Testing Boolean Function Isomorphism

Sourav Chakraborty
(Chennai Mathematical Institute, India)

based on the works with

Noga Alon, Eric Blais, Eldar Fischer,
David García Soriano, Arie Matsliah
Let’s dive into property testing of functions

A property $P$ is just a collection of boolean functions on $\{0, 1\}^n$.

The distance between two functions $f$ and $g$ is $\text{dist}(f, g) = \Pr_{x \in \{0, 1\}^n} [f(x) \neq g(x)]$.

The function $g$ is $\epsilon$-far from $P$ if for all $f \in P$, $\text{dist}(f, g) \geq \epsilon$.

We have oracle access to some unknown boolean function $g : \{0, 1\}^n \rightarrow \{0, 1\}$.

Want to test if $g$ satisfies property $P$ or is $\epsilon$-far from it.
Let’s dive into property testing of functions

- A *property* \( \mathcal{P} \) is just a collection of boolean functions on \( \{0, 1\}^n \).
Let’s dive into property testing of functions

- A *property* $\mathcal{P}$ is just a collection of boolean functions on $\{0,1\}^n$.
- The *distance* between two functions $f$ and $g$ is
  \[
  \text{dist}(f, g) = \Pr_{x \in \{0,1\}^n} [f(x) \neq g(x)].
  \]
Let’s dive into property testing of functions

- A *property* \( \mathcal{P} \) is just a collection of boolean functions on \( \{0, 1\}^n \).
- The *distance* between two functions \( f \) and \( g \) is

\[
\text{dist}(f, g) = \Pr_{x \in \{0,1\}^n} [f(x) \neq g(x)].
\]

- The function \( g \) is *\( \epsilon \)-far from \( \mathcal{P} \) if for all \( f \in \mathcal{P} \), \( \text{dist}(f, g) \geq \epsilon \).
Let’s dive into property testing of functions

- A property $\mathcal{P}$ is just a collection of boolean functions on $\{0,1\}^n$.
- The distance between two functions $f$ and $g$ is
  $$\text{dist}(f,g) = \Pr_{x \in \{0,1\}^n} [f(x) \neq g(x)].$$
- The function $g$ is $\epsilon$-far from $\mathcal{P}$ if for all $f \in \mathcal{P}$, $\text{dist}(f,g) \geq \epsilon$.
- We have oracle access to some unknown boolean function $g : \{0,1\}^n \to \{0,1\}.$
Let’s dive into property testing of functions

- A property $\mathcal{P}$ is just a collection of boolean functions on $\{0, 1\}^n$.
- The distance between two functions $f$ and $g$ is
  \[
  \text{dist}(f, g) = \Pr_{x \in \{0, 1\}^n} [f(x) \neq g(x)].
  \]
- The function $g$ is $\epsilon$-far from $\mathcal{P}$ if for all $f \in \mathcal{P}$, $\text{dist}(f, g) \geq \epsilon$.
- We have oracle access to some unknown boolean function $g : \{0, 1\}^n \rightarrow \{0, 1\}$.
- Want to test if $g$ satisfies property $\mathcal{P}$ or is $\epsilon$-far from it.
Definition
Let $\mathcal{P}$ be a property of boolean functions on $\{0, 1\}^n$. A tester for $\mathcal{P}$ is a randomized algorithm $A$ with black box access to a function $g : \{0, 1\}^n \rightarrow \{0, 1\}$ that satisfies:

- $g \in \mathcal{P} \Rightarrow \Pr[A \text{ accepts}] \geq 2/3$.
- $g$ is $\epsilon$-far from $\mathcal{P} \Rightarrow \Pr[A \text{ rejects}] \geq 2/3$.

We allow the algorithm to be adaptive (queries may depend on the outcome of previous queries).

Can we test if $f$ is a constant function?
Query complexity for the tester $A$ is the maximum number of queries queried by the tester on any input.
Query complexity for the tester $A$ is the maximum number of queries queried by the tester on any input.

Query complexity of a property $P$ is the query complexity of the tester that has the minimum query complexity.
Query complexity for the tester $A$ is the maximum number of queries queried by the tester on any input.

Query complexity of a property $P$ is the query complexity of the tester that has the minimum query complexity.

Trivial example: let $P$ be the property “$g \equiv 0$”. Then taking $O(1/\epsilon)$ independent samples works w.h.p.
Motivation for function isomorphism

The property \( \mathcal{P} \) can be defined in terms of some known boolean function

\[
f : \{0, 1\}^n \rightarrow \{0, 1\}.
\]

- If \( \mathcal{P} = \{f\} \), it’s easy to test \( \mathcal{P} \) in \( O(1/\epsilon) \).
- But what if we are allowed to shuffle around the input variables? (\( \mathcal{P} = \{\text{permuted versions of } f\} \) )
The property $\mathcal{P}$ can be defined in terms of some known boolean function

$$f : \{0, 1\}^n \rightarrow \{0, 1\}.$$ 

- If $\mathcal{P} = \{f\}$, it's easy to test $\mathcal{P}$ in $O(1/\epsilon)$.
- But what if we are allowed to shuffle around the input variables? ($\mathcal{P} = \{\text{permuted versions of } f\}$)

Various function property testing questions can be reduced to testing of function isomorphism.
Function isomorphism

Definition (isomorphism)

Two boolean functions are *isomorphic* (in short, $f \cong g$) if they are the same up to relabelling of the variables, i.e.

$$f(x_1x_2\ldots x_n) = g(x_{\pi(1)}x_{\pi(2)}\ldots x_{\pi(n)}) \triangleq g^\pi(x_1\ldots x_n)$$

for some permutation $\pi : [n] \rightarrow [n]$. 

