$\widehat{p}_{\mathcal{C}}(s) := \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbf{Pr}[\mathcal{C} \text{ is in state } s \text{ at time } t]$$

is the long-run probability that $\mathcal{C}$ is in state s. Then we have

$$\widehat{p}_{\mathcal{C}}(s) = \frac{||W_s(\mathcal{C})||}{||W(\mathcal{C})||}.$$

A.2 From relative error to close long-term probabilities

Lemma 10. Let $\mathcal{C}_1, \mathcal{C}_2$ be two M-state Markov chains, with the same start state, such that they are $\frac{1}{10M}$-indistinguishable. Then for every state s, it holds that $\widehat{p}_{\mathcal{C}_1}(s) \leq 1.5\widehat{p}_{\mathcal{C}_2}(s)$.

Proof. Let $[M]$ be the set of states of the chains and $s \in [M]$ be any state. Both chains have the same start state $s^\star$, hence we will use the Markov chain tree theorem to show that $\widehat{p}_{\mathcal{C}_1}(s) \leq 1.5\widehat{p}_{\mathcal{C}_2}(s)$. First, observe that since $\mathcal{C}_1$ and $\mathcal{C}_2$ are $\frac{1}{10M}$-indistinguishable, it must be that

$$\mathcal{C}_1(s, s') = 0 \iff \mathcal{C}_2(s, s') = 0.$$

Hence, both chains have the same set of edges when viewed as graphs over $[M]$. So F is an arborescence in $\mathcal{C}_1$ if and only if it's also one in $\mathcal{C}_2$. In particular, we have $\mathcal{F}(\mathcal{C}_1) = \mathcal{F}(\mathcal{C}_2)$ and also $\mathcal{F}_s(\mathcal{C}_1) = \mathcal{F}_s(\mathcal{C}_2)$.

Hence, we will bound the ratios $\frac{W(\mathcal{C}_1,F)}{W(\mathcal{C}_2,F)}$ and $\frac{W(\mathcal{C}_2,F)}{W(\mathcal{C}_1,F)}$ whenever F doesn't have weight 0. First, we have that:

$$\frac{W(\mathcal{C}_1, F)}{W(\mathcal{C}_2, F)} \leq \prod_{(s,s')\in F} \frac{\mathcal{C}_1(s, s')}{\mathcal{C}_2(s, s')}.$$

Since there are $|F| \leq M$ edges in the product, we have

$$\frac{W(\mathcal{C}_1, F)}{W(\mathcal{C}_2, F)} \leq (1 + \frac{1}{10M})^M \leq 1.2.$$

Similarly, for the other direction, we have:

$$\begin{aligned}\frac{W(\mathcal{C}_1,F)}{W(\mathcal{C}_2,F)} &\geq \prod_{(t,t')\in F}\left(\frac{\mathcal{C}_1(t,t')}{\mathcal{C}_2(t,t')}\right)\\ &\geq \left(1-\frac{1}{10M}\right)^M\\ &\geq 0.9.\end{aligned}$$

In particular, we now have that for every arborescence F of $\mathcal{C}_1, \mathcal{C}_2$:

$$\frac{W(\mathcal{C}_1,F)}{W(\mathcal{C}_2,F)} \leq 1.2 \text{ and } \frac{W(\mathcal{C}_2,F)}{W(\mathcal{C}_1,F)} \leq 1.2.$$

Now the lemma follows by applying the Markov chain tree theorem. For any state s:

$$\begin{aligned}\widehat{p}_{\mathcal{C}_1}(s) &= \frac{\sum_{F\in\mathcal{F}_s(\mathcal{C}_1)} W(\mathcal{C}_1,F)}{\sum_{F\in\mathcal{F}(\mathcal{C}_1)} W(\mathcal{C}_1,F)}\\ &\leq 1.2\frac{\sum_{F\in\mathcal{F}_s(\mathcal{C}_2)} W(\mathcal{C}_2,F)}{\sum_{F\in\mathcal{F}(\mathcal{C}_2)} W(\mathcal{C}_1,F)}\\ &\leq 1.2\cdot 1.2\frac{\sum_{F\in\mathcal{F}_s(\mathcal{C}_2)} W(\mathcal{C}_2,F)}{\sum_{F\in\mathcal{F}(\mathcal{C}_2)} W(\mathcal{C}_2,F)}\\ &\leq 1.5\widehat{p}_{\mathcal{C}_2}(s).\end{aligned}$$

□

Corollary 11. Let $\mathcal{A}$ be an M-state tester and $\mathcal{P}$ a property. Suppose that there exist functions f, g such that (1) $f \in \mathcal{P}$; (2) g is ε-far from every function in $\mathcal{P}$; and (3) $\mathcal{A}_f$ and $\mathcal{A}_g$ are $1/(10M)$-indistinguishable. Then $\mathcal{A}$ is not an $(\varepsilon, 1/3)$-tester for $\mathcal{P}$.

