# A canonical form for testing Boolean function properties

Dana Dachman-Soled*
Columbia University
dg2342@columbia.edu

Rocco A. Servedio†
Columbia University
rocco@cs.columbia.edu

April 13, 2011

**Abstract**

In a well-known result on graph property testing, [GT03] showed that every testable graph property has a "canonical" testing algorithm in which a set of vertices is selected uniformly at random and the edges queried are the complete graph over the selected vertices. In this paper we define a similar-in-spirit canonical form for Boolean function testing algorithms, and show that under some mild conditions on the function class and testing algorithm, property testers for Boolean functions can be transformed into this canonical form.

We establish two main results. The first shows, roughly speaking, that every "nice" family of Boolean functions that has low noise sensitivity and is testable by an "independent tester," has a canonical testing algorithm. The second result is similar but holds instead for families of Boolean functions that are closed under ID-negative minors. Taken together, these two results cover almost all of the constant-query Boolean function testing algorithms that we know of in the literature, and show that all of these testing algorithms can be automatically converted into a canonical form.

*Supported in part by an FFSEAS Presidential Fellowship.
†Supported by NSF grants CCF-0347282, CCF-0523664 and CNS-0716245, and by DARPA award HR0011-08-1-0069.

# 1 Introduction

Over more than a decade property testing has emerged as a exciting and intensively studied area of theoretical computer science, with close connections to diverse topics such as sublinear-time algorithms and probabilistically checkable proofs in complexity theory. As the field has matured several distinct strands of research have emerged corresponding to different types of objects to be tested: graphs, Boolean functions, error-correcting codes, probability distributions, and so on. Over the years each of these sub-areas has developed its own body of standard tools and proof techniques, and the results that have been obtained have different flavors across the different areas. As one example, in graph property testing powerful and general characterizations [AFNS06, AT08, AS05a, AS05b] have been given for what properties are constant-query testable with a number of queries depending only on the error parameter $\epsilon$ but not on the number of vertices $n$. In Boolean function testing, on the other hand, many specific properties have been investigated (see e.g. [BLR93, BCH$^+$96, BGS98, PRS02, AKK$^+$03], [Sam07, FKR$^+$04, Bla08, Bla09, MORS10]) but no general characterizations have been given of what makes a property of Boolean functions constant-query testable. Given this state of affairs, a natural goal is to obtain a more unified view of property testing by uncovering deeper underlying similarities between general results on testing different kinds of objects. This high-level goal provides the impetus behind the current work.

The aim of this paper is to obtain "canonical" testers for testable Boolean function properties, similar to the canonical testers that have been established for testable graph properties by Goldreich and Trevisan [GT03].[1] Specialized to properties that are testable with a constant number of queries independent of $n$, the [GT03] result is essentially as follows: Let $\mathcal{P}$ be any graph property[2] that has a $q(\epsilon)$-query testing algorithm, independent of the number of vertices $n$. The [GT03] result states that property $\mathcal{P}$ is efficiently testable by an algorithm that follows a simple prescribed "canonical form:" it draws $q(\epsilon)$ vertices independently at random, queries all $\binom{q(\epsilon)}{2}$ edges between those vertices, does some deterministic computation on the $q(\epsilon)$-node graph thus obtained, and outputs "accept" or "reject."

Our work addresses the following natural question: is there a similar "canonical form" for Boolean function property testing algorithms? Such a result would presumably say that any property of Boolean functions that is constant-query testable is in fact constant-query testable by an algorithm that works roughly as follows: it tosses some fair coins, exhaustively queries the function on "all inputs defined by those coin tosses" (in some suitable sense), does some deterministic computation on the resulting query-response pairs, and outputs "accept" or "reject." (Note that in this first investigation we consider only constant-query testable properties; indeed, the number of queries our canonical tester makes will be doubly exponential in the number of queries made by the original tester.) We elaborate below in Section 1.1, where we give a precise definition of a "canonical Boolean function testing algorithm". But first it is useful to explain how we may view any Boolean function property testing algorithm as a collection of probability distributions, as a prelude to explaining our notion of a canonical tester for Boolean function properties.

**Viewing a testing algorithm as a collection of distributions.** Let $\mathcal{P}$ be any class of Boolean functions that has a $q(\epsilon)$-query testing algorithm $A$, with query complexity independent of the number of input variables $n$. We may assume without loss of generality that $A$ is nonadaptive, since if it is adaptive we can convert it to a nonadaptive algorithm in a standard way (this incurs an exponential penalty in the query complexity but it remains independent of $n$).

Since $A$ is nonadaptive, it first generates its entire sequence of $q(\epsilon)$ query strings and then queries them and performs some computation on the results. We may view the first (query generation) stage of

[1] We note that P. Valiant [Val08] has given a "Canonical Tester" for a wide class of properties of discrete probability distributions. Our setting of testing Boolean functions is much closer to graph property testing (in both scenarios the tester actively chooses inputs and queries them) than it is to testing probability distributions (where the tester passively receives independent draws from the distribution).

[2] Recall that by definition, a graph property is closed under relabeling vertices.

the algorithm as proceeding in the following way: The first query string $x^1$ is drawn from a probability distribution $\mathcal{D}_\emptyset$ (which may be arbitrary) over $\{0,1\}^n$. The outcome $x^1 \in \{0,1\}^n$ of this draw determines a probability distribution $\mathcal{D}_{x^1}$ over $\{0,1\}^n$ from which the second query string $x^2$ is drawn. The outcomes $x^1, x^2$ of the first two draws determines a distribution $\mathcal{D}_{x^1,x^2}$ from which the third query string $x^3$ is drawn, and so on. (Note that while later query strings do not depend on the *answers* to earlier queries, they may depend on the *outcome of the randomness* that was used to construct earlier queries.) In the second stage, once the $q(\epsilon)$ strings $x^1, \ldots, x^{q(\epsilon)}$ have been generated, the algorithm makes its queries on those strings and gets a response bit $f(x^i)$ for each query string. It then performs a computation on the $q(\epsilon)$ query-response pairs, and outputs the result ("accept/reject") of that computation. This computation may *a priori* be randomized, but a straightforward argument (given in Section 4.2.3 of [GT03]) shows that without loss of generality it may be assumed to be deterministic.

Thus (ignoring for the moment the second "deterministic computation" stage that is performed once all the queries have been made), any nonadaptive testing algorithm that makes $q(\epsilon)$ queries corresponds to the collection of all distributions $\mathcal{D}_{x^1,\ldots,x^t}$ described above, where $t$ ranges from 1 to $q(\epsilon)$ and each $x^i$ ranges over all possible $n$-bit strings. This collection is somewhat complicated and cumbersome to reason about; different testing algorithms may correspond to different collections of probability distributions. Is there a simpler "canonical form" for the query generation stage of every Boolean function testing algorithm?

## 1.1 A canonical form for Boolean function testing algorithms

In the [GT03] result, intuitively there is only one type of distribution over queries for all testing algorithms (and this distribution is very simple) – all of the difference between two $q(\epsilon)$-query testing algorithms comes from the deterministic computation they do once all the query-answer pairs have been obtained. We would like an analogous result for testing Boolean functions, which similarly involves only one kind of (simple) distribution over queries, and where the difference between different testers comes from the deterministic computation that is done once all query-answer pairs are in hand.

Motivated by these considerations, we consider the following canonical form for a testing algorithm (we make this precise in Section 3):

- **First stage (query generation):** Let $z^1, \ldots, z^k$ be independent and uniform random strings from $\{0,1\}^n$. This defines a natural partition of $[n]$ into $2^k$ blocks, which are expected to be of approximately equal size: an element $i \in [n]$ lies in block $B_{(b_1,\ldots,b_k)}$ if $(z^1_i, \ldots, z^k_i) = (b_1, \ldots, b_k)$, i.e. in each query string $z^j$, the $i$-th bit is set to $b_i$.
- We say that a string $x = x_1 \ldots x_n \in \{0,1\}^n$ "respects the partition" if within each block $B_b$, either all variables $x_i$ are set to 0 or all are set to 1. There are $2^{2^k}$ strings in $\{0,1\}^n$ that respect the partition; these are the $2^{2^k}$ queries that a canonical tester makes.
- **Second stage:** With these $2^{2^k}$ query-answer pairs in hand, the algorithm does some (deterministic) computation and outputs "accept" or "reject."

We view this canonical form for Boolean function testers as both simple and natural.

Some examples of known testers that can easily be converted to canonical form as described above include the tester of [BLR93] for $GF(2)$ linear functions and the tester of [AKK$^+$03] for degree $k$ polynomials over $GF(2)$. Let us consider the [AKK$^+$03] tester and see how to convert it to canonical form. The [AKK$^+$03] tester works by choosing $k+1$ strings $z^1, \ldots z^{k+1} \in \{0,1\}^n$ uniformly at random and then querying all points in the induced subspace. If a string $x$ is in the induced subspace of $z^1, \ldots, z^{k+1}$ then it must also "respect the partition" induced by $z^1, \ldots, z^{k+1}$. So to convert the [AKK$^+$03] tester to our canonical form, all we have to do is ask some more queries.

A natural first hope is to generalize the above examples and show that *every* Boolean function property that is testable using constantly many queries has a "canonical form" constant-query testing algorithm of the above sort. However, E. Blais [Bla10] has observed that there is a simple property that is testable with $O(1/\epsilon)$ queries but does not have a constant-query canonical tester of the above sort: this is the property of being a symmetric Boolean function. Let SYM be the set of all symmetric Boolean functions (i.e., all functions where $f(x)$ is determined by $|x|$, the Hamming weight of $x$). SYM can be tested with a constant number of queries with the following algorithm:

- Pick $O(1/\epsilon)$ pairs of points $(x^i, y^i) \in \{0,1\}^n \times \{0,1\}^n$ by choosing $x$ uniformly at random from $\{0,1\}^n$ then choosing $y$ uniformly at random from all inputs with the same weight as $x$.
- Check that for each pair $f(x^i) = f(y^i)$. Accept if this holds, and otherwise reject.

It is clear that if $f \in \mathrm{SYM}$ then the above test accepts with probability 1. On the other hand, for any $f$ that is $\epsilon$-far from SYM, with probability at least $\epsilon$ the string $x^i$ is one of the "bad" inputs that has the minority output in its level, and with probability at least $1/2$ the string $y^i$ is one of the inputs with the majority output for the same level. So with probability at least $\epsilon/2$, $(x^i, y^i)$ is a witness to the fact that $f$ is not symmetric, and $O(1/\epsilon)$ queries are sufficient to reject $f$ with probability at least $2/3$.

To show that SYM cannot be tested by a constant-query canonical tester, it suffices to show that for $k = o(\log \log n)$, with high probability each of the $2^{2^k} = n^{o(1)}$ queries generated by the tester has different Hamming weight. This can be established by a straightforward but somewhat tedious argument which we omit here (the main ingredients are the observation that each query string $x$ generated by the canonical tester has Hamming weight distributed according to a binomial distribution $B(n, \frac{j}{2^k})$ for some integer $j$, together with standard anti-concentration bounds on binomial distributions with sufficiently large variance).

The example above shows that, unlike the graph testing setting, it is not the case that *every* constant-query Boolean function tester can be "canonicalized." So in order to obtain meaningful results on "canonicalizing" Boolean function testers, one must restrict the types of properties and/or testers that are considered; this is precisely what we do in our results, as explained below.

## 1.2 Our results

Our main results are that certain "nice" testing algorithms, for certain "nice" types of Boolean function properties, can automatically be converted into the above-described canonical form. Roughly speaking, the testing algorithms we can handle are ones for which every distribution $\mathcal{D}_{x^1,\ldots,x^t}$ in the query generation phase is a product of $n$ Bernoulli distributions over the $n$ coordinates (with some slight additional technical restrictions that we describe later). This is a restricted class of algorithms, but it includes many different testing algorithms that have been proposed and analyzed in the Boolean function property testing literature. We call such testing algorithms "Independent testers" (see Section 2.1 for a precise definition), and we give two results showing that independent testers for certain types of properties can be "canonicalized."