Examples:

$f(x_1x_2x_3) = x_1 \lor (x_2 \land x_3)$ is isomorphic to $g(x_1x_2x_3) = x_3 \lor (x_1 \land x_2)$.

The function $f(x_1x_2x_3) = \text{majority}(x_1x_2x_3)$ is only isomorphic to itself (because it is symmetric).
Function isomorphism

Definition (isomorphism)

Two boolean functions are *isomorphic* (in short, $f \cong g$) if they are the same up to relabelling of the variables, i.e.

$$f(x_1 x_2 \ldots x_n) = g(x_{\pi(1)} x_{\pi(2)} \ldots x_{\pi(n)}) \triangleq g^{\pi}(x_1 \ldots x_n)$$

for some permutation $\pi : [n] \to [n]$.

Examples:

- $f(x_1 x_2 x_3) = x_1 \vee (x_2 \land x_3)$ is isomorphic to $g(x_1 x_2 x_3) = x_3 \vee (x_1 \land x_2)$.
- The function $f(x_1 x_2 x_3) = \text{majority}(x_1 x_2 x_3)$ is only isomorphic to itself (because it is *symmetric*).
The *distance up to isomorphism* between $f$ and $g$ is

$$\text{distiso}(f, g) = \min_{\pi \in \mathcal{S}_n} \text{dist}(f, g^\pi)$$

For example, consider two parities $f(x_1, \ldots, x_n) = x_1 \oplus x_2 \ldots \oplus x_k$ and $g(x_1, \ldots, x_n) = x_100 \oplus \ldots \oplus x_{100+k}$. Then $k = k' \Rightarrow \text{distiso}(f, g) = 0$. $k \neq k' \Rightarrow \text{distiso}(f, g) = \frac{1}{2}$.
The \emph{distance up to isomorphism} between $f$ and $g$ is
\[
\text{distiso}(f, g) = \min_{\pi \in S_n} \text{dist}(f, g^{\pi})
\]

For example, consider two parities
\[ f(x_1 \ldots x_n) = x_1 \oplus x_2 \ldots \oplus x_k \]
and
\[ g(x_1 \ldots x_n) = x_{100} \oplus \ldots \oplus x_{100+k'}. \]

Then
\begin{itemize}
  \item $k = k' \Rightarrow \text{distiso}(f, g) = 0.$
  \item $k \neq k' \Rightarrow \text{distiso}(f, g) = \frac{1}{2}.$
\end{itemize}
Testing function isomorphism

**Definition (restated)**

A property tester of isomorphism to a known function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is an *adaptive* algorithm $A$ with black box access to some $g : \{0, 1\}^n \rightarrow \{0, 1\}$ such that satisfies:

- $f \cong g \Rightarrow \Pr[A \text{ accepts}] \geq 2/3$.
- $\text{distiso}(f, g) \geq \epsilon \Rightarrow \Pr[A \text{ rejects}] \geq 2/3$,

where $\epsilon$ is a distance parameter.

**Goal:** minimize the number of queries to $g$.
We will think of $\epsilon$ as a *constant*. 
Analogous testing problems

The analogous of testing isomorphism between graphs is well-understood:

- [AFKS00] characterized graphs for which isomorphism is testable in $O(1)$.
- [FM08] gave tight bounds on the query complexity of testing graph isomorphism.
- [BC10] studied the question for uniform hypergraphs.
Some examples for function isomorphism testing

Many testing problems can be cast as testing isomorphism:

1. Testing if $g$ is a dictator, i.e. $g(x_1 x_2 \ldots x_n) = x_i$ for some $i \in [n]$. 
   
   Equivalent to testing isomorphism to $f(x_1 x_2 \ldots x_n) = x_1$.
   
   Takes $O(1)$ queries $[PRS02]$.

2. Testing if $g$ is a $k$-monomial. 
   
   Same as testing isomorphism to $f(x_1 x_2 \ldots x_k) = x_1 \wedge x_2 \ldots \wedge x_k$.
   
   Takes $O(1)$ queries too $[PRS02]$.

3. Testing if $g$ is a parity on $k$ variables ($k$-parity). 
   
   Same as isomorphism to $f(x_1 x_2 \ldots x_k) = x_1 \oplus x_2 \ldots \oplus x_k$. 
   
   Takes $O(1)$ queries $[PRS02]$. 

Some examples for function isomorphism testing

Many testing problems can be cast as testing isomorphism:

1. Testing if $g$ is a dictator, i.e. $g(x_1x_2 \ldots x_n) = x_i$ for some $i \in [n]$.
   Equivalent to testing isomorphism to $f(x_1x_2 \ldots x_n) = x_1$.
   Takes $O(1)$ queries [PRS02].
Some examples for function isomorphism testing

Many testing problems can be cast as testing isomorphism:

1. Testing if \( g \) is a dictator, i.e. \( g(x_1x_2\ldots x_n) = x_i \) for some \( i \in [n] \).
   Equivalent to testing isomorphism to \( f(x_1x_2\ldots x_n) = x_1 \).
   Takes \( O(1) \) queries [PRS02].

2. Testing if \( g \) is a \( k \)-monomial.
   Same as testing isomorphism to
   \( f(x_1x_2\ldots x_n) = x_1 \land x_2 \ldots \land x_k \).
   Takes \( O(1) \) queries too [PRS02].
Some examples for function isomorphism testing

Many testing problems can be cast as testing isomorphism:

1. Testing if $g$ is a dictator, i.e. $g(x_1 x_2 \ldots x_n) = x_i$ for some $i \in [n]$.
   Equivalent to testing isomorphism to $f(x_1 x_2 \ldots x_n) = x_1$.
   Takes $O(1)$ queries [PRS02].

