A Separation Between QNC^0 and AC^0

Adam Bene Watts, Robin Kothari, Luke Schaeffer, Avishay Tal

January 30, 2019

Coauthors

Adam Bene Watts

Robin Kothari

Avishay Tal

Section 1

Introduction

Quantum Advantage

Broad Goal

Prove quantum computers are more powerful than classical computers.

Quantum Advantage

Broad Goal

Prove *unconditionally* that quantum computers are more powerful than classical computers.

Quantum Advantage

Broad Goal

Prove *unconditionally* that quantum computers are more powerful than classical computers.

Quantum Advantage

Broad Goal

Prove *unconditionally* that quantum computers are more powerful than classical computers.

Previous work:

- Shor's algorithm.
  - What if factoring is easy classically too?

Quantum Advantage

Broad Goal

Prove *unconditionally* that quantum computers are more powerful than classical computers.

Previous work:

- Shor's algorithm.
  - What if factoring is easy classically too?
- Boson Sampling. Several hardness assumptions.

Quantum Advantage

Broad Goal

Prove *unconditionally* that quantum computers are more powerful than classical computers.

Previous work:

- Shor's algorithm.
  - What if factoring is easy classically too?
- Boson Sampling. Several hardness assumptions.
- Grover's algorithm.
  - $O(\sqrt{N})$ oracle calls. How do you implement the oracle?

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

Features:

- ✓ Completely unconditional!

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

Features:

- ✓ Completely unconditional!
- ✓ Fair comparison

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

Features:

- ✓ Completely unconditional!
- ✓ Fair comparison
- ✓ Simple quantum circuit
  - A variant (2DHLF) uses only local gates on a 2D grid

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

Features:

- ✓ Completely unconditional!
- ✓ Fair comparison
- ✓ Simple quantum circuit
  - A variant (2DHLF) uses only local gates on a 2D grid
- ✗ Doesn't use any complexity theory

"Quantum Advantage with Shallow Circuits"

Theorem (Bravyi, Gosset, König)

The hidden linear function (HLF) problem can be solved by constant depth quantum circuits but not constant depth classical circuits.

Features:

- ✓ Completely unconditional!
- ✓ Fair comparison
- ✓ Simple quantum circuit
  - A variant (2DHLF) uses only local gates on a 2D grid
- ✗ Doesn't use any complexity theory

Open Problem

Can we improve this result using ideas from circuit complexity?

Relevant Circuit Classes

"The class of problems solved by constant depth **classical/quantum** circuits (of poly size) with **constant/unbounded** fan-in gates."

	Constant Fan-In	Unbounded Fan-In
Classical	NC^0	AC^0
Quantum	QNC^0	QAC^0?

Relevant Circuit Classes

"The class of problems solved by constant depth **classical/quantum** circuits (of poly size) with **constant/unbounded** fan-in gates."

	Constant Fan-In	Unbounded Fan-In
Classical	NC^0	AC^0
Quantum	QNC^0	QAC^0?

Technicality

Actually, these classes (NC^0, QNC^0, AC^0) are for *decision problems* with 1 bit of output. This talk is about *relation problems* with multiple bits of output and multiple answers.

Relevant Circuit Classes

"The class of problems solved by constant depth **classical/quantum** circuits (of poly size) with **constant/unbounded** fan-in gates."

	Constant Fan-In	Unbounded Fan-In
Classical	NC^0	AC^0
Quantum	QNC^0	QAC^0?

Technicality

Actually, these classes (NC^0, QNC^0, AC^0) are for *decision problems* with 1 bit of output.
This talk is about *relation problems* with multiple bits of output and multiple answers.
And this is necessary!

Relevant Circuit Classes

"The class of problems solved by constant depth **classical/quantum** circuits (of poly size) with **constant/unbounded** fan-in gates."

	Constant Fan-In	Unbounded Fan-In
Classical	NC^0	AC^0
Quantum	QNC^0	QAC^0?