Our first result applies to classes $C$ that are closed under negating variables and contain only functions with low *noise sensitivity*. We say that such a class $C$ is *closed under* Noisy-Neg *minors* (see Definition 1). For such classes $C$ we show the following:

**Theorem 1 (Informal)** *If $C$ is closed under* Noisy-Neg *minors and there exists a (two-sided) independent tester for $C$, then there exists a (two-sided) canonical tester for $C$.*

Our second result applies to classes $C$ that are closed under identification of variables, negation of variables, and adding or removing irrelevant variables. Following [HR05], we say that such a class is *closed under* ID-Neg *minors* (see Definition 2). For such classes $C$ we show the following:

**Theorem 2 (Informal)** *If $C$ is closed under* ID-Neg *minors and there exists a one-sided independent tester for $C$, then there exists a one-sided canonical tester for $C$.*

As we describe in Section A, these two results allow us to give "canonical" versions of many different Boolean function property testing algorithms that have appeared in the literature.

### 1.3 Our approach

Developing a canonical tester for Boolean function properties seems to be significantly more challenging than for graph properties. The high-level idea behind the [GT03] graph testing canonicalization result is that if $k$ edges have been queried so far, then all "untouched" vertices (that are not adjacent to any of the $k$ queried edges) are equally good candidates for the next vertex to be involved in the query set. For Boolean function testing the situation is more complicated because of the structure imposed by the Boolean hypercube; for example, if the two strings $0^n$ and $1^n$ have been queried so far, then it is clearly not the case that all possible 3rd query strings are "created equal" in relation to these first two query strings.

A natural first effort to canonicalize a Boolean function property tester is to design a canonical tester that makes its queries in the first stage, and then simply directly uses those queries to "internally" simulate a run of the independent tester in its second stage. Ideally, in such an "internal" simulation, each time the original independent tester makes a query the canonical tester would use a query-response pair obtained in its first stage that corresponds reasonably well to the string queried by the independent tester. However, this naive approach does not seem to suffice, since an independent tester can easily make queries which do not correspond well to any query made by the canonical tester. As a simple example, the first query of an independent tester could independently set each variable $x_i$ to 1 with probability $1/3$. The number of 1s in the first query of this independent tester is distributed as a draw from the binomial distribution $B(n, 1/3)$, but the number of 1s in any query made by a $q$-query canonical tester is distributed as a draw from the binomial distribution $B(n, p)$, where $p$ is of the form (integer)$/2^q$. If $q$ is a constant independent of $n$, these two distributions have variation distance nearly 1.

The high-level idea of both our constructions is that instead of trying to approximately simulate an execution of the independent tester on the $n$-variable function $f$ (which it cannot do), the canonical tester *perfectly* simulates an execution of the independent tester on a different function $f'$ over $n'$ relevant variables. Since this simulation is perfect, the canonical tester successfully tests whether $f'$ has property $C$. For the case of Noisy-Neg minors the analysis shows that w.h.p. the independent tester's view looks the same whether the target function is $f'$ or $f$. Therefore, a "good" answer for $f'$ must also be a good answer for $f$. For the case of ID-Neg minors, the analysis shows that because of the way $f'$ is determined, we have that (1) if $f$ belongs to $C$ then so does $f'$; and (2) if $f$ is far from $C$, then $f'$ is at least slightly far from $C$. Along with the fact that the canonical tester tests $f'$ successfully, these conditions imply that the canonical tester tests $f$ successfully.

## 2 Preliminaries

A *Boolean function property* is simply a class of Boolean functions. Throughout the paper we write $\mathcal{F}_n$ to denote the class of all $2^{2^n}$ Boolean functions mapping $\{0,1\}^n$ to $\{0,1\}$. We write $C_n$ to denote a class of $n$-variable Boolean functions, i.e. functions from $\{0,1\}^n$ to $\{0,1\}$.

We adopt all the standard definitions of Boolean function property testing (see e.g. [PRS02, FKR$^+$04, MORS10]) and do not repeat them here because of space limitations. As mentioned in the Introduction, we may view any nonadaptive testing algorithm $T$ as consisting of two phases: an initial "query generation phase" $T_1$ in which the query strings are selected (at the end of this phase the queries are performed), and a subsequent "computation" phase $T_2$ in which some computation is performed on the query-answer pairs and

the algorithm either accepts or rejects. Throughout the paper we will describe and analyze testing algorithms in these terms.

**The classes we consider.** Our first main result deals with classes of Boolean functions that are closed under Noisy-Neg minors; we give the relevant definitions below.

**Definition 1 (Noise Sensitivity of $f$)** *Let $f : \{0,1\}^n \to \{0,1\}$, let $\epsilon \in [0, 1/2]$, and let $(x, y)$ be a pair of $(1-2\epsilon)$-correlated random inputs (i.e. $x$ is uniform from $\{0,1\}^n$ and $y$ is formed by independently setting each $y_i$ to equal $x_i$ with probability $1-2\epsilon$, and to be uniform random otherwise). The* noise sensitivity of $f$ at noise rate $\epsilon$ *is defined to be $NS_\epsilon(f) := \Pr[f(x) \neq f(y)]$.*

**(Noise Sensitivity of a class $C$)** *Let $C = \cup_{n\geq 1} C_n$ be a class of Boolean functions. We define $NS_\epsilon(C) := \max_n \max_{f\in C_n}\{NS_\epsilon(f)\}$, the* noise sensitivity of $C$ at noise rate $\epsilon$, *to be the maximum noise sensitivity of any $f \in C$.*

**($C$ is closed under Noisy-Neg minors)** *Let $C = \cup_{n\geq 1} C_n$ be a class of Boolean functions. We say that $C$ is* closed under Noisy-Neg Minors *if $C$ is closed under negating input variables and there is a function $g(\epsilon)$ (not depending on $n$) which is such that $\lim_{\epsilon\to 0^+} g(\epsilon) = 0$ and $NS_\epsilon(C) \leq g(\epsilon)$.*

Our second main result deals with classes $C$ that are *closed under ID-Neg Minors*.

**Definition 2 (ID-Neg Minors)** *Let $f \in \mathcal{F}_n$ and let $f' \in \mathcal{F}_{n'}$. We say that $f'$ is an* ID-Neg Minor *of $f$ if $f'$ can be produced from $f$ by a (possibly empty) sequence of the following operations: (i) Adding/Removing irrelevant variables (recall that variable $x_i$ is irrelevant if there is no input string where flipping $x_i$ changes the value of $f$); (ii) Identifying input variables (e.g. the function $f(x_1, x_1, x_3)$ is obtained by identifying variable $x_2$ with $x_1$); and (iii) Negating input variables.*

**($C$ is closed under ID-Neg Minors)** *Let $C = \cup_{n\geq 1} C_n$ be a class of Boolean functions, let $f \in \mathcal{F}_n$, and let $f' \in \mathcal{F}_{n'}$. We say that $C$ is* closed under ID-Neg Minors *if the following holds: If $f \in C_n$ and $f'$ is an ID-Neg Minor of $f$, then $f' \in C$.*

The class of $GF(2)$ degree-$d$ polynomials is an example of a class closed under ID-Neg minors. The class of halfspaces is an example of a class closed under Noisy-Neg minors. For more examples and discussion, see Section A.

We close this preliminaries section with two definitions that will be useful:

**Definition 3** *Let $f$ be a function in $\mathcal{F}_n$ and let $F_+, F_-$ be two disjoint subsets of $[n]$. We define $\mathrm{Noisy}(f, F_+, F_-) \in \mathcal{F}_n$ to be the function $\mathrm{Noisy}(f, F_+, F_-)(x_1, \ldots, x_n) = f(t_1, \ldots, t_n)$, where $t_i := 1$ if $i \in F_+$, $t_i := 0$ if $i \in F_-$; and $t_i := x_i$ otherwise.*

Intuitively, given a target function $f$, our canonical tester for classes $C$ that are closed under Noisy-Neg minors will choose $F_+, F_-$ according to some distribution (defined later) and will instead test the target function $f' = \text{Noisy-Neg}(f, F_+, F_-)$.

**Definition 4** *Let $f$ be a function in $\mathcal{F}_n$, $F_+$ and $F_-$ be two disjoint subsets of $[n]$, and $id$ be an element of $F_+$. For $n' = n - |F_+| - |F_-| + 1$, we define the function $\text{ID-Neg}(f, F_+, F_-, id) \in \mathcal{F}_{n'}$ to be the function $\text{ID-Neg}(f, F_+, F_-, id)(x_1, \ldots, x_{n'}) = f(t_1, \ldots, t_n)$, where $t_i := x_{id}$ if $i \in F_+$; $t_i := \overline{x}_{id}$ if $i \in F_-$; and $t_i := x_i$ otherwise.*

Similarly to the case above, given a target function $f$ our canonical tester for classes $C$ that are closed under ID-Neg minors will choose $F_+, F_-, id$ according to some distribution (defined later) and will instead test the target function $f' = \text{ID-Neg}(f, F_+, F_-, id)$.

### 2.1 The testing algorithms we can canonicalize: Independent Testers

**Definition 5** *A $q(\epsilon)$-query* independent tester for class $C$ *is a probabilistic oracle machine $T = (T_1, T_2)$ which takes as input a distance parameter $\epsilon$ and is given access to a black-box oracle for an arbitrary function $f : \{0,1\}^n \to \{0,1\}$.*

**(First Stage)** *The* query generation algorithm $T_1$ *chooses $q(\epsilon)$ query strings in the following way: To choose the $i$-th string, the algorithm partitions the set $[n]$ into $2^{i-1}$ blocks. The block $B_{b_1,\ldots,b_{i-1}}$ contains those indices that were set to $b_j$ in the $j$th query string $x^j$ for all $j = 1, \ldots, i-1$. For each block $B_{b_1,\ldots,b_{i-1}}$, for each $m \in B_{b_1,\ldots,b_{i-1}}$, the algorithm sets $x^i_m$ to 1 with probability $p_{b_1,\ldots,b_i}$ and to 0 with probability $1 - p_{b_1,\ldots,b_i}$. The resulting string $x^i$ is the $i$-th query string. After choosing all the strings, $T_1$ queries all $q(\epsilon)$ strings $x^1, \ldots, x^{q(\epsilon)}$ and gets back responses $f(x^1), \ldots, f(x^{q(\epsilon)})$.*

**(Second Stage)** *The* computation stage $T_2$ *gets as input the $q(\epsilon)$ query-answer pairs $(x_1, f(x_1)), \ldots, (x_{q(\epsilon)}, f(x_{q(\epsilon)}))$, does some deterministic computation on this input, and outputs either "accept" or "reject."*

*In an independent tester the query generation algorithm $T_1$ must satisfy the following conditions:*

- *For each string $b = (b_1, \ldots, b_t)$ the probability $p_b = p_b(\epsilon)$ is a value $0 \le p_b \le 1$ (which may depend on $\epsilon$ but is independent of $n$).*
- *For each $t$, the $2^t$ values $p_{b_1,\ldots,b_t}$ (as $b$ ranges over $\{0,1\}^t$) are all rational numbers, and (over all $t$) the denominator of each of these rational numbers is at most $c = c(\epsilon)$ ($c$ may depend on $\epsilon$ but is independent of $n$). We say that $c(\epsilon)$ is the* granularity *of the independent tester $T$.*

If $T$ is a one-sided tester then for any $f : \{0,1\}^n \to \{0,1\}$, if $f$ belongs to $C$ then $\Pr[T^f = \text{"accept"}] = 1$, and if $f$ is $\epsilon$-far from $C$ then $\Pr[T^f = \text{"reject"}] \ge r(\epsilon)$, where $r(\epsilon) > 0$ is a positive-valued function of $\epsilon$ only. We say that $r(\epsilon)$ is the *rejection parameter* of the tester.