2. Testing if $g$ is a $k$-monomial.
   Same as testing isomorphism to $f(x_1 x_2 \ldots x_n) = x_1 \land x_2 \ldots \land x_k$.
   Takes $O(1)$ queries too [PRS02].

3. Testing if $g$ is a parity on $k$ variables ($k$-parity).
   Same as isomorphism to $f(x_1 x_2 \ldots x_k) = x_1 \oplus x_2 \ldots \oplus x_k$. 
Driving questions

- How easy is to test isomorphism to a given function?
- What is the *query complexity* of testing isomorphism to the *worst* possible function $f$?
- Does the task become easier if $f$ enjoys some additional property? (e.g. if $f$ depends only on $k < n$ variables (*$k$-junta*)).
- Can we characterize the functions for which testing isomorphism to can be tested with constant number of queries?
Results from the recent past

Theorem (lower bound) [C-G.Soriano-Matsliah (SODA’11), Alon-Blais (RANDOM’10)]

There are functions $f : \{0, 1\}^n \to \{0, 1\}$ requiring $\Omega(n)$ queries to test isomorphism to (even for adaptive, two-sided algorithms).

Moreover, for any $k \leq n$ for most $k$-juntas $f : \{0, 1\}^n \to \{0, 1\}$ testing isomorphism to $f$ requires $\Omega(k)$ queries.
Results from the recent past

**Theorem (lower bound) [C-G.Soriano-Matsliah (SODA’11), Alon-Blais (RANDOM’10)]**

There are functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ requiring $\Omega(n)$ queries to test isomorphism to (even for adaptive, two-sided algorithms).

Moreover, for any $k \leq n$ for most $k$-juntas $f : \{0, 1\}^n \rightarrow \{0, 1\}$ testing isomorphism to $f$ requires $\Omega(k)$ queries.

**Theorem (upper bound) [CGM 2011, AB 2010]**

Isomorphism to any $k$-junta can be tested with $O(k \log k)$ queries.
Main Question: What are functions easy to test isomorphism to?

Proof.

Pick a random $k$ from $\sqrt{n}$ to $\sqrt{n}$.

Pick randomly a constant number of $x$'s of weight $k$ and query these $g(x)$'s.

If $g$ is $\epsilon$-far from being isomorphic to $f$ then you catch a witness whp.
Main Question: What are functions easy to test isomorphism to?

- \(O(1)\)-juntas. [Fischer et al, Alon-Blais-C-G.Soriano-Matsliah]
Main Question: What are functions easy to test isomorphism to?

- $O(1)$-juntas. [Fischer et al, Alon-Blais-C-G.Soriano-Matsliah]
- Symmetric function.

Proof.

Pick a random $k$ from $\frac{n}{2} \pm \sqrt{n}$. Pick randomly a constant number of $x$'s of weigh $k$ and query these $g(x)$'s. If $g$ is $\epsilon$-far from being isomorphic $f$ then you catch a witness whp.
What are functions easy to test isomorphism to?

- $O(1)$-juntas.
- Symmetric functions.
What are functions easy to test isomorphism to?

- $O(1)$-juntas.
- Symmetric functions.
- Functions with small isomorphisms.
The set of all distinct permutations of $f$ be
$\text{Isom}(f) = \{ f^{\pi} \mid \pi \in S_n \}$.

Observe that
- The function $f$ is symmetric if and only if $|\text{Isom}(f)| = 1$.
- A dictator $f(x) = x_1$ has $|\text{Isom}(f)| = n$.
- A $k$-junta satisfies $|\text{Isom}(f)| \leq \binom{n}{k} k! \leq n^k$. 

Hence $|\text{Isom}(f)|$ measures the "degree of symmetry" of $f$. 

$|\text{Isom}(f)|$ is also equal to the index of the automorphism group of $f$ in $S_n$. In fact $n$ is the smallest possible size of $\text{Isom}(f)$ for non-symmetric functions.
The set of all distinct permutations of $f$ be

$$\text{Isom}(f) = \{ f^{\pi} \mid \pi \in S_n \}.$$  

Observe that

- The function $f$ is symmetric if and only if $|\text{Isom}(f)| = 1$.
- A dictator $f(x) = x_1$ has $|\text{Isom}(f)| = n$.
- A $k$-junta satisfies $|\text{Isom}(f)| \leq \binom{n}{k} k! \leq n^k$.

Hence $|\text{Isom}(f)|$ measures the “degree of symmetry” of $f$. 
The set of all distinct permutations of $f$ be $\text{Isom}(f) = \{ f^\pi \mid \pi \in S_n \}$.

Observe that
- The function $f$ is symmetric if and only if $|\text{Isom}(f)| = 1$.
- A dictator $f(x) = x_1$ has $|\text{Isom}(f)| = n$.
- A $k$-junta satisfies $|\text{Isom}(f)| \leq \binom{n}{k} k! \leq n^k$.

Hence $|\text{Isom}(f)|$ measures the “degree of symmetry” of $f$.

$|\text{Isom}(f)|$ is also equal to the index of the automorphism group of $f$ in $S_n$. In fact $n$ is the smallest possible size of $\text{Isom}(f)$ for non-symmetric functions.
Some easy-to-test functions

**Observation**

\[ O(\log |\text{Isom}(f)|) \text{ queries are enough to test isomorphism to } f. \]
### Some easy-to-test functions

**Observation**

\[ O(\log |\text{Isom}(f)|) \] queries are enough to test isomorphism to \( f \).

For a \( k \)-junta \( f \) and \( k = O(1) \), \( \text{Isom}(f) \leq n^k = n^{O(1)} \). Yet we know that isomorphism to \( k \)-juntas can be tested with \( O(1) \) queries.

