Outline

1. Introduction
   - Property testing
   - Results

2. Linearity testing
   - Definitions
   - The algorithm

3. Symmetry testing
   - Definitions
   - The algorithm

4. Other results and further work
   - Homomorphism testing
   - Quasi-symmetry testing
   - Further work
Outline

1 Introduction
   • Property testing
     • Results

2 Linearity testing
   • Definitions
   • The algorithm

3 Symmetry testing
   • Definitions
   • The algorithm

4 Other results and further work
   • Homomorphism testing
   • Quasi-symmetry testing
   • Further work
We wish to know if some black-box function has a particular property or is far from having it.
We wish to know if some black-box function has a particular property or is far from having it. We want to do so by making as few queries as possible to the function.
For example

Does $f$ output The Twelve Days of Christmas?

On the first day of Christmas, my true love gave to me...
   A Partridge in a Pear Tree.
On the second day of Christmas, my true love gave to me...
   2 Turtle Doves
   And a Partridge in a Pear Tree.
On the third day of Christmas, my true love gave to me...
   3 French Hens
   2 Turtle Doves
   And a Partridge in a Pear Tree.
Obfuscated code

```c
#include <stdio.h>
main(t,_,a)
char * a;
{
  return !
0<t?
t<3?
main(−79,−13,a+)
main(−87,1−,main(−86,0,a+1)+a))::1,t<−?
main(t+1,_,a):3,
main(−94,−27+t,a)
&&t == 2 ? _<13 ?
main(2,−1, "%s",%d,%d

n"
):
9:16:t<0?t<−72?
main( _, t,
"@n+,#/\* {w+/w#cdnr/+.,} r/*de++,*{*,/w{%/w#q#n+/,#{l,+/,n{n+n,+##n+,/#:q#n+,[+,k#:++,"r.:d*3,} {w+KwK:'++} e:'#;dq:'#l q#)+d'K!/+k#;
q #'r}eKK#}w'r)eKK{nl}'/#;#q#n'){)w'}{)nl}'/+#n';d}rw'−i;#−}{nl}!/n{n#'−
rl'w'nc{nl}/#{l,+'}K{rw'−iK{[{nl}']/w#q#
\n n'wk_nw'−iwk{KK{nl}!/%l##w#'−i;−:nl}'*/{q#'ld;r'}{nlwb!/*de} 'c−;\n{nl'}−{}rw]'/+.,}##'*}#nc,',#nw]'/+kd'++e}+;\n#rdq#w!−nr'/−')−}+){rl#'{n'−'}#−'}+}##(!/" ::
t<−50?==*a ?
putchar(31[a]);
main(−65,−,a+1):
main((*a == '/') + t, _, a + 1 )::0<t?
main(2,2 ,"%s"):*a=='/ ||
main(0,
```

Philippe Lamontagne, Gilles Brassard, Alain Tapp

Property testing in a quantum world
We define the distance between two functions $f, g : A \to B$ by

$$d(f, g) = \frac{|\{x \in A \mid f(x) \neq g(x)\}|}{|A|}$$
We define the distance between two functions $f, g : A \to B$ by

$$d(f, g) = \frac{|\{x \in A \mid f(x) \neq g(x)\}|}{|A|}$$

For a set of functions $L$, we say that $f$ is $\epsilon$-far from $L$ if $d(f, g) \geq \epsilon$ for all $g \in L$. 
The task at hand is then, for property $L$, black-box function $f$ and distance parameter $\epsilon$,

- to accept $f$ if $f \in L$ with probability at least $2/3$ and
- to reject $f$ if it is $\epsilon$-far from $L$ with probability at least $2/3$. 
Outline

1 Introduction
   - Property testing
   - Results

2 Linearity testing
   - Definitions
   - The algorithm

3 Symmetry testing
   - Definitions
   - The algorithm

4 Other results and further work
   - Homomorphism testing
   - Quasi-symmetry testing
   - Further work
We provide algorithms achieving a quadratic gain over the best classical testers for 4 properties:

- linearity of Boolean functions
- linearity of mappings between abelian groups (homomorphism)
- symmetry (permutation invariance) of Boolean functions
- quasi-symmetry* of Boolean functions
Our results

We provide algorithms achieving a quadratic gain over the best classical testers for 4 properties:

- linearity of Boolean functions
- linearity of mappings between abelian groups (homomorphism)
- symmetry (permutation invariance) of Boolean functions
- quasi-symmetry* of Boolean functions

The previously best known quantum algorithms for testing linearity and symmetry of Boolean functions by Hillery and Andersson achieved $O((1/\epsilon)^{2/3})$ queries. Better than the classical $O(1/\epsilon)$. 
Our results

We provide algorithms achieving a quadratic gain over the best classical testers for 4 properties:

- linearity of Boolean functions
- linearity of mappings between abelian groups (homomorphism)
- symmetry (permutation invariance) of Boolean functions
- quasi-symmetry* of Boolean functions

The previously best known quantum algorithms for testing linearity and symmetry of Boolean functions by Hillery and Andersson achieved $O((1/\epsilon)^{2/3})$ queries. Better than the classical $O(1/\epsilon)$. We improve on their algorithms to achieve $O(\sqrt{1/\epsilon})$ queries.
Our results

We provide algorithms achieving a quadratic gain over the best classical testers for 4 properties:

- linearity of Boolean functions
- linearity of mappings between abelian groups (homomorphism)
- symmetry (permutation invariance) of Boolean functions
- quasi-symmetry* of Boolean functions

The previously best known quantum algorithms for testing linearity and symmetry of Boolean functions by Hillery and Andersson achieved $O((1/\epsilon)^{2/3})$ queries. Better than the classical $O(1/\epsilon)$. We improve on their algorithms to achieve $O(\sqrt{1/\epsilon})$ queries.

* $O(n/\sqrt{\epsilon})$ for quasi-symmetry
Outline

1 Introduction
   • Property testing
   • Results

2 Linearity testing
   • Definitions
   • The algorithm

3 Symmetry testing
   • Definitions
   • The algorithm

4 Other results and further work
   • Homomorphism testing
   • Quasi-symmetry testing
   • Further work
We say a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) is linear if
\[
f(x \oplus y) = f(x) \oplus f(y)
\]
where \( \oplus \) is the exclusive-or (addition modulo 2).
We say a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is linear if

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

where $\oplus$ is the exclusive-or (addition modulo 2).

**Lemma**

$f$ is linear if there exists $s \in \{0, 1\}^n$ such that $f(x) = s \cdot x$ where

$$s \cdot x = \bigoplus_{i=1}^{n} s_i x_i.$$
The function $f$ is implemented by the quantum operator $S_f$ acting as

$$S_f |x\rangle = (-1)^{f(x)} |x\rangle$$

for computational basis state $|x\rangle$. 
Outline

1 Introduction
   • Property testing
   • Results

2 Linearity testing
   • Definitions
   • The algorithm

3 Symmetry testing
   • Definitions
   • The algorithm

4 Other results and further work
   • Homomorphism testing
   • Quasi-symmetry testing
   • Further work
Overview

We start by finding a candidate function $g$ such that

1. $g$ is linear and
2. if $f$ is linear, then $f = g$. 
Overview

We start by finding a candidate function $g$ such that

1. $g$ is linear and
2. if $f$ is linear, then $f = g$.

We then test the equality of $f$ and $g$ by searching for an input on which they disagree.
In order to define $g$, define the state

$$|v_f\rangle = S_f H^\otimes n |0\rangle = \frac{1}{\sqrt{N}} \sum_x (-1)^{f(x)} |x\rangle.$$
The Bernstein-Vazirani algorithm

In order to define \( g \), define the state

\[
|v_f\rangle = S_f H^\otimes n |0\rangle = \frac{1}{\sqrt{N}} \sum_x (-1)^{f(x)} |x\rangle.
\]

If \( f \) is linear, then \( f(x) = s \cdot x \) and this state is just

\[
\frac{1}{\sqrt{N}} \sum_x (-1)^{s \cdot x} |x\rangle = H^\otimes n |s\rangle.
\]

Applying the self-inverting transformation \( H^\otimes n \) and measuring yields \( s \). This is the Bernstein-Vazirani algorithm.
Defining $g$

Let $s$ be the string obtained by applying the Bernstein-Vazirani algorithm on $f$. 
Defining $g$

Let $s$ be the string obtained by applying the Bernstein-Vazirani algorithm on $f$.

- Our candidate function is then $g(x) = s \cdot x$. 
Defining $g$

Let $s$ be the string obtained by applying the Bernstein-Vazirani algorithm on $f$.

- Our candidate function is then $g(x) = s \cdot x$.
- If $f$ is linear, then $f(x) = s \cdot x = g(x)$. 
Defining $g$

Let $s$ be the string obtained by applying the Bernstein-Vazirani algorithm on $f$.

- Our candidate function is then $g(x) = s \cdot x$.
- If $f$ is linear, then $f(x) = s \cdot x = g(x)$.
- If $f$ is $\epsilon$-far from linear, then $d(f, g) \geq \epsilon$. 
Let $s$ be the string obtained by applying the Bernstein-Vazirani algorithm on $f$.

- Our candidate function is then $g(x) = s \cdot x$.
- If $f$ is linear, then $f(x) = s \cdot x = g(x)$.
- If $f$ is $\epsilon$-far from linear, then $d(f, g) \geq \epsilon$.