If $T$ is a two-sided tester then for any $f : \{0,1\}^n \to \{0,1\}$, if $f$ belongs to $C$ then $\Pr[T^f = \text{"accept"}] = 1 - a(\epsilon)$, and if $f$ is $\epsilon$-far from $C$ then $\Pr[T^f = \text{"reject"}] \ge r(\epsilon)$ where $a$ and $r$ are functions of $\epsilon$ only and for $0 < \epsilon < 1/2$, $a(\epsilon) < r(\epsilon)$. We say that $a(\epsilon)$ and $r(\epsilon)$ are the *acceptance* and *rejection parameters* of the tester respectively.

Given an independent tester $T$ as described above, we let $\mathrm{Prod}(\epsilon)$ denote the product of the denominators of all probabilities $p_{b_1,\ldots,b_t}(\epsilon)$ where $t$ ranges over all possible values $1, 2, \ldots, q(\epsilon)$ and $b = (b_1, \ldots, b_t)$ ranges over all $t$-bit strings. If the tester $T$ is $c(\epsilon)$-granular, it is easy to see that $\mathrm{Prod}(\epsilon)$ is at most $c(\epsilon)^{2^{q(\epsilon)+1}}$. It is clear that each subset $B_{b_1,\ldots,b_t}$ of $[n]$ described above has size binomially distributed according to $B(n, \ell/\mathrm{Prod}(\epsilon))$ for some integer $\ell$.

## 3 A canonical form for testers, and our main results

Before stating our main results precisely, we give a precise description of the canonical form mentioned in Section 1.1.

**Definition 6** *Let $q' : [0,1) \to \mathbb{N}$. A* $q'$-canonical tester for class $C$ *is a probabilistic oracle machine $T = (T_1, T_2)$ which takes as input a distance parameter $\epsilon$ and is given access to a black-box oracle for an arbitrary function $f : \{0,1\}^n \to \{0,1\}$, and performs as follows.*

*Given input parameter $\epsilon$, the query generation algorithm $T_1$ works as follows.*

1. *$z^1, \ldots, z^{q'(\epsilon)}$ are selected to be independent uniformly random $n$-bit strings. These strings define a partition $\mathcal{B}$ of $[n]$ into $2^{q'(\epsilon)}$ blocks: an element $i \in [n]$ lies in block $B_{b_1,\ldots,b_{q'(\epsilon)}}$ if the $i$-th bit of string $z^j$ equals $b_j$ for all $j = 1, \ldots, q'(\epsilon)$.*

2. *Let $Q^{\mathcal{B}} \subseteq \{0,1\}^n$ be the set of all strings $x$ such that the following condition holds: $\forall i,j \in [n]$, if $i$ and $j$ are in the same partition subset $B_{b_1,\ldots,b_{q'(\epsilon)}} \in \mathcal{B}$ then $x_i = x_j$.*

3. *Using the oracle for $f$, $T_1$ queries all $2^{2^{q'(\epsilon)}}$ strings $x \in Q^{\mathcal{B}}$.*

*The computation stage $T_2$ gets as input the $2^{2^{q'(\epsilon)}}$ query-answer pairs $[(x, f(x))]_{x \in Q^{\mathcal{B}}}$, does some deterministic computation, and outputs either "accept" or "reject."*

The success criteria for one-sided (two-sided, respectively) canonical testers are entirely similar to the criteria defined above for independent testers. We note that a $q'$-canonical tester makes $2^{2^{q'(\epsilon)}}$ queries when run with input parameter $\epsilon$.

### 3.1 Main results

As our main results, we show that (i) any class that is closed under Noisy-Neg minors and is constant-query testable by a (two-sided) independent tester is also constant-query testable by a (two-sided) canonical tester; and (ii) any class that is closed under ID-Neg minors and is constant-query testable by a one-sided independent tester is also constant-query testable by a one-sided canonical tester. More precisely, we prove the following:

**Theorem 3** *Let $C$ be any class of functions closed under Noisy-Neg Minors and let $g(\epsilon)$ be as in Definition 1. Let $T$ be a $q(\epsilon)$-query independent tester for property $C$ with acceptance and rejection parameters $a(\epsilon)$, $r(\epsilon)$. Let $q_2'(\epsilon)$ be the smallest integer value that satisfies the following bound:*

$$NS_{\frac{\mathrm{Prod}(\epsilon)}{2} \cdot \frac{1}{2^{q_2'(\epsilon)}}}(C) \leq \frac{r(\epsilon) - a(\epsilon)}{16q(\epsilon)}.$$

*Let $\eta' = \frac{2^{q_2'(\epsilon)} \bmod \mathrm{Prod}(\epsilon)}{2^{q_2'(\epsilon)}}$ where $\mathrm{Prod}$ is as defined in Section 2.1and let $q_1'(\epsilon) = \left\lceil \frac{32}{NS_{\eta'}(C)} \ln \frac{8}{r(\epsilon)-a(\epsilon)} \right\rceil$. Let $q'(\epsilon) = q_2'(\epsilon) \cdot (q_1'(\epsilon) + 1)$. Then there is a $q'$-canonical tester $\mathrm{Canon}(T)$ for $C$ with acceptance and rejection parameters $a'(\epsilon), r'(\epsilon)$, where $a'(\epsilon) = \frac{3}{4}a(\epsilon) + \frac{1}{4}r(\epsilon)$, and $r'(\epsilon) = \frac{1}{4}a(\epsilon) + \frac{3}{4}r(\epsilon)$.*

**Theorem 4** *Let $C$ be any class of functions closed under $\mathrm{ID\text{-}Neg}$ Minors. Let $T$ be a one-sided independent tester for property $C$ that has query complexity $q(\epsilon)$, granularity $c(\epsilon)$, and rejection parameter $r(\epsilon)$. Let $\epsilon_1 = \frac{r(\epsilon)}{4q(\epsilon)}$ and let $q'(\epsilon)$ be a defined as $q'(\epsilon) = \lceil \log(\mathrm{Prod}(\epsilon) \cdot \mathrm{Prod}(\epsilon_1)) \rceil$ where $\mathrm{Prod}$ is as described in Section 2.1. Then there is a one-sided $q'$-canonical tester $\mathrm{Canon}(T)$ for property $C$ which, on input parameter $\epsilon$, has rejection parameter $(\frac{3r(\epsilon)/4}{1-r(\epsilon)/4}) \cdot r(\epsilon_1)$.*

Throughout the rest of the paper whenever we write "$T$" or "$C$" without any other specification, we are referring to the tester and property from Theorem 3 or Theorem 4 (which one will be clear from context).

**Applications of our main results.** We can canonicalize known testers for many constant-query testable Boolean function classes found in the literature by applying either our first or second result. Because of space constraints we describe these applications in Appendix A.

## 4 Overview of the proofs of Theorems 3 and 4

In this section we give a high-level explanation of our arguments and of the constructions of our canonical testers. Full details and complete proofs of Theorems 3 and 4 are given in the Appendix.

We first note that an execution of the Independent Tester $T = (T_1, T_2)$ (see Definition 5) with input parameter $\epsilon$ creates a $2^{q(\epsilon)}$-way partition of the $n$ variables by independently assigning each variable to a randomly chosen subset in the partition with the appropriate probability (of course these probabilities need not all be equal). All queries made by the independent tester then respect this partition.

Consider the following first attempt at constructing a canonical tester $\mathrm{Canon}(T) = (\mathrm{Canon}(T)_1, \mathrm{Canon}(T)_2)$ from an independent tester $T$. In the first stage, $\mathrm{Canon}(T)_1$ partitions the $n$ variables into $2^{q'}$ subsets of expected size $n/2^{q'}$, as specified in Definition 6, and makes all corresponding queries; it then passes both the queries and the responses to the second stage, $\mathrm{Canon}(T)_2$. The value $q'$ will be such that $2^{q'}$ equals $\mathrm{Prod}(\epsilon) \cdot k + rem$, where $0 \leq rem < \mathrm{Prod}(\epsilon)$, and $k$ is a positive integer. In the second stage, $\mathrm{Canon}(T)_2$ chooses the first $\mathrm{Prod}(\epsilon) \cdot k$ subsets of $\mathrm{Canon}(T)_1$'s partition (let us say these subsets collectively contain $n'$ variables) and ignores the variables in the last $rem$ subsets. For the $n'$ variables contained in these first $\mathrm{Prod}(\epsilon) \cdot k$ subsets, $\mathrm{Canon}(T)_2$ can perfectly simulate a partition created by an execution of the independent tester $T$ on these $n'$ variables with parameter $\epsilon$, by "coalescing" these $\mathrm{Prod}(\epsilon) \cdot k$ subsets into $2^{q'(\epsilon)}$ subsets of the appropriate expected sizes. (To create a subset whose size is binomially distributed according to $B(n', \ell/\mathrm{Prod}(\epsilon))$, $\mathrm{Canon}(T)_2$ "coalesces" a collection of $k\ell$ of the $\mathrm{Prod}(\epsilon)$ subsets.) To simulate each of the $q = q(\epsilon)$ queries that $T$ makes, $\mathrm{Canon}(T)_2$ sets the $n'$ variables as $T_1$ would set them given this partition.

Obviously, the problem with the above simulation is how to set the extra variables in the remaining $rem$ subsets in each of the $q$ queries. The $n'$ variables described above are faithfully simulating the distribution over query strings that $T$ would make if it were run on an $n'$-variable function with input parameter $\epsilon$, but of course the actual queries that $\mathrm{Canon}(T)$ makes have the additional $rem$ variables, and the corresponding responses are according to the $n$-variable function $f$. Thus, we have no guarantee that $T_2$ will answer correctly w.h.p. when executed on the query-response strings generated by the simulator $\mathrm{Canon}(T)_2$ as described above. Nevertheless, the simulation described above is a good starting point for our actual construction.

The underlying idea of our canonical testers is that instead of (imperfectly) simulating an execution of the independent tester on the actual $n$-variable target function $f$, the canonical tester *perfectly* simulates an execution of the independent tester on a related function $f'$. Our analysis shows that due to the special properties of the independent tester and of the classes we consider, the response of the independent tester on target function $f'$ is also a legitimate response for $f$.

Below we describe the construction of a canonical tester for two different types of independent testers and classes. The first construction shows how to transform $T$, where $T$ is a two-sided independent tester for a class $C$ that is closed under Noisy-Neg Minors, into $\mathrm{Canon}(T)$, a two-sided canonical tester for class $C$. The second construction shows how to transform $T$, where $T$ is a one-sided independent tester for a class $C$ closed under ID-Neg minors, into $\mathrm{Canon}(T)$, a one-sided canonical tester for $C$.

### 4.1 Construction for two-sided independent testers and classes closed under $\mathrm{Noisy}$-$\mathrm{Neg}$ Minors

We first note that it is easy to construct an algorithm that approximates $NS_\eta(f)$ of a target function $f$ by non-adaptively drawing pairs of points $(x, y)$ where $x$ is chosen uniformly at random and $y$ is $1 - 2\eta$ correlated with $x$. It is also easy to see that if $\eta$ is a rational number $c_1/c_2$ where $c_2$ is a power of 2, then the distribution over queries made by such an algorithm can be simulated using a canonical tester.

For ease of understanding we view our second canonical tester as having two parts (it will be clear that these two parts can be straightforwardly combined to obtain a canonical tester that follows the template of Definition 6). The first part is an algorithm that approximates $NS_{\eta'}(f)$ and rejects any $f$ for which $NS_{\eta'}(f)$ is noticeably higher than $NS_{\eta'}(C)$ (here $\eta'$ is a parameter of the form (integer)/(power of 2) that will be specified later).

The second part of the tester simulates the partition generated by the independent tester $T$ as described at the start of Section 4. Let $F_+$ contain the variables assigned to the first $rem/2$ subsets from the $rem$ "remaining" subsets, and let $F_-$ contain the variables assigned to the last $rem/2$ of those subsets. As a thought experiment, we may imagine that the variables in $F_+ \cup F_-$ are each independently assigned to a randomly selected one of the $\mathrm{Prod}(\epsilon)$ partition subsets with the appropriate probability. In this thought experiment, we have perfectly simulated a partition generated by running $T_1$ over an $n$-variable function. We now define $f'$ based on the subsets $F_+$ and $F_-$. The function $f'$ is simply the restriction of $f$ under which all variables in $F_+$ are fixed to 1 and all variables in $F_-$ are fixed to 0.