Theorem (BGK Result)

The Hidden Linear Function Problem (HLF) is in QNC^0 *but not* NC^0.

Relevant Circuit Classes

"The class of problems solved by constant depth **classical/quantum** circuits (of poly size) with **constant/unbounded** fan-in gates."

	Constant Fan-In	Unbounded Fan-In
Classical	NC^0	AC^0
Quantum	QNC^0	QAC^0?

Theorem (BGK Result)

The Hidden Linear Function Problem (HLF) is in QNC^0 *but not* NC^0.

Theorem (Our Result)

The Relaxed Parity Halving Problem (RPHP) is in QNC^0 *but not* AC^0.

Outline

Main Result

Extensions

Outline

Main Result

- Parity Halving Problem (separate QNC^0/qpoly and AC^0)

- Relaxed Parity Halving Problem (separate QNC^0 and AC^0)

Extensions

Outline

Main Result

- Parity Halving Problem (separate $\mathrm{QNC}^0/\mathrm{qpoly}$ and AC^0)
  - Quantum circuit with advice ($\mathrm{PHP} \in \mathrm{QNC}^0/\mathrm{qpoly}$)
  - Classical hardness ($\mathrm{PHP} \notin \mathrm{AC}^0$)

- Relaxed Parity Halving Problem (separate QNC^0 and AC^0)

Extensions

Outline

Main Result

- Parity Halving Problem (separate $\mathrm{QNC}^0/\mathrm{qpoly}$ and AC^0)
  - Quantum circuit with advice ($\mathrm{PHP} \in \mathrm{QNC}^0/\mathrm{qpoly}$)
  - Classical hardness ($\mathrm{PHP} \notin \mathrm{AC}^0$)
    - Hard as a game,
    - Hard against NC^0 (via locality),
    - Hard against AC^0 (via Switching Lemma).
- Relaxed Parity Halving Problem (separate QNC^0 and AC^0)

Extensions

Outline

Main Result

- Parity Halving Problem (separate $\mathrm{QNC}^0/\mathrm{qpoly}$ and AC^0)
  - Quantum circuit with advice ($\mathrm{PHP} \in \mathrm{QNC}^0/\mathrm{qpoly}$)
  - Classical hardness ($\mathrm{PHP} \notin \mathrm{AC}^0$)
    - Hard as a game,
    - Hard against NC^0 (via locality),
    - Hard against AC^0 (via Switching Lemma).
- Relaxed Parity Halving Problem (separate QNC^0 and AC^0)
  - Quantum algorithm ($\mathrm{RPHP} \in \mathrm{QNC}^0$)
  - Classical hardness ($\mathrm{RPHP} \notin \mathrm{AC}^0$)

Extensions

Outline

Main Result

- Parity Halving Problem (separate $\mathsf{QNC}^0/\mathsf{qpoly}$ and AC^0)
  - Quantum circuit with advice ($\mathsf{PHP} \in \mathsf{QNC}^0/\mathsf{qpoly}$)
  - Classical hardness ($\mathsf{PHP} \notin \mathsf{AC}^0$)
    - Hard as a game,
    - Hard against NC^0 (via locality),
    - Hard against AC^0 (via Switching Lemma).
- Relaxed Parity Halving Problem (separate QNC^0 and AC^0)
  - Quantum algorithm ($\mathsf{RPHP} \in \mathsf{QNC}^0$)
  - Classical hardness ($\mathsf{RPHP} \notin \mathsf{AC}^0$)

Extensions

- Better parameters, Geometric locality, Relation to HLF

Section 2

Parity Halving Problem

Parity Halving

Notation: For $x \in \{0,1\}^n$ define the *Hamming weight* $|x| := \sum_i x_i$.

Parity Halving

Notation: For $x \in \{0,1\}^n$ define the *Hamming weight* $|x| := \sum_i x_i$.

Parity Halving Problem

Given $x \in \{0,1\}^n$ with even Hamming weight ($|x| \equiv 0 \pmod 2$), output $y \in \{0,1\}^n$ such that

$$|y| \equiv \frac{1}{2}|x| \pmod 2$$

Parity Halving

Notation: For $x \in \{0,1\}^n$ define the *Hamming weight* $|x| := \sum_i x_i$.