---

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Majority on ( n-1 )</td>
<td>( \text{Maj}_{n-1} ) similar to ( \text{Maj}_n ), can use trivial isomorphism tester.</td>
</tr>
<tr>
<td>Parity on ( n-1 )</td>
<td>( \chi_{n-1} = \chi_n \oplus x_n ). Queries for ( x_n ) can be translated into queries for ( \chi_{n-1} ). The problem turns into testing isomorphism to ( x_n ).</td>
</tr>
</tbody>
</table>

What do these two have in common?
Some easy-to-test functions

**Observation**

$O(\log |\text{Isom}(f)|)$ queries are enough to test isomorphism to $f$.

For a $k$-junta $f$ and $k = O(1)$, $\text{Isom}(f) \leq n^k = n^{O(1)}$. Yet we know that isomorphism to $k$-juntas can be tested with $O(1)$ queries.

Are there any other such functions?
Some easy-to-test functions

Observation

$O(\log |\text{Isom}(f)|)$ queries are enough to test isomorphism to $f$.

For a $k$-junta $f$ and $k = O(1)$, $\text{Isom}(f) \leq n^k = n^{O(1)}$. Yet we know that isomorphism to $k$-juntas can be tested with $O(1)$ queries.

Are there any other such functions?

- Majority on the first $n - 1$ variables $\text{Maj}_{n-1}$. This is very close to $\text{Maj}_n$, so we can use the trivial isomorphism tester for $\text{Maj}_n$.
Some easy-to-test functions

Observation

\( O(\log |\text{Isom}(f)|) \) queries are enough to test isomorphism to \( f \).

For a \( k \)-junta \( f \) and \( k = O(1) \), \( \text{Isom}(f) \leq n^k = n^{O(1)} \). Yet we know that isomorphism to \( k \)-juntas can be tested with \( O(1) \) queries.

Are there any other such functions?

- Majority on the first \( n - 1 \) variables \( \text{Maj}_{n-1} \). This is very close to \( \text{Maj}_n \), so we can use the trivial isomorphism tester for \( \text{Maj}_n \).
- Parity on the first \( n - 1 \) variables \( \chi_{n-1} \). This satisfies \( \chi_{n-1} = \chi_n \oplus x_n \). We can translate queries for the dictator \( x_n \) into queries for \( \chi_n \), and the problem turns into testing isomorphism to \( x_n \).
Some easy-to-test functions

**Observation**

\[ O(\log |\text{Isom}(f)|) \] queries are enough to test isomorphism to \( f \).

For a \( k \)-junta \( f \) and \( k = O(1) \), \( \text{Isom}(f) \leq n^k = n^{O(1)} \). Yet we know that isomorphism to \( k \)-juntas can be tested with \( O(1) \) queries.

Are there any other such functions?

- Majority on the first \( n - 1 \) variables \( \text{Maj}_{n-1} \). This is very close to \( \text{Maj}_n \), so we can use the trivial isomorphism tester for \( \text{Maj}_n \).
- Parity on the first \( n - 1 \) variables \( \chi_{n-1} \). This satisfies \( \chi_{n-1} = \chi_n \oplus x_n \). We can translate queries for the dictator \( x_n \) into queries for \( \chi_n \), and the problem turns into testing isomorphism to \( x_n \).

What do these two have in common?
A function $f : \{0,1\}^n \to \{0,1\}$ is called $k$-junto-symmetric if it can be written in the form

$$f(x) = \hat{f}(|x|, x\upharpoonright_J)$$

for some $\hat{f} : \{0, \ldots, n\} \times \{0,1\}^{|J|} \to \{0,1\}$ and $|J| = k$. 

Theorem ($O(1)$-junto-symmetric $\equiv$ poly-symmetric)

The following are equivalent:
(a) $|\text{Isom}(f)| = n$ ($f$ is poly-symmetric);
(b) $f$ is an $O(1)$-junto-symmetric;
(c) each $f^n$ is a boolean combination of $O(1)$-many dictators and $O(1)$-many symmetric functions;
Junto-symmetric functions

**Definition (Junto-Symmetric)**

A function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is called $k$-junto-symmetric if it can be written in the form

$$f(x) = \hat{f}(|x|, x|_J)$$

for some $\hat{f} : \{0, \ldots, n\} \times \{0, 1\}^{|J|} \rightarrow \{0, 1\}$ and $|J| = k$.

**Theorem ($O(1)$-junto-symmetric $\equiv$ poly-symmetric)**

The following are equivalent:

(a) $|\text{Isom}(f)| = n^{O(1)}$ ($f$ is poly-symmetric);

(b) $f$ is an $O(1)$-junto-symmetric;

(c) each $f_n$ is a boolean combination of $O(1)$-many dictators and $O(1)$-many symmetric functions;
Testing junto-symmetry

Theorem

[C-Fischer-G.Soriano-Matsliah (CCC’12)] There are \( \text{poly}(k/\epsilon) \) algorithms to test if \( f \) is \( k \)-junto-symmetric and to test isomorphism to \( k \)-junto-symmetric functions.
Theorem

[C-Fischer-G.Soriano-Matsliah (CCC’12)] There are $\text{poly}(k/\epsilon)$ algorithms to test if $f$ is $k$-junto-symmetric and to test isomorphism to $k$-junto-symmetric functions.

Theorem

[C-Fischer-G.Soriano-Matsliah (CCC’12)] There are $\text{poly}(k/\epsilon)$ algorithms to test if $f$ is “close” to $k$-junto-symmetric and to test isomorphism to functions that are “close” to $k$-junto-symmetric functions.
Testing junto-symmetry

Theorem

[C-Fischer-G.Soriano-Matsliah (CCC'12)] There are $\text{poly}(k/\epsilon)$ algorithms to test if $f$ is $k$-junto-symmetric and to test isomorphism to $k$-junto-symmetric functions.