Proof. Let $S_{\text{Acc}} \subseteq [M]$ be the set of states with value Acc in $\mathcal{A}$. Since g is ε-far from $\mathcal{P}$, we have:

$$\widehat{p}_{\mathcal{A}_g}(S_{\text{Acc}}) = \sum_{s\in S_{\text{Acc}}} \widehat{p}_{\mathcal{A}_g}(s) \leq 1/3.$$

However, for all states s we have $\widehat{p}_{\mathcal{A}_f}(s) \leq 1.5\widehat{p}_{\mathcal{A}_g}(s)$ and thus

$$\widehat{p}_{\mathcal{A}_f}(S_{\text{Acc}}) = \sum_{s\in S_{\text{Acc}}} \widehat{p}_{\mathcal{A}_f}(s) \leq 1.5\cdot \sum_{s\in S_{\text{Acc}}} \widehat{p}_{\mathcal{A}_g}(s) \leq 1/2.$$

Thus $\mathcal{A}$ can't be a tester for $\mathcal{P}$ as that would require $\widehat{p}_{\mathcal{A}_f}(S_{\text{Acc}}) \geq 2/3$. □

A.3 Normal form

In this section, we prove that we can assume, at the cost of one bit of memory, that a tester $\mathcal{A}$ is in normal form:

Lemma 13. Let $\mathcal{A}$ be an (ε, δ)-tester for a property $\mathcal{P}$ using M states. Then there exists an (ε, δ)-tester $\mathcal{A}'$ in normal form using $2M$ states.

Proof. Given $\mathcal{A}$, we create a new algorithm $\mathcal{A}'$ as follows: for each state $s \in [M]$, we add two states s_0, s_1 to $\mathcal{A}'$. If s is an Acc state, then so are s_0, s_1; otherwise, s_0, s_1 are Rej states. If the start state of $\mathcal{A}$ was $s^\star$, we make $s_0^\star$ the start state of $\mathcal{A}'$. Hence $\mathcal{A}'$ has $2M$ states.

Our goal is now for the transition probabilities δ' in $\mathcal{A}'$ to mimic the transitions of $\mathcal{A}$ except that we also store the bit $f(x)$ for the last query made. To achieve this, we set the query distributions $\mathcal{Q}'_{s_0}$ and $\mathcal{Q}'_{s_1}$ at states s_0, s_1 in $\mathcal{A}'$ to be the same as $\mathcal{Q}_s$ (the query distribution at s in $\mathcal{A}$). For $s, s' \in [M]$, $x \in \{0,1\}^n$, and $i, j, b \in \{0,1\}$, define

$$\mathcal{M}'_{s_i}(x, b, s'_j) := \begin{cases} \mathcal{M}_s(x, b, s') & \text{if } j = b, \\ 0 & \text{if } j \neq b. \end{cases}$$

The above guarantees that for all $s, s' \in [M]$, $i \in \{0,1\}$, and $x \in \{0,1\}^n$, we have

$$\delta'_{s'_i, s_0}(x, 0) = \delta_{s', s}(x, 0) \quad \text{and} \quad \delta'_{s'_i, s_0}(x, 1) = 0,$$

and similarly

$$\delta'_{s'_i, s_1}(x, 1) = \delta_{s', s}(x, 1) \quad \text{and} \quad \delta'_{s'_i, s_1}(x, 0) = 0.$$

From the above construction, it's easy to see that for any function f,

$$\mathbf{Pr}[\mathcal{A}_f \text{ is at state } s \text{ at time } t] = \sum_{i \in \{0,1\}} \mathbf{Pr}[\mathcal{A}'_f \text{ is at state } s_i \text{ at time } t].$$

Let $S'_{\mathrm{Acc}} := \{s_i : s \in S_{\mathrm{Acc}},\ i \in \{0,1\}\}$, where S_{Acc} is the set of accepting states of $\mathcal{A}$. Since s_0, s_1 have the same label as s, we have

$$\widehat{p}_{\mathcal{A}_f}(S_{\mathrm{Acc}}) = \widehat{p}_{\mathcal{A}'_f}(S'_{\mathrm{Acc}}).$$

Thus the long-term probability of being in an accept state under oracle access to f is the same for $\mathcal{A}$ and $\mathcal{A}'$. Hence, $\mathcal{A}'$ is also an (ε, δ)-tester for $\mathcal{P}$. □

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il