Now, we would like $\mathrm{Canon}(T)$ to generate the $q$ query-answer pairs for $f$ that $T_1$ would make given the partition from the thought experiment described above. While $\mathrm{Canon}(T)$ cannot do this, a crucial observation is that $\mathrm{Canon}(T)$ *can* perfectly simulate $q$ query-answer pairs that $T_1$ would make given the above-described partition *where the answers are generated according to* $f'$. Moreover, our analysis will show (using the fact that $C$ is closed under negation) that we may assume w.l.o.g. that each of these $q$ queries is individually uniformly distributed over $\{0,1\}^n$.

Thus for each individual (uniform random) query string $x$, we have that $f'(x)$ is equivalent to $f(y)$ where $y$ is a random string that is $(1-2\eta')$-correlated with $x$, where $\eta' = rem/2^{q'}$. Now since $NS_{\eta'}(C)$ depends only on $\eta'$, by choosing $\eta'$ small enough (and $q'$ large enough), we have by a union bound that with high probability $f(x)$ equals $f'(x)$ for all the queries $x$ that were generated. Since this is the case, then $T_2$ must with high probability generate the same output on target function $f$ and $f'$. So since $T$ is (by hypothesis) an effective tester for $f$ it must be the case that $T$'s responses on $f'$ are also "good" for $f$.

### 4.2 Construction for one-sided independent testers and classes closed under ID-Neg minors

Our second canonical tester construction also begins by simulating a partition of the independent tester $T$ over $n'$ variables as described above. However, now we will think of the parameters as being set somewhat differently: we view the canonical tester as partitioning the $n$ variables into $2^{q'(\epsilon)}$ subsets where now $q' = q'(\epsilon)$ is such that $2^{q'}$ equals $\mathrm{Prod}(\epsilon_1) \cdot k + rem$, where $\epsilon_1 \ll \epsilon$ (more precisely $\epsilon_1 = \frac{r(\epsilon)}{4q(\epsilon)}$, though this exact expression is not important for now), $0 \leq rem < \mathrm{Prod}(\epsilon)$, and $k$ is a positive integer. The canonical tester then defines a new function $f'$ over $n'$ variables by applying an operator $\mathcal{F}^{\epsilon_1}$ to the set $X$ of $2^{2^{q'(\epsilon)}}$ query strings that $\mathrm{Canon}(T)_1$ generates; we now describe how this operator acts. Let $F_+$ contain the variables assigned to the first $rem/2$ subsets from the $rem$ "remaining" subsets, and let $F_-$ contain the variables assigned to the last $rem/2$ of the $rem$ subsets. Given $f$, the function $f'$ that is obtained by applying $\mathcal{F}^{\epsilon_1}$ to $X$ is chosen in the following way: $f'$ is the same as $f$ except that

- A variable $x_{id}$ is chosen by taking the lexicographically first element of $F_+$;
- All variables in $F_+$ are identified with $x_{id}$, and all variables in $F_-$ are identified with $\overline{x}_{id}$.

$\mathrm{Canon}(T)$ places $id$ at random in one of the remaining partition subsets and then selects the appropriate set of query strings that $T_1$ would make given the simulated partition over $n'$ variables described above, and constructs query-answer pairs for these strings in which the answers are the corresponding values of $f'$ on these strings (note that similar to the crucial observation in Section 4.1, it is indeed possible for $\mathrm{Canon}(T)$ to do this). Finally, $\mathrm{Canon}(T)$ passes these queries and responses to $T_2$ and responds as $T_2$ does.

The proof of correctness of this canonical tester proceeds in two parts. First, we show that with high probability $\mathrm{Canon}(T)$ successfully tests target function $f'$. (This is an easy consequence of the fact, mentioned above, that $\mathrm{Canon}(T)$ perfectly simulates $T$'s partition over the $n'$ variables.) Second, we show that (1) if $f \in C$ and $C$ is closed under ID-Neg minors then $f' \in C$; and (2) if $f$ is $\epsilon$-far from $C$ then w.h.p. $f'$ is $\epsilon_1$-far from $C$, where the value of $\epsilon_1$ depends only on $\epsilon$. We note that (2) does not hold in general for $f, f'$

where $f'$ is an arbitrary ID-Neg minor of $f$. However, our analysis shows that assuming that there exists a one-sided independent tester for class $C$, (2) holds for $f'$ chosen in the way described above.

**Organization of the rest of the paper.** Appendix A presents applications of our main results. In Appendix B we describe how to normalize a tester, which is a useful preliminary stage employed in both of our constructions. Then in Appendix C and Appendix D we present the construction for classes closed under $\mathrm{Noisy\text{-}Neg}$ minors and the analysis. Finally, in Appendix E and Appendix F we present the construction for classes closed under $\mathrm{ID\text{-}Neg}$ minors and the analysis.

**Conclusions.** Our work is the first attempt we know of to establish a canonical form for Boolean function property testers. Our results show that a wide range of efficient testing algorithms for many well-studied Boolean function classes can be transformed into a natural canonical form.

Building on our work, it would be nice to have more general results that do not require bounded noise sensitivity or closure under ID-Neg minors, or alternately to have examples showing that some such conditions are necessary for canonicalization. Another natural goal is to improve the quantitative bounds that we obtain.

# References


[AFNS06] N. Alon, E. Fischer, I. Newman, and A. Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. In *Proc. STOC*, 2006.

[AKK+03] N. Alon, T. Kaufman, M. Krivelevich, S. Litsyn, and D. Ron. Testing low-degree polynomials over GF(2). In *Proc. RANDOM*, pages 188–199, 2003.

[AS05a] Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. In *Proc. FOCS*, pages 429–438, 2005.

[AS05b] Noga Alon and Asaf Shapira. Every monotone graph property is testable. In *Proc. STOC*, pages 128–137, 2005.

[AT08] Tim Austin and Terry Tao. On the testability and repair of hereditary hypergraph properties. *Submitted to Random Structures and Algorithms*, 2008.

[BCH+96] M. Bellare, D. Coppersmith, J. Hastad, M. Kiwi, and M. Sudan. Linearity testing in characteristic two. *IEEE Trans. on Information Theory*, 42(6):1781–1795, 1996.

[BGS98] M. Bellare, O. Goldreich, and M. Sudan. Free bits, pcps and non-approximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.

[Bla08] Eric Blais. Improved bounds for testing juntas. In *Proc. RANDOM*, pages 317–330, 2008.

[Bla09] Eric Blais. Testing juntas nearly optimally. In *Proc. 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158, 2009.

[Bla10] Eric Blais. Personal communication. 2010.

[BLR93] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comp. Sys. Sci.*, 47:549–595, 1993. Earlier version in STOC'90.

[DLM+07] I. Diakonikolas, H. Lee, K. Matulef, K. Onak, R. Rubinfeld, R. Servedio, and A. Wan. Testing for concise representations. In *Proc. 48th Ann. Symposium on Computer Science (FOCS)*, pages 549–558, 2007.

[FKR+04] E. Fischer, G. Kindler, D. Ron, S. Safra, and A. Samorodnitsky. Testing juntas. *J. Computer & System Sciences*, 68(4):753–787, 2004.

[GOS+09] P. Gopalan, R. O'Donnell, R. Servedio, A. Shpilka, and K. Wimmer. Testing Fourier dimensionality and sparsity. In *Proc. 36th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 500–512, 2009.

[GT03] Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures and Algorithms*, 23(1):23–57, August 2003.

[HR05] Lisa Hellerstein and Vijay Raghavan. Exact learning of DNF formulas using DNF hypotheses. *Journal of Computer & System Sciences*, 70(4):435–470, 2005.

[MORS10] K. Matulef, R. O'Donnell, R. Rubinfeld, and R. Servedio. Testing halfspaces. *SIAM J. Comp.*, 39(5):2004–2047, 2010.

[PRS02] M. Parnas, D. Ron, and A. Samorodnitsky. Testing basic boolean formulae. *SIAM J. Disc. Math.*, 16:20–46, 2002.

[Sam07] A. Samorodnitsky. Low-degree tests at large distances. In *Proc.* $39^{th}$ *ACM Symposium on the Theory of Computing (STOC'07)*, pages 506–515, 2007.

[Val08] P. Valiant. *Testing Symmetric Properties of Distributions*. PhD thesis, M.I.T., 2008.

# A Applications of our main results.

In this section, we discuss some of the applications of our two main results (also see Table A). Recall that Theorem 4 applies to classes closed under ID-Neg minors and to one-sided independent testers, while Theorem 3 applies to classes closed under Noisy-Neg minors and to (two-sided) independent testers. Many natural classes of Boolean functions that have been studied in the property testing literature are closed under either ID-Neg minors or Noisy-Neg minors. Classes closed under ID-Neg minors include $J$-juntas [FKR+04, Bla08, Bla09] (for $J$ independent of $n$), $s$-term DNFs [DLM+07] (for $s$ independent of $n$), halfspaces [MORS10], $GF(2)$-linear functions (i.e. parities and their negations) [BLR93], and degree-$d$ $GF(2)$ polynomials [AKK+03]. Classes closed under Noisy-Neg minors include all the classes of Boolean functions for which testing algorithms were given in [DLM+07]; these include decision lists, size-$s$ decision trees, size-$s$ branching programs, $s$-term DNFs, size-$s$ Boolean formulas, $s$-sparse polynomials over $GF(2)$, size-$s$ Boolean circuits and functions with Fourier degree at most $s$, where throughout $s$ is independent of $n$. For each of these classes, using the fact that functions in the class are well-approximated by juntas (which is the key to the testing results of [DLM+07]) it is not difficult to verify that the class has low noise sensitivity.

We further note that most of the Boolean function property testers found in the literature can be characterized as independent testers. These include the 1-sided "size test" for testing $J$-juntas of [FKR+04], the (nonadaptive version of) the junta tester of [Bla09], the parity tester of [BLR93], the testers of [PRS02] for Boolean literals, conjunctions and $s$-term monotone DNFs, the general "testing by implicit learning" algorithm of [DLM+07] for the classes listed above, and the tester of [MORS10] for halfspaces.

There are several classes that are both closed under Noisy-Neg minors and additionally are known to have an independent tester, and so our Theorem 3 can be applied to "canonicalize" these algorithms; these include the testers of [DLM+07] and [MORS10] for the classes listed above.

Since our ID-Neg result (Theorem 4) requires independent testers that are also one-sided, the setting is more restrictive, but there are still several testers and classes found in the literature that satisfy our requirements. These include the tester for singletons of [PRS02], the $GF(2)$-linear function tester of [BLR93], the [AKK$^+$03] tester for degree-$d$ $GF(2)$ polynomials, and the one-sided junta testers given by [FKR$^+$04].