This is done using only one query to $f$. 
Search for inequality

We now need to search for an input $x$ such that $f(x) \neq g(x)$. 
Search for inequality

We now need to search for an input $x$ such that $f(x) \neq g(x)$. We define a Boolean function $\chi(x) = |f(x) - g(x)|$ and we apply the BBHTT algorithm to speed up the search for a solution to $\chi$. 
BBHT’s algorithm consists of applying a random number of times $j$ the Grover operator $G$. Increase exponentially the the number of possible values of $j$ until a solution is found.

Lemma

Let $t$ be the number of solutions to $\chi$ and $M$ be an arbitrary positive integer. Pick $j$ uniformly at random between $0$ and $M-1$. Applying $G^j$ to initial state $H|0\rangle$ and measuring yields a solution with probability at least $\frac{1}{4}$ when $M \geq \sqrt{2n/t}$.
BBHT’s algorithm consists of applying a random number of times $j$ the Grover operator $G$. Increase exponentially the the number of possible values of $j$ until a solution is found.

**Lemma**

Let $t$ be the number of solutions to $\chi$ and $M$ be an arbitrary positive integer. Pick $j$ uniformly at random between 0 and $M - 1$. Applying $G^j$ to initial state $H|0\rangle$ and measuring yields a solution with probability at least $1/4$ when $M \geq \sqrt{2^n}/t$. 
The search

We know either $t = 0$ or $t \geq \epsilon 2^n$. 
We know either $t = 0$ or $t \geq \epsilon 2^n$. 
Taking $M = \lceil \sqrt{1/\epsilon} \rceil \geq \sqrt{2^n/t}$ and applying $j \in \mathbb{R} \{0, \ldots, M - 1\}$ times the Grover operator yields a good solution with probability at least $1/4$. 

Repeat 4 times. The probability of not finding a solution if $f$ is $\epsilon$-far from linear is less than $(3/4)^4 < 1/3$. 

Philippe Lamontagne, Gilles Brassard, Alain Tapp
We know either $t = 0$ or $t \geq \epsilon 2^n$. Taking $M = \lceil \sqrt{1/\epsilon} \rceil \geq \sqrt{2^n/t}$ and applying $j \in \mathbb{R} \{0, \ldots, M-1\}$ times the Grover operator yields a good solution with probability at least $1/4$.

Repeat 4 times. The probability of not finding a solution if $f$ is $\epsilon$-far from linear is less than $(3/4)^4 < 1/3$. 
Finally

- We found a candidate $g$ for $f$.
- If we found $x$ such that $f(x) \neq g(x)$, declare $f$ $\epsilon$-far from linear.
- If no such $x$ was found, declare $f$ linear.
Finally

- We found a candidate $g$ for $f$.
- If we found $x$ such that $f(x) \neq g(x)$, declare $f$ $\epsilon$-far from linear.
- If no such $x$ was found, declare $f$ linear.

This has one-sided error of at most $1/3$ and makes $O(\sqrt{1/\epsilon})$ queries to $f$. 
Symmetry

A function is symmetric if it depends only on the Hamming weight of its input.
Symmetry

A function is symmetric if it depends only on the Hamming weight of its input. Symmetric functions are also called permutation invariant because they are invariant under the permutation of their input bits.
In order to test symmetry, we define the symmetric subspace $S$ spanned by the vectors of the form

$$|00\ldots0\rangle, \frac{1}{2^{n/2}}(|10\ldots0\rangle + |01\ldots0\rangle + \cdots + |00\ldots1\rangle), \text{ etc.}$$
The symmetric subspace

In order to test symmetry, we define the symmetric subspace $S$ spanned by the vectors of the form

$$|00 \ldots 0\rangle, \frac{1}{2^{n/2}}(|10 \ldots 0\rangle + |01 \ldots 0\rangle + \cdots + |00 \ldots 1\rangle), \text{etc.}$$

If $f$ is symmetric, then the vector

$$|v_f\rangle = \frac{1}{\sqrt{N}} \sum_x (-1)^{f(x)} |x\rangle$$

completely lies in $S$. 
The orthogonal complement

Let \( S^\perp \) be the orthogonal complement of \( S \).
If \( f \) is \( \epsilon \)-far from symmetric, then it can be shown that the following inequality holds

\[
\langle v_f | P_{S^\perp} | v_f \rangle \geq 2\epsilon.
\]
Let $S^\perp$ be the orthogonal complement of $S$. If $f$ is $\epsilon$-far from symmetric, then it can be shown that the following inequality holds