Parity Halving Problem

Given $x \in \{0,1\}^n$ with even Hamming weight ($|x| \equiv 0 \pmod 2$), output $y \in \{0,1\}^n$ such that

$$|y| \equiv \frac{1}{2}|x| \pmod 2$$

Example ($n = 3$)

$$000 \to 000, 011, 101, 110 \text{ (even)}$$
$$011 \to 001, 010, 100, 111 \text{ (odd)}$$
$$101 \to 001, 010, 100, 111 \text{ (odd)}$$
$$110 \to 001, 010, 100, 111 \text{ (odd)}$$

Parity Halving Problem

Given $x \in \{0,1\}^n$ such that $|x| \equiv 0 \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv 0 \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv 2 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

Parity Halving Game

Nonlocal n player game:

- each player gets one input bit x_j,
- responsible for one output bit y_j.

The players win if

$$|y| \equiv \frac{1}{2}|x| \pmod 2$$

Parity Halving Game

Nonlocal n player game:

- each player gets one input bit x_j,
- responsible for one output bit y_j.

The players win if

$$|y| \equiv \frac{1}{2}|x| \pmod 2$$

- Special case $n = 3$ is the GHZ game.

Parity Halving Game

Nonlocal n player game:

- each player gets one input bit x_j,
- responsible for one output bit y_j.

The players win if

$$|y| \equiv \frac{1}{2}|x| \pmod 2$$

- Special case $n = 3$ is the GHZ game.
- General case independently discovered by Mermin (1990) and Brassard, Broadbent, Tapp (2005).

Quantum Strategy

PHP: Given even parity x, find y such that $|y| \equiv \frac{1}{2}|x| \pmod 2$.

Theorem

Given the state $|\text{🐱}\rangle = \frac{1}{\sqrt{2}}(|0\cdots0\rangle + |1\cdots1\rangle)$, *quantum players can always win.*

Quantum Strategy

PHP: Given even parity x, find y such that $|y| \equiv \frac{1}{2}|x| \pmod 2$.

Theorem

Given the state $|\text{😸}\rangle = \frac{1}{\sqrt{2}}(|0\cdots0\rangle + |1\cdots1\rangle)$, *quantum players can always win.*

Proof.

Each player applies $S = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ to their qubit if $x_j = 1$. State is $|0\cdots0\rangle + i^{|x|}|1\cdots1\rangle$.

Quantum Strategy

PHP: Given even parity x, find y such that $|y| \equiv \frac{1}{2}|x| \pmod 2$.

Theorem

Given the state $|\text{😸}\rangle = \frac{1}{\sqrt{2}}(|0\cdots0\rangle + |1\cdots1\rangle)$, *quantum players can always win.*

Proof.

Each player applies $S = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ to their qubit if $x_j = 1$. State is $|0\cdots0\rangle + i^{|x|}|1\cdots1\rangle$.

$$|0\cdots0\rangle + |1\cdots1\rangle \qquad\qquad |0\cdots0\rangle - |1\cdots1\rangle$$

□

Quantum Strategy

PHP: Given even parity x, find y such that $|y| \equiv \frac{1}{2}|x| \pmod 2$.

Theorem

Given the state $|\text{🐱}\rangle = \frac{1}{\sqrt{2}}\left(|0\cdots 0\rangle + |1\cdots 1\rangle\right)$, *quantum players can always win.*

Proof.

Each player applies $S = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ to their qubit if $x_j = 1$. State is $|0\cdots 0\rangle + i^{|x|}|1\cdots 1\rangle$.

$$|0\cdots 0\rangle + |1\cdots 1\rangle \xrightarrow{H^{\otimes n}} \sum_{|x| \text{ even}} |x\rangle \qquad |0\cdots 0\rangle - |1\cdots 1\rangle \xrightarrow{H^{\otimes n}} \sum_{|x| \text{ odd}} |x\rangle$$

□

Quantum Strategy

PHP: Given even parity x, find y such that $|y| \equiv \frac{1}{2}|x| \pmod 2$.