Thus, we can canonicalize known testers for many constant-query testable Boolean function classes found in the literature by applying either our first or second result. One exception comes from [GOS$^+$09]; that work gives testing algorithms for the class of Boolean functions with Fourier dimension $d$ (i.e. for "$d$-juntas of parities") and for the class of Boolean functions with $s$-sparse Fourier spectra. It is easy to see that these classes are not closed under Noisy-Neg minors, since each class contains all parity functions, and the testers of [GOS$^+$09] do not have 1-sided error. (However, we note that inspection of the testers provided in [GOS$^+$09] shows that they can be straightforwardly "canonicalized" just like the [AKK$^+$03] tester discussed in Section 1.1.)

| **Function class** | **Reference** | **1-sided** | **ID-Neg** | **Noisy-Neg** | **Theorem** |
|---|---|---|---|---|---|
| literals (dictators) | [PRS02] | YES | YES | YES | Thm. 3, Thm. 4 |
| $GF(2)$-linear functions | [BLR93] | YES | YES | NO | Thm. 4 |
| $GF(2)$-deg-$d$ functions | [AKK$^+$03] | YES | YES | NO | Thm. 4 |
| $J$-juntas | [FKR$^+$04, Bla08, Bla09] | (some) | YES | YES | Thm. 3, Thm. 4 |
| decision lists; size-$s$ decision trees; size-$s$ branching programs; $s$-term DNFs; size-$s$ Boolean formulas; $s$-sparse $GF(2)$ polynomials; size-$s$ Boolean circuits; functions with Fourier degree $d$ | [DLM$^+$07] | NO | YES | YES | Thm. 4 |
| halfspaces | [MORS10] | NO | YES | YES | Thm. 4 |
| functions with Fourier dimension $d$; functions with $s$-sparse Fourier spectra | [GOS$^+$09] | NO | NO | NO | |

Table 1: An overview of some Boolean function property testing results in the literature and how they relate to our results. All of the testing algorithms listed in the table are independent testers in the sense of Definition 5. The parameters $J, s, d$ are always viewed as independent of $n$. The "Theorem" column indicates which of our theorems yields a canonical tester. The testers of [GOS$^+$09] are the only algorithms that cannot be canonicalized using either Theorem 4 or Theorem 3; these testers can be verified to already essentially be in canonical form.

## B Normalizing a tester

In this section we explain how any independent tester $T$ can be "normalized;" having independent testers that satisfy this normalization condition will be useful in both our constructions that follow.

We will use the following lemma:

**Lemma 1** *Let $T$ and $C$ be as described in Theorem 3 (resp. Theorem 4). Let $S \subseteq [n]$ be any subset of $[n]$. Then the following algorithm $T(S) = (T(S)_1, T(S)_2)$ (with input parameter $\epsilon$) is also a two-sided (resp. one-sided) non-adaptive tester for $C$ with the same query and success parameters as $C$.*

*$T(S)_1$ works as follows:*

1. *Run $T_1$ with input parameter $\epsilon$ to generate the set of query strings $Q$.*
2. *For each query string $x \in Q$, if $i \in S$ then flip the bit $x_i$. Let $Q'$ be the resulting set of query strings.*
3. *Submit the query strings in $Q'$ to the oracle for $f$ and receive responses $R$.*

*$T(S)_2$ works by running $T_2$ with the set of queries $Q$ and the responses $R$.*

**Proof:** Let $f$ be the function that is being tested. Define the following function $f^*$: $f^*(x_1, \ldots, x_n) = f(t_1, \ldots, t_n)$ where $t_i = \overline{x}_i$ for $i \in S$ and $t_i = x_i$ for $i \notin S$. Thus the responses $R$ correspond to the output of $f^*$ on the strings in $Q$.

If $f$ belongs to $C$, then $f^*$ also belongs to $C$ since $C$ is closed under negating input variables. Since the pair $(Q, R)$ is consistent with $f^*$, we have $\Pr[(T')^f = \text{"accept"}] = \Pr[T^{f^*} = \text{"accept"}]$ as required.

On the other hand, if $f$ is $\epsilon$-far from $C$, then $f^*$ is $\epsilon$-far from $C$. (If this were not the case, then $f^*$ would be $\epsilon$-close to some function $g^* \in C$. Then the function $g = (g^*)^*$ would belong to $C$, and would be such that $f$ is $\epsilon$-close to $g$.) Thus, $\Pr[(T(S))^f = \text{"reject"}] = \Pr[T^{f^*} = \text{"reject"}]$ as required. ■

Given $T$,$C$ as in Theorem 3 (resp. Theorem 4), we consider the following tester $T'$. (Intuitively, in each of its executions $T'$ randomly chooses the set $S$ and runs $T(S)$ with the chosen $S$.)

The first phase $(T')_1$ works as follows:

1. Choose a subset $S \subseteq [n]$ uniformly at random (i.e. each $i \in [n]$ has independent probability $1/2$ of being in $S$).
2. Run $T(S)_1$ with input parameter $\epsilon$ to generate the set of queries $Q'$.
3. Submit the queries in $Q'$ to the oracle and receive responses $R$.

The second phase $(T')_2$ works as follows:

1. Run $T(S)_2$ with the set of queries $Q'$ and the responses $R$.

Note that by Lemma 1, $T'$ is a one-sided (resp. two-sided) non-adaptive tester for class $C$. In fact, $T'$ is a one-sided (resp. two-sided) *independent* tester for $C$ since it can be viewed in the following way: $T'$ chooses the a first query string according to probability $p_\emptyset = 1/2$. Thus $T'$ starts off with an initial partition of $[n]$ into two subsets $B_0, B_1$. Now, $T'$ executes $T$ on the set $B_0$ and executes the "negation" of $T$ (i.e. all probability values $p_b$ are replaced with $1 - p_b$) on the set $B_1$. So henceforth it will be convenient to view $T'$ as a one-sided (resp. two-sided) independent tester for $C$, with a first query string whose probability $p_\emptyset$ is set to $1/2$.

## C Construction of $\mathrm{Canon}(T)$ for classes closed under $\mathrm{Noisy\text{-}Neg}$ Minors

In this section we describe how the canonical tester $\mathrm{Canon}(T)$ for $C$ is constructed from $T$ (or more precisely, from $T'$). Before going into details we give the idea behind the construction.

We describe the canonical tester as two separate testers that are run sequentially. It should be obvious how the two testers can be combined into one tester that satisfies the canonical tester definition given in Section 3.

The first part of $\mathrm{Canon}(T)$ is denoted by $\mathrm{NoiseTest}(T)$ and the second part of $\mathrm{Canon}(T)$ is denoted by $\mathrm{Simulator}(T)$. $\mathrm{Canon}(T)$ outputs "reject" iff either $\mathrm{NoiseTest}(T)$ outputs "reject" or $\mathrm{Simulator}(T)$ outputs "reject", and outputs "accept" iff both $\mathrm{NoiseTest}(T)$ and $\mathrm{Simulator}(T)$ output "accept."

We now describe $\mathrm{NoiseTest}(T) = (\mathrm{NoiseTest}(T)_1, \mathrm{NoiseTest}(T)_2)$:

**Description of $\mathrm{NoiseTest}(T)_1$ for class $C$ running on input function $f$ with input parameter $\epsilon$:**

1. Let $\mathrm{Prod}(\epsilon)$ be as defined in Section 2.1 for the tester $T'$ with input parameter $\epsilon$.
2. We define $q'(\epsilon)$ to be the smallest integer value that satisfies the following bound:

$$2NS_{\frac{\mathrm{Prod}(\epsilon)}{2}\cdot\frac{1}{2^{q'(\epsilon)}}}(C)\cdot(q(\epsilon)) \leq \frac{r(\epsilon)-a(\epsilon)}{8}$$

   ($\mathrm{Simulator}(T)$ will subsequently make $2^{2^{q'(\epsilon)}}$ queries.)
3. Define $\eta' = \frac{rem}{2\cdot 2^{q'(\epsilon)}}$, where $rem = 2^{q'(\epsilon)} \mod \mathrm{Prod}(\epsilon)$. (We note that $\eta' < \frac{\mathrm{Prod}(\epsilon)}{2\cdot 2^{q'(\epsilon)}}$, since $rem < \mathrm{Prod}(\epsilon)$. We further recall that $NS_\delta(f)$ decreases as $\delta$ decreases for every non-constant function $f$; this is well known and is easy to verify from the Fourier expression for noise sensitivity.).
4. Let $m = \left\lceil \frac{32}{NS_{\eta'}(C)} \ln \frac{8}{r(\epsilon)-a(\epsilon)} \right\rceil$. (This choice of $m$ will be used in a Chernoff bound later in the analysis; note that it gives $\exp((-1/16)\cdot NS_{\eta'}(C)\cdot m/2) \leq \frac{r(\epsilon)-a(\epsilon)}{8}$. )
5. Let $Q$ be a set of pairs $(x^1, y^1), \ldots, (x^m, y^m)$, where each $x$ is chosen uniformly at random, and $y$ is $1-2\eta'$ correlated with $x$.
6. Submit these $m$ pairs of queries and receive the set of responses $R$: $(f(x^1), f(y^1), \ldots, (f(x^m), f(y^m))$.

**Description of $\mathrm{NoiseTest}(T)_2$ for class $C$ running on input function $f$ with input parameter $\epsilon$, queries $Q$ and responses $R$:** Output "reject" if the number of pairs such that $[f(x^i) \neq f(y^i)]$ is at least $(3/2)NS_{\eta'}(C)\cdot m$, and output "accept" otherwise.

This concludes our description of $\mathrm{NoiseTest}(T)$. Before defining $\mathrm{Simulator}(T)$, we define two useful distributions:

- **Simulator's Queries:** $\mathcal{S}_n^\epsilon$ is the distribution over sets $X$ of $2^{2^{q'(\epsilon)}}$ query strings to $n$-variable functions that a $q'$-canonical tester makes with parameter $\epsilon$.
- **Independent Tester's Queries:** $\mathcal{I}_n^\epsilon$ is the distribution over sets of queries $Y$ to $n$-variable functions that the independent tester $T'$ makes with input parameter $\epsilon$.

We are now ready to define $\mathrm{Simulator}(T) = (\mathrm{Simulator}(T)_1, \mathrm{Simulator}(T)_2)$:

**Description of $\mathrm{Simulator}(T)_1$ running on input function $f$ with parameter $\epsilon$:**

1. Draw $X \sim \mathcal{S}_n^\epsilon$. As described in Step 1 of Definition 6, this draw of $X$ induces a partition of $[n]$ into $2^{q'(\epsilon)}$ subsets: $\mathcal{P} = \{P_1, \ldots, P_{2^{q'(\epsilon)}}\}$. Note that for each fixed $j$, each variable is independently placed in subset $P_j$ with probability $\frac{1}{2^{q'(\epsilon)}}$.

2. Ask all $2^{2^{q'(\epsilon)}}$ queries $x \in X$ to the oracle and receive responses $[f(x)]_{x \in X}$.

**Description of $\mathrm{Simulator}(T)_2$ running on input parameter $\epsilon$ with queries $X$ and responses $[f(x)]_{x \in X}$:**

1. Given $X$ as above, draw $Y \sim Alt\mathcal{I}_n^\epsilon(X)$ as follows. We will later show that for $X \sim \mathcal{S}_n^\epsilon$, a set of queries $Y$ generated by this draw is distributed identically to a set of queries $Y$ generated by a draw from $\mathcal{I}_n^\epsilon$. Moreover, $\mathrm{Simulator}(T)$ can always respond correctly to queries in $y \in Y$ with the value $f'(y)$ by returning $f(x)$ for some $x \in X$ (as described below).