Theorem

[C-Fischer-G.Soriano-Matsliah (CCC'12)] There are $\text{poly}(k/\epsilon)$ algorithms to test if $f$ is “close” to $k$-junto-symmetric and to test isomorphism to functions that are “close” to $k$-junto-symmetric functions.

Open:

isomorphism to $f$ can be tested with $O(1)$ queries

$\iff$

$f$ is close to $O(1)$-junto-symmetric?
Similar statement has been independently proved by Blais-Weinstein-Yoshida (FOCS'12).

**Theorem**

There are $\text{poly}(k/\epsilon)$ algorithms to test if $f$ is “close” to $k$-junto-symmetric and to test isomorphism to functions that are “close” to $k$-junto-symmetric functions.

**Open:**

isomorphism to $f$ can be tested with $O(1)$ queries

$\iff$

$f$ is close to $O(1)$-junto-symmetric?
How far we from a lower bound?

Conjecture

If $f$ is “far” from a $k$-junto-symmetric then testing isomorphism to $f$ requires $\log^* k$ queries.
Results from the recent past

Theorem (lower bound) [C-G.Soriano-Matsliah (SODA’11), Alon-Blais (RANDOM’10)]

There are functions $f : \{0, 1\}^n \to \{0, 1\}$ requiring $\Omega(n)$ queries to test isomorphism to (even for adaptive, two-sided algorithms).

Moreover, for any $k \leq n$ for most $k$-juntas $f : \{0, 1\}^n \to \{0, 1\}$ testing isomorphism to $f$ requires $\Omega(k)$ queries.

Theorem (upper bound) [CGM 2011, AB 2010]

Isomorphism to any $k$-junta can be tested with $O(k \log k)$ queries.
Pick $f, g$ to be two random functions from $\{0, 1\}^n \rightarrow \{0, 1\}$.
Ω(n) lower bound: First attempt

Pick $f, g$ to be two random functions from $\{0, 1\}^n \rightarrow \{0, 1\}$.

Make $f$ the known function. And with let the unknown function be $f$ with probability 1/2 and $g$ with probability 1/2.
Pick $f, g$ to be two random functions from $\{0, 1\}^n \rightarrow \{0, 1\}$.

Make $f$ the known function. And with let the unknown function be $f$ with probability $1/2$ and $g$ with probability $1/2$.

Prove that $f$ and $g$ are $\epsilon$-far.
Pick $f, g$ to be two random functions from $\{0, 1\}^n \rightarrow \{0, 1\}$.

Make $f$ the known function. And with let the unknown function be $f$ with probability $1/2$ and $g$ with probability $1/2$.

Prove that $f$ and $g$ are $\epsilon$-far.

Prove that any small set of queries cannot distinguish $f$ from $g$. 

Does NOT work: Since $f$ is known so the "light weight" queries reveal a lot and helps to distinguish $f$ from $g$. Infact $\sqrt{n}$ number of queries suffices.
Ω(n) lower bound: First attempt

Pick \( f, g \) to be two random functions from \( \{0, 1\}^n \rightarrow \{0, 1\} \).

Make \( f \) the known function. And with let the unknown function be \( f \) with probability \( 1/2 \) and \( g \) with probability \( 1/2 \).

Prove that \( f \) and \( g \) are \( \epsilon \)-far.

Prove that any small set of queries cannot distinguish \( f \) from \( g \).

Does NOT work: Since \( f \) is known so the “light weight” queries reveal a lot and helps to distinguish \( f \) from \( g \). Infact \( \sqrt{n} \) number of queries suffices.
We show there is $f : \{0, 1\}^n \rightarrow \{0, 1\}$ whose permutations look “almost random” to any tester making $o(n)$ queries. Our functions are non-zero only for *balanced* inputs ($x$ with $|x| \in [n/2 - 2\sqrt{n}, n/2 + 2\sqrt{n}]$).
We show there is $f : \{0, 1\}^n \rightarrow \{0, 1\}$ whose permutations look “almost random” to any tester making $o(n)$ queries. Our functions are non-zero only for balanced inputs ($x$ with $|x| \in [n/2 - 2\sqrt{n}, n/2 + 2\sqrt{n}]$).

**Definition**

$f$ is $q$-regular if for all sets $Q = \{x_1, \ldots, x_q\}$ of balanced queries and all assignments $a : \{0, 1\}^q \rightarrow \{0, 1\}$,

$$\Pr_{\pi}[f_{\pi}(x_1) = a_1 \land f_{\pi}(x_2) = a_2 \land \ldots \land f_{\pi}(x_q) = a_q] = (1 \pm 1/6)2^{-q}.$$
We show there is $f : \{0, 1\}^n \rightarrow \{0, 1\}$ whose permutations look “almost random” to any tester making $o(n)$ queries. Our functions are non-zero only for balanced inputs ($x$ with $|x| \in [n/2 - 2\sqrt{n}, n/2 + 2\sqrt{n}]$).