Theorem

Given the state $|\text{🐱}\rangle = \frac{1}{\sqrt{2}}(|0\cdots0\rangle + |1\cdots1\rangle)$, *quantum players can always win.*

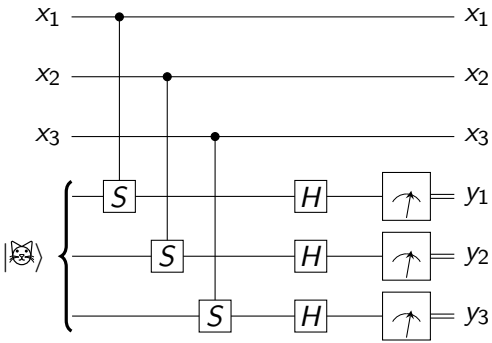
Proof.

Each player applies $S = \left(\begin{smallmatrix} 1 & 0 \\ 0 & i \end{smallmatrix}\right)$ to their qubit if $x_j = 1$. State is $|0\cdots0\rangle + i^{|x|}|1\cdots1\rangle$.

$$|0\cdots0\rangle + |1\cdots1\rangle \xrightarrow{H^{\otimes n}} \sum_{|x| \text{ even}} |x\rangle \qquad |0\cdots0\rangle - |1\cdots1\rangle \xrightarrow{H^{\otimes n}} \sum_{|x| \text{ odd}} |x\rangle$$

Therefore all players apply a Hadamard, measure, and output the result. □

QNC^0/qpoly circuit



Classical Strategy

PHP: Given even parity x, find y such that $|y| = \frac{1}{2}|x| \pmod 2$.

Theorem (Game Hardness – Brassard, Broadbent, Tapp)

Any deterministic strategy wins on a random input with probability at most $\frac{1}{2} + 2^{-\lceil n/2 \rceil}$.

Classical Strategy

PHP: Given even parity x, find y such that $|y| = \frac{1}{2}|x| \pmod 2$.

Theorem (Game Hardness – Brassard, Broadbent, Tapp)

Any deterministic strategy wins on a random input with probability at most $\frac{1}{2} + 2^{-\lceil n/2 \rceil}$.

Vague Intuition

- Output parity depends on input HW modulo 4.

Classical Strategy

PHP: Given even parity x, find y such that $|y| = \frac{1}{2}|x| \pmod 2$.

Theorem (Game Hardness – Brassard, Broadbent, Tapp)

Any deterministic strategy wins on a random input with probability at most $\frac{1}{2} + 2^{-\lceil n/2 \rceil}$.

Vague Intuition

- Output parity depends on input HW modulo 4.
- Any one bit is *almost* completely independent of HW mod 4.

Classical Strategy

PHP: Given even parity x, find y such that $|y| = \frac{1}{2}|x| \pmod 2$.

Theorem (Game Hardness – Brassard, Broadbent, Tapp)

Any deterministic strategy wins on a random input with probability at most $\frac{1}{2} + 2^{-\lceil n/2 \rceil}$.

Vague Intuition

- Output parity depends on input HW modulo 4.
- Any one bit is *almost* completely independent of HW mod 4.
- Fraction of strings with HW $i \pmod 4$ is $\frac{1}{4} + O(2^{-n/2})$.

Locality in circuits

Definition

A circuit is ℓ-*local* if each output bit depends on at most ℓ input bits.

Locality in circuits

Definition

A circuit is ℓ-*local* if each output bit depends on at most ℓ input bits.

Fact

A strategy for the game implies a 1*-local circuit for* PHP*.*

Locality in circuits

Definition

A circuit is ℓ-*local* if each output bit depends on at most ℓ input bits.

Fact

A strategy for the game implies a 1*-local circuit for* PHP*.*

Can improve game hardness to 1-local hardness.