   - Let $\mathrm{Prod}(\epsilon_1)$ be as described in Section 4 and let $rem$ denote $2^{q'(\epsilon)} \mod \mathrm{Prod}$ (note that $rem$ is always divisible by 2 due to the way $T'$ chooses the set $S$).
   - Choose the first $rem$ subsets $P_1, \ldots, P_{rem}$ of the partition $\mathcal{P}$ defined by $X$. Let $F_+ = \bigcup_{i=1}^{rem/2} P_i$ and let $F_- = \bigcup_{i=rem/2+1}^{rem} P_i$.
   - Partition $F_+$ multinomially into $\mathrm{Prod}(\epsilon)$ subsets $\mathcal{F}_+ = \{F_+^1, \ldots, F_+^{\mathrm{Prod}(\epsilon)}\}$, where each variable in $F_+$ is assigned to subset $F_+^i$, for $1 \leq i \leq \mathrm{Prod}(\epsilon)$ independently with probability $1/\mathrm{Prod}(\epsilon)$.
   - Partition $F_-$ multinomially into $\mathrm{Prod}(\epsilon)$ subsets $\mathcal{F}_- = \{F_-^1, \ldots, F_-^{\mathrm{Prod}(\epsilon)}\}$, where each variable in $F_-$ is assigned to subset $F_-^i$, for $1 \leq i \leq \mathrm{Prod}(\epsilon)$ independently with probability $1/\mathrm{Prod}(\epsilon)$.
   - Recall that a run of $T'$ with parameter $\epsilon$ induces a partition of $[n]$ into $2^{q(\epsilon)}$ subsets. For each $1 \leq i \leq 2^{q(\epsilon)}$, let $k_i$ be such that $k_i \cdot n/\mathrm{Prod}(\epsilon)$ is the expected size of the $i$-th subset in this partition. Let $K$ equal $\lfloor \frac{2^{q'(\epsilon)}}{\mathrm{Prod}(\epsilon)} \rfloor$.
   - We now describe how to construct a partition $\mathcal{R} = \{R_1, \ldots, R_{2^{q(\epsilon)}}\}$. To construct subset $R_1$, remove the first $Kk_1$ subsets $P_{rem+1}, \ldots, P_{rem+Kk_1}$ remaining in the partition $\mathcal{P}$, the first $k_1$ subsets $F_+^1, \ldots, F_+^{k_1}$ in the partition $\mathcal{F}_+$, and the first $k_1$ subsets $F_-^1, \ldots, F_-^{k_1}$ in the partition $\mathcal{F}_-$, and place the elements of each of these sets in $R_1$. To construct subset $R_i$, for $2 \leq i \leq 2^{q(\epsilon)}$, remove the first $Kk_i$ remaining subsets in $\mathcal{P}$, the first $k_i$ remaining subsets in $\mathcal{F}_+$, and the first $k_i$ remaining subsets in $\mathcal{F}_-$ and place the elements of each of these sets in $R_i$.
   - Let $Y$ be the set of queries asked by $T'$ running with input parameter $\epsilon$ given the partition $\mathcal{R}$.
   - $\mathrm{Simulator}(T)$ answers queries in $Y$ of the form $(x_1, \ldots, x_n)$ with $f'(x_1, \ldots, x_n)$ where the function $f'(x_1, \ldots, x_n)$ equals $\mathrm{Noisy}(f, F_+, F_-)(x_1, \ldots, x_n)$ (recall Definition 3).

   We denote by $\mathcal{F}^\epsilon$ the (randomized) operator that takes input $X$ and returns the triple $(f', F_+, F_-)$

2. $\mathrm{Simulator}(T)_2$ hands the query strings and responses to $T_2'$ and outputs whatever $T_2'$ does.

The reader may have noticed that according to the above description the procedure $\mathrm{Simulator}(T)_2$ is a randomized algorithm, whereas our definition of a canonical tester requires that the "computation stage" be a deterministic algorithm. However, it is easy to derandomize the computation stage by directly applying the argument of Section 4.2.3 of [GT03].

## D Analysis of $\mathrm{Canon}(T)$ for classes closed under $\mathrm{Noisy\text{-}Neg}$ Minors

**Theorem 5** $\mathrm{Canon}(T)$ *is a canonical property tester for $C$ which makes $2^{2^{q'(\epsilon)}}$ queries and satisfies the following:*

- *if $f$ belongs to $C$ then* $\Pr[\mathrm{Canon}(T)^f = \text{“accept”}] \geq 1 - a'(\epsilon)$*, and*
- *if $f$ is $\epsilon$-far from $C$ then* $\Pr[\mathrm{Canon}(T)^f = \text{“reject”}] \geq r'(\epsilon)$*,*

*where $a'(\epsilon) = 3a(\epsilon)/4 + r(\epsilon)/4$ and $r'(\epsilon) = 3r(\epsilon)/4 + a(\epsilon)/4$.*

Theorem 5 is proved via the following three intermediate lemmas:

**Lemma 2** *If $NS_{\eta'}(f) \geq 2NS_{\eta'}(C)$ then* $\mathrm{NoiseTest}(T)$ *outputs “accept” with probability at most $\frac{r(\epsilon)-a(\epsilon)}{8}$. If $NS_{\eta'}(f) \leq NS_{\eta'}(C)$ then* $\mathrm{NoiseTest}(T)$ *outputs “reject” with probability at most $\frac{r(\epsilon)-a(\epsilon)}{8}$.*

This is a straightforward consequence of the monotonicity of $NS_\delta$ as a function of $\delta$, standard Chernoff bounds, and the choice of $m$ in Step 4 of $\mathrm{NoiseTest}(T)_1$.

**Lemma 3** *The distributions $\mathcal{I}_n^\epsilon$ and $X \sim \mathcal{S}_n^\epsilon$, $Alt\mathcal{I}_n^\epsilon(X)$ are identical.*

This lemma follows directly from inspection of the procedure described in Step 1 of $\mathrm{Simulator}(T)_2$ to obtain a draw from $Alt\mathcal{I}_n^\epsilon(X)$. The $i$-th subset in a partition induced by a set of queries $Y$ drawn from $\mathcal{I}_n^\epsilon$ is a random subset of $[n]$ obtained by independently including each variable with probability $k_i \cdot n/\mathrm{Prod}(\epsilon)$, and it is straightforward to check that the same is true for $Y$ drawn from $Alt\mathcal{I}_n^\epsilon(X)$; similar equivalences are easily seen to be true for all of the other subsets, and indeed for all collections of subsets.

To state the third lemma we need the following definition and claim: $\mathrm{Indep}(n, p_1, p_2)$ is a distribution over triples $(f', F_+, F_-)$, where $f'$ is an $n$-variable function and $F_+, F_- \subseteq [n]$ are disjoint sets. A draw from $\mathrm{Indep}(n, p_1, p_2)$ is obtained in the following way:

- Each index $i \in [n]$ is independently placed in $F_+$ with probability $p_1$, placed in $F_-$ with probability $p_2$, and placed in neither set with probability $1 - p_1 - p_2$.
- $f'$ is set to be $\mathrm{Noisy}(f, F_+, F_-)(x_1, \ldots, x_n)$.

**Claim 4** *The following two distributions are identical:*

- *Draw $Y \sim (X \sim \mathcal{S}_n^\epsilon, Alt\mathcal{I}_n^\epsilon(X))$. Draw $(f', F_+, F_-) \sim \mathrm{Indep}(n, \eta' = \frac{rem}{2^{q'(\epsilon)+1}}, \eta' = \frac{rem}{2^{q'(\epsilon)+1}})$ and output $(Y, f', F_+, F_-)$; and*
- *Draw $X \sim \mathcal{S}_n^\epsilon$. Draw $Y \sim Alt\mathcal{I}_n^\epsilon(X)$ and output $(Y, \mathcal{F}^\epsilon(X))$.*

This claim follows from inspection of the specification of the operator $\mathcal{F}^\epsilon(X)$ given in Step 1 of the description of $\mathrm{Simulator}(T)_2$.

Now we give the third lemma needed to prove Theorem 5:

**Lemma 5** *If $NS_{\eta'}(f) \leq 2NS_{\eta'}(C)$ then*

$$\Pr_{X \sim \mathcal{S}_n^\epsilon, (f', F_+, F_-) = \mathcal{F}^\epsilon(X), Y \sim Alt\mathcal{I}_n^\epsilon(X)}[f(y) \neq f'(y) \text{ for some } y \in Y] \leq \frac{r(\epsilon) - a(\epsilon)}{8}.$$

**Proof:** By Claim 4 we have that

$$\begin{aligned} & \Pr_{X\sim\mathcal{S}_n^\epsilon,(f',F_+,F_-)=\mathcal{F}^\epsilon(X),Y\sim Alt\mathcal{I}_n^\epsilon(X)}[f(y)\neq f'(y) \text{ for some } y\in Y] \\ = & \Pr_{Y\sim(X\sim\mathcal{S}_n^\epsilon,Alt\mathcal{I}_n^\epsilon(X)),(f',F_+,F_-)\sim\mathrm{Indep}(n,\eta',\eta')}[f(y)\neq f'(y) \text{ for some } y\in Y]. \end{aligned}$$

Thus, it is sufficient to consider

$$\Pr_{Y\sim(X\sim\mathcal{S}_n^\epsilon,Alt\mathcal{I}_n^\epsilon(X)),(f',F_+,F_-)\sim\mathrm{Indep}(n,\eta',\eta')}[f(y)\neq f'(y) \text{ for some } y\in Y].$$

We first observe that each individual query $y \in Y$ above is uniformly distributed. Now given an individual query $y \in Y$, we define $y' = y'_1, \ldots, y'_n$ where $y'_i = 1$ if $i \in F_+$, $y'_i = -1$ if $i \in F_-$ and $y'_i = y_i$ otherwise. Note that by definition of $f'$ we have $f'(y) = f(y')$. Moreover, since variables are placed in $F_+$ (resp. $F_-$) and set to 1 (resp. $-1$) independently with probability $\frac{rem}{2^{q'(\epsilon)+1}} = \eta'$, we have that $y$ and $y'$ are $(1-2\eta')$-correlated. Thus, since by assumption $NS_{\eta'}(f) \leq 2NS_{\eta'}(C) \leq \frac{r(\epsilon)-a(\epsilon)}{8q(\epsilon)}$, we have that for each individual query $y$, the probability that $f(y) \neq f(y') = f'(y)$ is at most $\frac{r(\epsilon)-a(\epsilon)}{8q(\epsilon)}$. By a union bound over all $y \in Y$, we have that the probability that $f(y) \neq f'(y)$ for some $y \in Y$ is at most $\frac{r(\epsilon)-a(\epsilon)}{8}$. ■

Having established Lemmas 2, 3, and 5, we now show that they imply Theorem 5.

**Proof of Theorem 5:** First, suppose that $f$ belongs to $C$. Thus we have that $NS_{\eta'}(f) \leq NS_{\eta'}(C)$. Since by Lemma 3, the distributions $\mathcal{I}_n^\epsilon$ and $X \sim \mathcal{S}_n^\epsilon$, $Alt\mathcal{I}_n^\epsilon(X)$ are identical, we have that the output of $\mathrm{Canon}(T)$ can only differ from the output of $T$ in two cases: (1) $\mathrm{NoiseTest}(T)$ outputs "reject". (2) $f(y) \neq f'(y)$ for some $y \in Y$. Since $f \in C$, we have $NS_{\eta'}(f) \leq NS_{\eta'}(C)$, so by Lemma 2, we have that (1) occurs with probability at most $\frac{r(\epsilon)-a(\epsilon)}{8}$. Moreover, since $NS_{\eta'}(f) \leq NS_{\eta'}(C)$, we have by Lemma 5, that (2) occurs with probability at most $\frac{r(\epsilon)-a(\epsilon)}{8}$. Thus, we have that $\mathrm{Canon}(T)$ outputs "accept" with probability at least

$$1 - (3a(\epsilon)/4 + r(\epsilon)/4) = 1 - a'(\epsilon).$$

Next, suppose that $f$ is $\epsilon$-far from $C$. There are two cases to consider. The first case is that $NS_{\eta'}(f) \geq 2NS_{\eta'}(C)$. In this case, $\mathrm{NoiseTest}(T)$ outputs reject with probability at least $1 - \frac{r(\epsilon)-a(\epsilon)}{8}$ by Lemmas 2. The second case is that $NS_{\eta'}(f) \leq 2NS_{\eta'}(C)$. Since $\mathrm{Canon}(T)$ always outputs "reject" if $\mathrm{NoiseTest}(T)$ outputs "reject," the probability that $\mathrm{Canon}(T)$ outputs "reject" is at least the probability that $\mathrm{Canon}(T)$ outputs "reject" given that $\mathrm{NoiseTest}(f)$ outputs "accept". Since by Lemma 3, the distributions $\mathcal{I}_n^\epsilon$ and $X \sim \mathcal{S}_n^\epsilon$, $Alt\mathcal{I}_n^\epsilon(X)$ are identical, we have that if $\mathrm{NoiseTest}(T)$ outputs "accept", the output of $\mathrm{Canon}(T)$ can only differ from the output of $T$ if $f(y) \neq f'(y)$ for some $y \in Y$. By Lemma 5, we have that this occurs with probability at most $\frac{r(\epsilon)-a(\epsilon)}{8}$. Thus, $\mathrm{Canon}(T)$ outputs "reject" with probability at least

$$3r(\epsilon)/4 + a(\epsilon)/4 = r'(\epsilon).$$

and the theorem is proved. ■

# E Construction of $\mathrm{Canon}(T)$ for classes closed under ID-Neg minors

In this section we describe how the canonical tester $\mathrm{Canon}(T)$ for $C$ is constructed from $T$ (or more precisely, from $T'$). Throughout this section $q(\epsilon)$ denotes the query complexity of $T'$ on input parameter $\epsilon$, and $c(\epsilon)$ denotes the granularity of $T'$ as defined in Definition 5.