$$\langle v_f | P_{S^\perp} | v_f \rangle \geq 2\epsilon.$$ 

We will want to measure a component of $|v_f\rangle$ in $S^\perp$. 
Outline

1 Introduction
   - Property testing
   - Results

2 Linearity testing
   - Definitions
   - The algorithm

3 Symmetry testing
   - Definitions
   - The algorithm

4 Other results and further work
   - Homomorphism testing
   - Quasi-symmetry testing
   - Further work
Amplitude amplification, a generalization of BBHT, can increase the probability of measuring a component of $|v_f\rangle$ in $S^\perp$. 
Amplitude amplification, a generalization of BBHT, can increase the probability of measuring a component of $|v_f\rangle$ in $S^\perp$. With the right substitutions in the Grover iteration $G$ and by measuring with respect to projector $P_{S^\perp}$, the preceding lemma still holds.
Amplitude amplification, a generalization of BBHT, can increase the probability of measuring a component of $|v_f\rangle$ in $S^\perp$. With the right substitutions in the Grover iteration $G$ and by measuring with respect to projector $P_{S^\perp}$, the preceding lemma still holds.

**Lemma**

*Let $a$ be the probability of obtaining 1 when measuring $P_{S^\perp}$ and $M$ be an arbitrary positive integer. Pick $j$ uniformly at random between 0 and $M - 1$. Applying $G^j$ to initial state $S_f H |0\rangle$ and measuring finds a component in $S^\perp$ with probability at least $1/4$ when $M \geq \sqrt{1/a}$.*
The search

Set $M = \lceil 1/2\epsilon \rceil \geq \sqrt{1/a}$. Apply $j \in \mathbb{R} \{0, \ldots, M - 1\}$ times the modified operator $G$ and measure $P_{S\perp}$. 
The search

Set $M = \lceil 1/2\epsilon \rceil \geq \sqrt{1/a}$. Apply $j \in_R \{0, \ldots, M - 1\}$ times the modified operator $G$ and measure $P_{S^\perp}$.

Repeat 4 times The probability of not measuring a component of $|v_f\rangle$ in $S^\perp$ if $f$ is $\epsilon$-far from symmetric is at most $(3/4)^4 < 1/3$. 
Finally

- If no component in $S^\perp$ was measured, declare $f$ symmetric.
- If a component in $S^\perp$ was measured, declare $f$ $\epsilon$-far from symmetric.

This algorithm has one-sided error of at most $1/3$ and makes $O(\sqrt{1/\epsilon})$ queries to $f$. 
Outline

1 Introduction
   - Property testing
   - Results

2 Linearity testing
   - Definitions
   - The algorithm

3 Symmetry testing
   - Definitions
   - The algorithm

4 Other results and further work
   - Homomorphism testing
   - Quasi-symmetry testing
   - Further work
Homomorphism testing

We test if a function between two abelian groups is linear. In other words, let $G, H$ be two abelian groups. We test wether $f : G \rightarrow H$ is a homomorphism or is $\epsilon$-far from being a homomorphism.
We test if a function between two abelian groups is linear. In other words, let $G, H$ be two abelian groups. We test whether $f : G \rightarrow H$ is a homomorphism or is $\epsilon$-far from being a homomorphism.

- Generalization of Bernstein-Vazirani by Høyer.
- Same search procedure with adapted function $\chi$. 

Philippe Lamontagne, Gilles Brassard, Alain Tapp

Property testing in a quantum world
We test if a function between two abelian groups is linear. In other words, let $G, H$ be two abelian groups. We test whether $f : G \rightarrow H$ is a homomorphism or is $\epsilon$-far from being a homomorphism.

- Generalization of Bernstein-Vazirani by Høyer.
- Same search procedure with adapted function $\chi$.

We do so with $O(\sqrt{1/\epsilon})$. 

Philippe Lamontagne, Gilles Brassard, Alain Tapp
A Boolean function $f$ is quasi-symmetric if it is symmetric on the bits on which it actually depends.
Quasi-symmetry testing

A Boolean function $f$ is quasi-symmetric if it is symmetric on the bits on which it actually depends.

- *Estimate* the bits on which $f$ depends.
- Apply our symmetry tester.
A Boolean function $f$ is quasi-symmetric if it is symmetric on the bits on which it actually depends.

- Estimate the bits on which $f$ depends.
- Apply our symmetry tester.

The first step induces a dependance on $n$, the number of input bits in the number of queries made. Our algorithm makes $O(n/\sqrt{\epsilon})$ queries to $f$. 
Outline

1. Introduction
   - Property testing
   - Results

2. Linearity testing
   - Definitions
   - The algorithm

3. Symmetry testing
   - Definitions
   - The algorithm

4. Other results and further work
   - Homomorphism testing
   - Quasi-symmetry testing
   - Further work
Further work

We are currently working on:

- eliminating the dependance on $n$,
- getting the article published.
Thank you!