**Definition**

$f$ is $q$-regular if for all sets $Q = \{x_1, \ldots, x_q\}$ of balanced queries and all assignments $a : \{0, 1\}^q \rightarrow \{0, 1\}$,

$$\Pr_{\pi}[f^\pi(x_1) = a_1 \land f^\pi(x_2) = a_2 \land \ldots \land f^\pi(x_q) = a_q] = (1 \pm 1/6)2^{-q}.$$ 

- $f$ is $q$-regular $\Rightarrow$ more than $q$ queries are needed to test if $g \cong f$.
- We use the probabilistic method to prove the existence of $\Omega(n)$-regular functions.
- An $\Omega(k)$ lower bound for $k$-juntas follows by padding.
Existence of $q$-regular functions

**Definition**

$f$ is $q$-regular if for all sets $Q = \{x_1, \ldots, x_q\}$ of balanced queries and all assignments $a : \{0, 1\}^q \to \{0, 1\}$,

$$\Pr[\pi f(x_1) = a_1 \land \pi f(x_2) = a_2 \land \ldots \land \pi f(x_q) = a_q] = (1 \pm 1/6)2^{-q}.$$ 

Even if $f$ is a random function on the balanced queries, it is not obvious it is $q$-regular - since $Q$ and $\pi(Q)$ can intersect and hence the event that $\pi f(x_1) = a_1 \land f(x_2) = a_2 \land \ldots \land f(x_q) = a_q$ and the event that $f(x_1) = a_1 \land f(x_2) = a_2 \land \ldots \land f(x_q) = a_q$ are not independent.

So we have to calculate the probability in a different way - using ideas from [BC10].
Existence of $q$-regular functions

Let $N \triangleq \binom{n}{n/2-\lceil \sqrt{n} \rceil}$ and $X(g, \tau) = \mathbb{I}[g^\tau | Q = a]$.

Let $G$ be the permutation of variables subgroup of $\text{Sym}(\{0, 1\}^n)$.

We have to compute $\Pr_{\tau \in G}[X(f, \tau) = 1]$.
Existence of \( q \)-regular functions

Let \( N \triangleq \binom{n}{n/2 - \lceil \sqrt{n} \rceil} \) and \( X(g, \tau) = \mathbb{I}[g^\tau | Q = a] \).

Let \( G \) be the permutation of variables subgroup of \( \text{Sym}(\{0, 1\}^n) \).

We have to compute \( \Pr_{\tau \in G}[X(f, \tau) = 1] \).

**Lemma**

There exist \( s \triangleq \lceil N / q^2 \rceil \) permutations \( \sigma_1, \ldots, \sigma_s \in G \) such that \( \sigma_1 Q, \ldots, \sigma_s Q \) are disjoint.
Existence of $q$-regular functions

Let $N \triangleq \binom{n}{n/2-\lceil \sqrt{n} \rceil}$ and $X(g, \tau) = \mathbb{I}[g^\tau|_Q = a]$.
Let $G$ be the permutation of variables subgroup of $\text{Sym}([0, 1]^n)$.

We have to compute $\Pr_{\tau \in G}[X(f, \tau) = 1]$.

**Lemma**

There exist $s \triangleq \lceil N/q^2 \rceil$ permutations $\sigma_1, \ldots, \sigma_s \in G$ such that $\sigma_1 Q, \ldots, \sigma_s Q$ are disjoint.

$$
\Pr_{\tau \in G}[X(f, \tau) = 1] = \mathbb{E}_{i \in [s]} \mathbb{E}_{\tau \in G} X(f, \tau \circ \sigma_i) = \mathbb{E}_{\tau \in G} \mathbb{E}_{i \in [s]} X(f, \tau \circ \sigma_i).
$$
Existence of $q$-regular functions

Let $N \triangleq \left(\frac{n}{/2\left\lceil \sqrt{n} \right\rceil}\right)$ and $X(g, \tau) = \mathbb{I}[g^\tau | Q = a]$.

Let $G$ be the permutation of variables subgroup of $\text{Sym}(\{0, 1\}^n)$.

We have to compute $\Pr_{\tau \in G}[X(f, \tau) = 1]$.

Lemma

There exist $s \triangleq \left\lceil N/q^2 \right\rceil$ permutations $\sigma_1, \ldots, \sigma_s \in G$ such that $\sigma_1 Q, \ldots, \sigma_s Q$ are disjoint.

$$\Pr_{\tau \in G}[X(f, \tau) = 1] = \mathbb{E}_{i \in [s]} \mathbb{E}_{\tau \in G} X(f, \tau \circ \sigma_i) = \mathbb{E}_{\tau \in G} \mathbb{E}_{i \in [s]} X(f, \tau \circ \sigma_i).$$

Now $\mathbb{E}_{i \in [s]} X(f, \tau \circ \sigma_i)$ is close to its expectation with high probability [by Chernoff Bound]. And by union bound we show that a $q$-regular function exists.
Consider two $q$-regular functions $f, g : \{0, 1\}^k \to \{0, 1\}$ with $\text{dist}(f, g) \geq \epsilon$.

- Random permutations of $f$ and $g$ look random, so it is also hard to distinguish random $f^{\pi}$ from random $g^{\pi'}$.
- Pad $f, g$ to obtain functions $f', g' : \{0, 1\}^n \to \{0, 1\}$ by ignoring the last $n - k$ variables.
- One can show $\frac{\text{distiso}(f', g')}{2} \leq \text{distiso}(f, g) \leq \text{distiso}(f', g')$.

Hence an $\Omega(k)$ lower bound for $k$-juntas follows from padding.
Theorem (lower bound) [C-G.Soriano-Matsliah (SODA’11),
Alon-Blais (RANDOM’10)]

There are functions $f : \{0, 1\}^n \to \{0, 1\}$ requiring $\Omega(n)$ queries to
test isomorphism to (even for adaptive, two-sided algorithms).

Moreover, for any $k \leq n$ for most $k$-juntas $f : \{0, 1\}^n \to \{0, 1\}$
testing isomorphism to $f$ requires $\Omega(k)$ queries.
For some $f$, testing isomorphism against $f$ needs $\Omega(n)$ queries.

- The proof is non-constructive; a truncated random function works.
- Random functions are usually very complicated to describe.
How “complex” is the hard-to-test $f$?