We begin by defining two useful distributions (the same definitions that were used in Appendix C):

- **Canonical Tester's Queries:** $\mathcal{S}_n^\epsilon$ is the distribution over sets of $2^{2^{q'(\epsilon)}}$ queries $X$ to $n$-variable functions that a $q'$-canonical tester makes when it is run with parameter $\epsilon$.
- **Independent Tester's Queries:** $\mathcal{I}_n^\epsilon$ is the distribution over sets of $q(\epsilon)$ queries $Y$ to $n$-variable functions that the independent tester $T'$ makes with input parameter $\epsilon$.

We are now ready to define the canonical tester $\mathrm{Canon}(T) = (\mathrm{Canon}(T)_1, \mathrm{Canon}(T)_2)$:

**Description of $\mathrm{Canon}(T)_1$ running on input function $f$ with parameter $\epsilon$:**

1. Draw $X \sim \mathcal{S}_n^\epsilon$. As described in Step 1 of Definition 6, this draw of $X$ induces a partition of $[n]$ into $2^{q'(\epsilon)}$ subsets: $\mathcal{P} = \{P_1, \ldots, P_{2^{q'(\epsilon)}}\}$. Note that for each fixed $j$, each variable is independently placed in subset $P_j$ with probability $\frac{1}{2^{q'(\epsilon)}}$.
2. Ask all $2^{2^{q'(\epsilon)}}$ queries $x \in X$ to the oracle and receive responses $[f(x)]_{x \in X}$.

**Description of $\mathrm{Canon}(T)_2$ running on input parameter $\epsilon$ with queries $X$ and responses $[f(x)]_{x \in X}$:**

1. Given input $X$ as above, the 4-tuple $(f', F_+, F_-, id)$ is obtained by applying $\mathcal{F}^{\epsilon_1}$ to $X$ as follows: (recall from the statement of Theorem 4 that $\epsilon_1 = \frac{r(\epsilon)}{4q(\epsilon)}$)
   - Let $\mathrm{Prod}(\epsilon_1)$ be as described in Section 4. (Recall that as noted in Section 4 we have $\mathrm{Prod}(\epsilon_1) \leq c(\epsilon_1)^{2^{q(\epsilon_1)+1}}$.) Let $rem$ denote $2^{q'(\epsilon)} \mod \mathrm{Prod}$ (as before, $rem$ is divisible by 2 due to the way $T'$ chooses the set $S$).
   - Choose the first $rem$ subsets $P_1, \ldots, P_{rem}$ of the partition $\mathcal{P}$ defined by $X$. Let $F_+ = \bigcup_{i=1}^{rem/2} P_i$ and let $F_- = \bigcup_{i=rem/2+1}^{rem} P_i$. Let $id$ be the lexicographically first element of $F_+$.
   - Let $n'$ be the value such that $n' - 1 = n - |F_+| - |F_-|$. (Intuitively, $n'$ is the number of elements in $[n]$ after the elements of $F_+$ and $F_-$ are removed and then $id$ is added back in.)
   - Let $f' = \text{ID-Neg}(f, F_+, F_-, id)(x_1, \ldots, x_n)$. We shall view $f'$ as a function over $n'$ variables.
2. Given $X, (f', F_+, F_-, id)$ generated as above, draw $Y \sim Alt\mathcal{I}_{n'}^{\epsilon_1}(X, id)$ as follows. We will later show that a set of $n'$-variable queries $Y$ generated by this draw is distributed identically to a set of queries $Y$ generated by a draw from $\mathcal{I}_{n'}^{\epsilon_1}$. Moreover, $\mathrm{Canon}(T)$ can always respond correctly to queries $y \in Y$ with the value $f'(y)$ by returning $f(x)$ for some $x \in X$ as described below.
   - Let $2^{q(\epsilon_1)}$ be the number of subsets in the partition $\mathcal{R} = \{R_1, \ldots, R_{2^{q(\epsilon_1)}}\}$ induced by $T'$ running with parameter $\epsilon_1$. Let $k_i$ be the nonnegative integer such that the expected size of subset $R_i$, equals $k_i \cdot n/\mathrm{Prod}(\epsilon_1)$. Let $K = \lfloor \frac{2^{q'(\epsilon)}}{\mathrm{Prod}(\epsilon_1)} \rfloor$.
   - To construct subset $R_1$, remove the first $k_1$ subsets $P_{rem+1}, \ldots, P_{rem+Kk_1}$ remaining in the partition $\mathcal{P}$ and place the elements of each of the sets in $R_1$. To construct subset $R_i$, for $2 \leq i \leq 2^{q(\epsilon_1)}$, remove the first $Kk_i$ remaining subsets in $\mathcal{P}$ and place the elements of each of the sets in $R_i$.
   - Randomly choose a subset $R_j$, where each subset $R_i$ has probability $k_i/\mathrm{Prod}(\epsilon_1)$ of being chosen. Place the variable $x_{id}$ in the subset $R_j$.
   - Let $Y$ be the set of queries asked by $T'$ running with input parameter $\epsilon_1$ given the partition $\mathcal{R}$.
   - $\mathrm{Canon}(T)$ answers queries in $Y$ of the form $(x_1, \ldots, x_{n'})$ with $f'(x_1, \ldots, x_n)$ where the function $f'(x_1, \ldots, x_n)$ equals $\text{ID-Neg}(f, F_+, F_-, id)$ (recall Definition 4).
3. $\mathrm{Canon}(T)$ hands the query strings and responses to $T_2'$ and outputs whatever $T_2'$ does.

As at the end of Appendix C, the randomness in the computation stage can be eliminated using the [GT03] approach.

# F Analysis of $\mathrm{Canon}(T)$ for classes closed under ID-Neg Minors

In this section we prove the following result which implies Theorem 4:

**Theorem 6** $\mathrm{Canon}(T)$ *is a canonical property tester for $C$ which makes $2^{2^{q'(\epsilon)}}$ queries and satisfies the following:*

- *if $f$ belongs to $C$ then* $\Pr[\mathrm{Canon}(T)^f =$ *"accept"*$] = 1$*; and*
- *if $f$ is $\epsilon$-far from $C$ then* $\Pr[\mathrm{Canon}(T)^f =$ *"reject"*$] \geq (1 - \frac{1-r(\epsilon)}{1-r(\epsilon)/4}) \cdot r(\epsilon_1)$.

We prove Theorem 6 using the following two intermediate lemmas. In Lemma 6 we write "$id(X)$" to denote the element $id$ that is determined by $X$ (see Step 1 of the description of $\mathrm{Canon}(T)_2$).

**Lemma 6** *Let $T$ be a one sided independent tester for $C$. Fix any possible outcome $\tilde{X}$ of draws from $\mathcal{S}^{\epsilon}_n$ and let $(\tilde{F}_+, \tilde{F}_-, \tilde{f}', \tilde{id})$ be the 4-tuple that is obtained from $\tilde{X}$ by applying the operator $\mathcal{F}^{\epsilon_1}$ to $\tilde{X}$ as described in Section 4.2. Then we have*

$$\Pr_{X \sim \mathcal{S}^{\epsilon}_n, Y \sim Alt\mathcal{I}^{\epsilon_1}_{n'}(X, id(X))} [\mathrm{Canon}(T) \textit{ outputs "accept"} \mid \mathcal{F}^{\epsilon_1}(X) \textit{ equals } (\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id}) \textit{ and } \tilde{f}' \in C] = 1$$

*and*

$$\Pr_{X \sim \mathcal{S}^{\epsilon}_n, Y \sim Alt\mathcal{I}^{\epsilon_1}_{n'}(X, id(X))} [\mathrm{Canon}(T) \textit{ outputs "reject"} \mid \mathcal{F}^{\epsilon_1}(X) \textit{ equals } (\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id}) \textit{ and } \tilde{f}' \textit{ is } \epsilon_1\textit{-far from } C] \geq r(\epsilon_1).$$

**Lemma 7** *Suppose $f$ is $\epsilon$-far from $C$. Let $X$ be drawn from $\mathcal{S}^{\epsilon}_n$ and let $(f', F_+, F_-, id)$ be obtained by applying $\mathcal{F}^{\epsilon_1}$ to $X$. Then with probability at least $1 - \frac{1-r(\epsilon)}{1-r(\epsilon)/4}$, the function $f'$ (over $n'$ variables) is $\epsilon_1$-far from $C$.*

We first show that Lemmas 6 and 7 imply Theorem 6.

**Proof:** First, assume $f$ belongs to $C$. Then since $C$ is closed under ID-Neg minors, we have that for any sequence $\tilde{X}$ of draws, the function $\tilde{f}'$ resulting from $\mathcal{F}^{\epsilon_1}(\tilde{X})$ is in $C$. Thus, by Lemma 6 we have that $\mathrm{Canon}(T)$ outputs "accept" with probability 1.

Next, assume $f$ is $\epsilon$-far from $C$. Then by Lemma 7 we have that with probability $1 - \frac{1-r(\epsilon)}{1-r(\epsilon)/4}$, $f'$ is $\epsilon_1$-far from $C$. If $f'$ is $\epsilon_1$-far from $C$ then we have by Lemma 6 that $\mathrm{Canon}(T)$ outputs "reject" with probability $r(\epsilon_1)$. Thus, $\mathrm{Canon}(T)$ outputs "reject" on $f$ with overall probability at least $(1 - \frac{1-r(\epsilon)}{1-r(\epsilon)/4}) \cdot r(\epsilon_1)$. ■

We now prove Lemmas 6 and 7. To prove Lemma 6, we will use the following claim, the correctness of which follows from inspection of the specification of $Alt\mathcal{I}^{\epsilon_1}_{\tilde{n}'}(X)$ as given in Step 2 of the description of $\mathrm{Canon}(T)_2$ in Appendix E.

**Claim 8** *Fix any possible outcome $\tilde{X}$ of draws from $\mathcal{S}^{\epsilon}_n$ and let $(\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id})$ be the 4-tuple that is obtained from $\tilde{X}$ by applying the operator $\mathcal{F}^{\epsilon_1}$ to $\tilde{X}$ as described in Section 4.2 (note that once $(\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id})$ has been determined this fixes the value of $\tilde{n}'$, the number of variables that $\tilde{f}'$ is defined over). Then the two distributions*

- *Draw $Y \sim \mathcal{I}^{\epsilon_1}_{\tilde{n}'}$ and output $Y$; and*

- *Draw $X \sim \mathcal{S}^{\epsilon}_n$, conditioned on $\mathcal{F}^{\epsilon_1}(X) = (\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id})$. Draw $Y \sim Alt\mathcal{I}^{\epsilon_1}_{\tilde{n}'}(X)$ and output $Y$*

*are identical.*

We are now ready to prove Lemma 6.

**Proof:** By Claim 8, the distributions $\mathcal{I}^{\epsilon_1}_{n'}$ and $Alt\mathcal{I}^{\epsilon_1}_{n'}(X, id(X))$ are identical. Additionally, the canonical tester is able to answer correctly with respect to $f'$ every query in $Y$ where $Y \sim Alt\mathcal{I}^{\epsilon_1}_{n'}(X)$. Since $T'$ is a one-sided tester for class $C$, it must be the case that if $f' \in C$, $\mathrm{Canon}(T)$ outputs "accept" and if $f'$ is $\epsilon_1$-far from $C$ then $\mathrm{Canon}(T)$ outputs "reject" with probability $r(\epsilon_1)$. ■

Before proving Lemma 7, we define the following distribution:

**Alternative view of distribution over functions $f'$:** $Alt\mathcal{F}^{\epsilon_1}(Y)$ is a distribution over 4-tuples $(f', F_+, F_-, id)$, where $f'$ is an $n'$-variable function (for $n' = n - |F_+| - |F_-| + 1$), $F_+$ and $F_-$ are disjoint subsets of $[n]$, $id$ is the lexicographically first element of $F_+$, and $Y$ is drawn from $\mathcal{I}^{\epsilon_1}_n$.