For some $f$, testing isomorphism against $f$ needs $\Omega(n)$ queries.

- The proof is non-constructive; a truncated random function works.
- Random functions are usually very complicated to describe.
- However, $\text{poly}(n)$-wise independence suffices for the proof.
- By standard constructions of $\text{poly}(n)$-wise independent generators, we can put $f$ in $NC$.
- Likewise, $f$ can be taken to be a truncated low-degree polynomial over $\mathbb{F}_2$. 
Consequences of the lower bound

**Corollary**

Testing if a function can be computed by a circuit of size $s$ takes at least $\text{poly}(s)$ queries (for $s$ up to $\text{poly}(n)$).

**Proof.** Let $n = s^{1/c}$ ($c > 1$). $\exists$ $n$-regular $f : \{0, 1\}^n \rightarrow \{0, 1\}$ computable by circuits of size $s^c = n$. Any $f^\pi$ still has size $n$, but is indistinguishable with $o(s)$ queries from a random function, which need circuits of size $2^{\Omega(n)} \gg s$. 

\[
\text{Corollary} \\
\text{Testing if the Fourier degree of } f \text{ is } \leq d \text{ requires } \Omega(d) \text{ queries.} \\
\text{Proof.} \text{ Any } k\text{-junta is a degree- } k \text{ polynomial, whereas a random } f \text{ has degree } \Omega(n). \\
\text{This settles open questions by [DLM}^{+07}]. \]
Consequences of the lower bound

**Corollary**
Testing if a function can be computed by a circuit of size $s$ takes at least $\text{poly}(s)$ queries (for $s$ up to $\text{poly}(n)$).

**Proof.** Let $n = s^{1/c}$ ($c > 1$). $\exists n$-regular $f : \{0, 1\}^n \rightarrow \{0, 1\}$ computable by circuits of size $s^c = n$. Any $f^\pi$ still has size $n$, but is indistinguishable with $o(s)$ queries from a random function, which need circuits of size $2^{\Omega(n)} \gg s$.

**Corollary**
Testing if the Fourier degree of $f$ is $\leq d$ requires $\Omega(d)$ queries.

**Proof.** Any $k$-junta is a degree-$k$ polynomial, whereas a random $f$ has degree $\Omega(n)$.

This settles open questions by [DLM+07].
Results from the recent past

**Theorem (lower bound) [C-G. Soriano-Matsliah (SODA’11), Alon-Blais (RANDOM’10)]**

There are functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$ requiring $\Omega(n)$ queries to test isomorphism to (even for adaptive, two-sided algorithms).

Moreover, for any $k \leq n$ for most $k$-juntas $f : \{0, 1\}^n \rightarrow \{0, 1\}$ testing isomorphism to $f$ requires $\Omega(k)$ queries.

**Theorem (upper bound) [CGM 2011, AB 2010]**

Isomorphism to any $k$-junta can be tested with $O(k \log k)$ queries.
When $k = n$, there is a simple $O(n \log n)$ query algorithm:

1. Draw $O(\log n!) = O(n \log n)$ uniformly random samples and query $g$ on them.
2. Accept iff there is some $f^{\pi}$ consistent with all samples.
When $k = n$, there is a simple $O(n \log n)$ query algorithm:

1. Draw $O(\log n!) = O(n \log n)$ uniformly random samples and query $g$ on them.

2. Accept iff there is some $f^\pi$ consistent with all samples.

Suppose the known function $f$ is a $k$-junta.

- Assume $g$ is a $k$-junta too: $g(x_1 \ldots x_n) = g'(x_{i_1} \ldots x_{i_k})$; $g'$ is the core of the $k$-junta $g$.

- The simple upper bound would still need $\log(\binom{n}{k} k!) = O(k \log n) \gg k$. 

---

**$O(k \log k)$ upper bound for $k$-juntas**
When $k = n$, there is a simple $O(n \log n)$ query algorithm:

1. Draw $O(\log n!) = O(n \log n)$ uniformly random samples and query $g$ on them.
2. Accept iff there is some $f^\pi$ consistent with all samples.

Suppose the known function $f$ is a $k$-junta.

- Assume $g$ is a $k$-junta too: $g(x_1 \ldots x_n) = g'(x_{i_1} \ldots x_{i_k})$; $g'$ is the core of the $k$-junta $g$.
- The simple upper bound would still need 
  $$\log\left(\binom{n}{k} k!\right) = O(k \log n) \gg k.$$ 
- We would like to sample $g'$ rather than $g$.
- In general, we would need to draw samples of the core of the $k$-junta closest to $g$, but let us ignore this issue.
Noisy samplers

Let $\eta > 0$ and $g : \{0, 1\}^n \rightarrow \{0, 1\}$ be a $k$-junta with core $g' : \{0, 1\}^k \rightarrow \{0, 1\}$, i.e. $g(x_1 \ldots x_n) = g'(x_{i_1} x_{i_2} \ldots x_{i_k})$.

**Definition**

An $\eta$-noisy sampler for the core of $g$ is a black-box probabilistic algorithm $\mathcal{A}$ that on each execution outputs $(x, a) \in \{0, 1\}^k \rightarrow \{0, 1\}$ such that

1. The distribution of $x$ is uniform in $\{0, 1\}^k$.

2. $\Pr[g'(x) = a] \geq 1 - \eta$.

The probability is over the randomness of $\mathcal{A}$. 
Let $\eta > 0$ and $g : \{0, 1\}^n \to \{0, 1\}$ be a $k$-junta with core $g' : \{0, 1\}^k \to \{0, 1\}$, i.e. $g(x_1 \ldots x_n) = g'(x_{i_1} x_{i_2} \ldots x_{i_k})$.

**Definition**

An $\eta$-noisy sampler for the core of $g$ is a black-box probabilistic algorithm $\mathcal{A}$ that on each execution outputs $(x, a) \in \{0, 1\}^k \to \{0, 1\}$ such that

1. The distribution of $x$ is uniform in $\{0, 1\}^k$.
2. $\Pr[g'(x) = a] \geq 1 - \eta$.

The probability is over the randomness of $\mathcal{A}$.

- We don’t know the relevant vars $i_1, \ldots, i_k$ of $g$, and finding even one of them takes $\Omega(\log n) \gg k$ queries.
- Still we can draw good uniform samples from $g'$!
Let $\eta > 0$ and $g : \{0, 1\}^n \to \{0, 1\}$ be a $k$-junta with core $g' : \{0, 1\}^k \to \{0, 1\}$, i.e. $g(x_1 \ldots x_n) = g'(x_{i_1}x_{i_2} \ldots x_{i_k})$.

**Definition**

An $\eta$-noisy sampler for the core of $g$ is a black-box probabilistic algorithm $\mathcal{A}$ that on each execution outputs $(x, a) \in \{0, 1\}^k \to \{0, 1\}$ such that

1. The distribution of $x$ is uniform in $\{0, 1\}^k$.
2. $\Pr[g'(x) = a] \geq 1 - \eta$.

The probability is over the randomness of $\mathcal{A}$.

- We don’t know the relevant vars $i_1, \ldots, i_k$ of $g$, and finding even one of them takes $\Omega(\log n) \gg k$ queries.
- Still we can draw good uniform samples from $g'$!
Construction of noisy samplers

**Theorem**

It is possible to construct a 0.1-noisy sampler for the core of a $k$-junta $g$. The sampler makes *one* query to $g$ on each execution, after $O(k \log k)$ preprocessing queries.

This allows us to test isomorphism to $k$-juntas in $O(k \log k + \log k!) = O(k \log k)$ queries.
Construction of noisy samplers

Theorem

It is possible to construct a $0.1$-noisy sampler for the core of a $k$-junta $g$. The sampler makes one query to $g$ on each execution, after $O(k \log k)$ preprocessing queries.

This allows us to test isomorphism to $k$-juntas in $O(k \log k + \log k!) = O(k \log k)$ queries.

- The algorithm builds on the $O(k \log k)$ junta tester of Blais.
- It starts by picking at random a partition $\mathcal{P}$ of $[n]$ into $k^{2+O(1)}$ blocks and finding the $k$-relevant blocks.
Construction of noisy samplers

**Theorem**

It is possible to construct a 0.1-noisy sampler for the core of a $k$-junta $g$. The sampler makes one query to $g$ on each execution, after $O(k \log k)$ preprocessing queries.

This allows us to test isomorphism to $k$-juntas in $O(k \log k + \log k!) = O(k \log k)$ queries.

- The algorithm builds on the $O(k \log k)$ junta tester of Blais.
- It starts by picking at random a partition $\mathcal{P}$ of $[n]$ into $k^{2+O(1)}$ blocks and finding the $k$-relevant blocks.
- For each sample we make one query that is constant inside each block.
- These queries are highly non-uniform for any given $\mathcal{P}$.
- Even so, for most partitions $\mathcal{P}$ this yields a noisy sampler.
## Summary

<table>
<thead>
<tr>
<th>testing problem</th>
<th>prior work</th>
<th>this work</th>
</tr>
</thead>
<tbody>
<tr>
<td>isom. to ( k )-juntas</td>
<td>( \Omega(\log k) ) \cite{FKR+02, BO10, AB10}</td>
<td>( \Omega(k) ) ( O(k \log k) )</td>
</tr>
<tr>
<td></td>
<td>( \tilde{O}(k^4) ) \cite{FKR+02, DLM+07}</td>
<td></td>
</tr>
<tr>
<td>isom. to ( k )-juntas, 1-sided error</td>
<td>( \Omega(\log \log n) ) \cite{FKR+02}</td>
<td>( \Omega(k \log (n/k)) ) ( O(k \log n) )</td>
</tr>
<tr>
<td>circuits of size ( s )</td>
<td>( \tilde{\Omega}(\log s) ) \cite{DLM+07}</td>
<td>( s^{\Omega(1)} )</td>
</tr>
<tr>
<td></td>
<td>( \tilde{O}(s^6) ) \cite{DLM+07}</td>
<td></td>
</tr>
<tr>
<td>Fourier degree ( \leq d )</td>
<td>( \Omega(\log d) ) \cite{DLM+07}</td>
<td>( \Omega(d) )</td>
</tr>
<tr>
<td></td>
<td>( 2^{O(d)} ) \cite{DLM+07}</td>
<td></td>
</tr>
<tr>
<td>isom. between unknown functions</td>
<td>( \Omega(2^{n/2}/n^{1/4}) ) \cite{AB10}</td>
<td>( \Omega(2^{n/2}/n^{1/4}) ) ( O(\sqrt{2^n n \log n}) ) \cite{AB10}</td>
</tr>
<tr>
<td></td>
<td>( O(\sqrt{2^n n \log n}) ) \cite{AB10}</td>
<td></td>
</tr>
</tbody>
</table>

**Table:** Summary of results
Noga Alon and Eric Blais.
Testing boolean function isomorphism.

Noga Alon, Eldar Fischer, Michael Krivelevich, and Mario Szegedy.
Efficient testing of large graphs.
10.1007/s004930070001.

Laszlo Babai and Sourav Chakraborty.
Property testing of equivalence under a permutation group action.
*To appear in the ACM Transactions on Computation Theory (ToCT)*, 2010.

Eric Blais and Ryan O’Donnell.
Lower bounds for testing function isomorphism.