Given $Y$ drawn from $\mathcal{I}^{\epsilon_1}_n$, a draw from $Alt\mathcal{F}^{\epsilon_1}(Y)$ is obtained in the following way:

- Let $\mathrm{Prod}(\epsilon)$ and $\mathrm{Prod}(\epsilon_1)$ be defined according to the tester $T'$ as described in Section 2.1.
- Let $R_+, R_-$ be a fixed pair of two subsets of the partition defined by $Y$ that have the following property: for every query string $y \in Y$, for all pairs $(i, j)$ such that $i \in R_+$, $j \in R_-$, we have $y_i = \overline{y}_j$. (Note that such subsets are guaranteed to exist by the way that $T'$ performs its first query.) Let $k_+$ be such that the expected size of $R_+$ is $k_+ \cdot n$, and similarly the expected size of $R_-$ is $k_- \cdot n$.
- Choose the subsets $F_+$, ($F_-$) from $R_+$, ($R_-$) by placing each variable from $R_+$ ($R_-$) in $F_+$ ($F_-$) independently with probability $p_{R_+} = \frac{rem}{2 \cdot k_+ \cdot 2^{q'(\epsilon)}}$. Note that due to the definition of $q'(\epsilon)$ and the fact that $rem \leq \mathrm{Prod}(\epsilon_1)$, $k_+ \geq 1/\mathrm{Prod}(\epsilon)$ ($k_- \geq 1/\mathrm{Prod}(\epsilon)$) we must have that $p_{R_+} \leq 1$ ($p_{R_-} \leq 1$).
- Choose the element $id$ to be the lexicographic first element of $F_+$.
- Set $f' = \text{ID-Neg}(f, F_+, F_-, id)$.

The following claim shows that the distribution $Y \sim \mathcal{I}^{\epsilon}_n$, $Alt\mathcal{F}^{\epsilon_1}(Y)$ is identical to the distribution $X \sim \mathcal{S}^{\epsilon}_n$, $\mathcal{F}^{\epsilon_1}(X)$ used by $\mathrm{Canon}(T)$.

**Claim 9** *The following two distributions are identical:*

- *$\mathcal{F}^{\epsilon_1}(X)$ (where $X$ is drawn from $\mathcal{S}^{\epsilon}_n$) and*
- *$Alt\mathcal{F}^{\epsilon_1}(Y)$ (where $Y$ is drawn from $\mathcal{I}^{\epsilon}_n$).*

We require two more definitions:

- **Restricting independent Tester's queries to $n'$ variables:** $Restrict(Y, f', F_+, F_-, id)$ is an operator over sets of queries $Z$, where $Y \sim \mathcal{I}^{\epsilon}_n$, $(f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)$. A draw from $Restrict(Y, f', F_+, F_-, id)$ is obtained in the following way:
  - Let $Z$ be the restriction of $Y$ to the variables in $[n] \setminus (F_+ \cup F_-) \cup \{id\}$.
- **Alternate view of distribution over independent Tester's queries over $n$ variables restricted to $n'$ variables:** $AltRestrict^{\epsilon}$ is a distribution over 5-tuples $(f', F_+, F_-, id, Z)$. A draw from $AltRestrict^{\epsilon}$ is obtained in the following way:

- Choose the subsets $F_+, F_-, [n] \setminus (F_+ \cup F_-)$ by independently placing each variable in $F_+$ with probability $\frac{rem}{2 \cdot 2^{q'(\epsilon)}}$, $F_-$ with probability $\frac{rem}{2 \cdot 2^{q'(\epsilon)}}$, and otherwise in $[n] \setminus (F_+ \cup F_-)$.
- Fix $id$ to be the lexicographically first element of $F_+$, and set $f'$ to be $f' = \text{ID-Neg}(f, F_+, F_-, id)$.
- Let $\mathcal{T}_n^\epsilon$ be the distribution over sets of $q(\epsilon)$ queries $Y$ to $n$-variable functions that $T$ makes with input parameter $\epsilon$. Note that for a draw of $Y \sim \mathcal{T}_n^\epsilon$ there is a subset $R$ that corresponds to $R_+ \cup R_-$, where $R_-, R_+$ are the subsets described in the "Alternative view of distribution over functions $f'$" given above. Draw $Y \sim \mathcal{T}_n^\epsilon$ conditioned on the indices in $F+, F_-$ being placed in this partition subset $R$. Set $Y'$ to be the restriction of $Y$ to the variables in $[n] \setminus (F_+ \cup F_-)$.
- Choose a subset $S' \subseteq [n] \setminus (F_+ \cup F_-) \cup \{id\}$ uniformly at random (i.e. each $i \in [n] \setminus (F_+ \cup F_-) \cup \{id\}$ has independent prob. $1/2$ of being in $S'$, as in Section B).
- For each query in $Y'$, flip the value of $x_i$ for all $i \in S'$ and add the query to $Z$.

We have the following claim:

**Claim 10** *The following two distributions over sets of queries $Z$ are identical:*

- *Draw $Y$ from $\mathcal{I}_n^\epsilon$, draw $(f', F_+, F_-, id)$ from $Alt\mathcal{F}^{\epsilon_1}(Y)$, and output $Z = Restrict(Y, f', F_+, F_-, id)$.*
- *Draw $(f', F_+, F_-, id, Z)$ from $AltRestrict^\epsilon$ and output $Z$.*

Finally, we define one more operator:

**Alternate view of a single query made by the independent Tester restricted to $n'$ variables:** For $1 \leq i \leq q(\epsilon)$ and $Z$ generated from $AltRestrict^\epsilon$ as described above, we define $Query^i(Z)$ as the single query string obtained by returning the $i$-th query string from $Z$.

The following claim shows that a draw from $Query^i(Z)$ is uniformly distributed.

**Claim 11** *Fix any possible outcome $(\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id}, \tilde{Z})$ from $AltRestrict^\epsilon$. Fix any $1 \leq i \leq q(\epsilon)$. The following two distributions are identical:*

- *Output $(\tilde{f}', U_{n'})$ (i.e. the second coordinate is a uniform random $n'$-bit string); and*
- *Draw $(f', F_+, F_-, id, Z)$ from $AltRestrict^\epsilon$, conditioned on $(f', F_+, F_-, id)$ being identical to $(\tilde{f}', \tilde{F}_+, \tilde{F}_-, \tilde{id})$. Output $(\tilde{f}', Query^i(Z))$*

**Proof:** By the definition of $AltRestrict^\epsilon$, no matter what is the outcome of $(f', F_+, F_-, id)$, the final choice of $S'$ makes the $i$-th query string in $Z$ uniform random. ■

Using the defined distributions and corresponding claims, we are now ready to prove Lemma 7:

**Proof of Lemma 7:** Assume that $f$ is $\epsilon$-far from $C$. Let $X$ be drawn from $\mathcal{C}_n^\epsilon$, and let $(f', F_+, F_-, id)$ be $\mathcal{F}^{\epsilon_1}(X)$. Let $p_1$ denote the probability that $f'$ (a function over $n'$ variables) is $\epsilon_1$-close to $C$. We will upper bound $p_1$ and thus prove the lemma.

Let $Y$ be drawn from $\mathcal{I}_n^\epsilon$, and let $(f', F_+, F_-, id)$ be drawn from $Alt\mathcal{F}^{\epsilon_1}(Y)$. By Claim 9 the two distributions $X \sim \mathcal{S}_n^\epsilon$, $\mathcal{F}^{\epsilon_1}(X)$ and $Y \sim \mathcal{I}_n^\epsilon$, $Alt\mathcal{F}^{\epsilon_1}(Y)$ are identical. So we view $f'$ as chosen by a draw from $Alt\mathcal{F}^{\epsilon_1}(Y)$. Let us now additionally consider a draw $Z$ from the distribution $Y \sim \mathcal{I}_n^\epsilon$, $(f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)$, $Restrict(Y, f', F_+, F_-, id)$.

We have that by Claim 10 that the two distributions over $Z$

$$Y \sim \mathcal{I}_n^\epsilon, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y), Z \sim Restrict(Y, f', F_+, F_-, id)$$

and

$$Z \sim AltRestrict^{\epsilon}$$

are identical, so we can alternatively view $Z$ as a draw from $AltRestrict^{\epsilon}$.

Since for $1 \leq i \leq q(\epsilon)$, $(f', Z \sim AltRestrict^{\epsilon}, Query^{i}(Z)) \equiv (f', U_{n'})$, Claim 11 tells us that each query in $Z$ is uniformly distributed. We note that if $f'$ over $n'$ variables is $\epsilon_1$-close to $C_{n'}$, then this implies that there is a function $g' \in C_{n'}$ such that $\Pr_{z \sim U_{n'}}[f'(z) \neq g'(z)] < \epsilon_1$.

By a union bound since the queries in $Z$ are individually uniformly distributed, the probability that one of the queries $z$ in $Z$ is such that $f'(z) \neq g'(z)$ is at most $q(\epsilon) \cdot \epsilon_1$. Alternatively, by viewing $Z$ as a draw from the distribution $Y \sim \mathcal{I}_n^{\epsilon}$, $(f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)$, $Restrict(Y, f', F_+, F_-, id)$, we have that the probability that all the queries $y$ in $Y \sim \mathcal{I}_n^{\epsilon}$ are such that $f'(y) = g(y)$ (where $g(y) \in C_n$ is obtained from $g'(y)$ by adding back the irrelevant variables corresponding to the indices in $(F_+ \cup F_-) \setminus \{id\}$, and we view $f'$ here as an $n$-variable function) is also at least $(1 - q(\epsilon) \cdot \epsilon_1)$. Since $T'$ is one-sided, and since we have that for any $Y \sim \mathcal{I}_n^{\epsilon}$, $(f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)$, it must be the case that $f(y) = f'(y)$ for all $y \in Y$, this means that:

$$\begin{aligned}
&\Pr_{Y \sim \mathcal{I}_n^{\epsilon}}[T_2'(Y) \text{ accepts}] \\
&= \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[T_2'(Y) \text{ accepts}] \\
&\geq \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[T_2'(Y) \text{ accepts} \wedge f' \text{ is } \epsilon_1\text{-close to } C_{n'}] \\
&\geq \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[f' \text{ is } \epsilon_1\text{-close to } C_{n'} \wedge f(y) = g(y) \text{ for all } y \in Y] \\
&= \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[f' \text{ is } \epsilon_1\text{-close to } C_{n'} \wedge f'(y) = g(y) \text{ for all } y \in Y] \\
&= \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[f' \text{ is } \epsilon_1\text{-close to } C_{n'}] \cdot \\
&\qquad \Pr_{Y \sim \mathcal{I}_n^{\epsilon}, (f', F_+, F_-, id) \sim Alt\mathcal{F}^{\epsilon_1}(Y)}[f'(y) = g(y) \text{ for all } y \in Y | f' \text{ is } \epsilon_1\text{-close to } C_{n'}]
\end{aligned}$$

So $\Pr_{Y \sim \mathcal{I}_n^{\epsilon}}[T_2'(Y) \text{ accepts }] \geq p_1 \cdot (1 - q(\epsilon) \cdot \epsilon_1)$. However, since $f$ is $\epsilon$-far from $C$, $T'$ running with input parameter $\epsilon$ must accept with probability at most $1 - r(\epsilon)$. Thus, $p_1 \cdot (1 - q(\epsilon) \cdot \epsilon_1) \leq 1 - r(\epsilon)$. So $p_1 \leq \frac{1 - r(\epsilon)}{1 - r(\epsilon)/4}$. ■