Theorem (1-local hardness)

A 1*-local classical circuit solves* PHP_n *on a random input w.p.* $\leq \frac{1}{2} + 2^{-\lceil n/2 \rceil}$.

Locality $\ell > 1$

Idea

Reduce to 1-local circuit

Locality $\ell > 1$

Idea

Reduce to 1-local circuit by *restricting* some input bits.

Locality $\ell > 1$

Idea

Reduce to 1-local circuit by *restricting* some input bits.

- How do we reduce locality to 1?
- What problem does a circuit for PHP solve after restriction?

Locality $\ell > 1$

Lemma

Consider a circuit with n inputs, n outputs, and locality ℓ.
We can find a subset of $\Omega(\frac{n}{\ell^2})$ input bits such that restricting all other inputs gives a 1-local circuit.

Locality $\ell > 1$

Lemma

Consider a circuit with n inputs, n outputs, and locality ℓ.
We can find a subset of $\Omega(\frac{n}{\ell^2})$ input bits such that restricting all other inputs gives a 1-local circuit.

Proof.

Consider a graph with

- a vertex for each input bit,
- an edge if both inputs affect some common output bit.

Locality $\ell > 1$

Lemma

Consider a circuit with n inputs, n outputs, and locality ℓ.
We can find a subset of $\Omega(\frac{n}{\ell^2})$ input bits such that restricting all other inputs gives a 1-local circuit.

Proof.

Consider a graph with

- a vertex for each input bit,
- an edge if both inputs affect some common output bit.

Choose an independent set of vertices to get locality 1.

Locality $\ell > 1$

Lemma

Consider a circuit with n inputs, n outputs, and locality ℓ.
We can find a subset of $\Omega(\frac{n}{\ell^2})$ input bits such that restricting all other inputs gives a 1-local circuit.

Proof.

Consider a graph with

- a vertex for each input bit,
- an edge if both inputs affect some common output bit.

Choose an independent set of vertices to get locality 1.

Turán's theorem: Largest independent set has size $\Omega(n^2/|E|)$.

Locality $\ell > 1$

Lemma

Consider a circuit with n inputs, n outputs, and locality ℓ.
We can find a subset of $\Omega(\frac{n}{\ell^2})$ input bits such that restricting all other inputs gives a 1-local circuit.

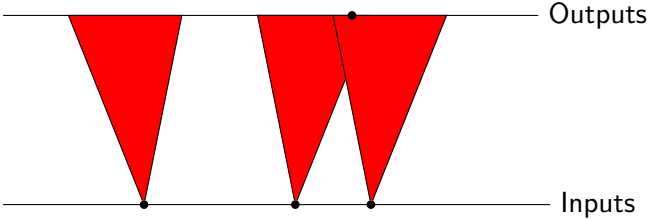
Proof.

Consider a graph with

- a vertex for each input bit,
- an edge if both inputs affect some common output bit.

Choose an independent set of vertices to get locality 1.

Turán's theorem: Largest independent set has size $\Omega(n^2/|E|)$.
Each output is responsible for at most $O(\ell^2)$ edges $\implies |E| = O(n\ell^2)$. □


Outputs
Inputs

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (Original)

Given $x \in \{0,1\}^n$ such that $|x| \equiv 0 \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv 0 \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv 2 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (Variant 1)

Given $x \in \{0,1\}^n$ such that $|x| \equiv 1 \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv 3 \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv 1 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (Variant 2)

Given $x \in \{0,1\}^n$ such that $|x| \equiv 0 \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv 2 \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv 0 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (Variant 3)

Given $x \in \{0,1\}^n$ such that $|x| \equiv 1 \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv 1 \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv 3 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (All Variants)

Given $x \in \{0,1\}^n$ such that $|x| \equiv b \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv b \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv b+2 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

where $b \in \{0,1,2,3\}$.

Restrictions of PHP

What happens when we take a circuit for PHP and fix some input bits?

E.g., $x_n = 1 \implies$ remaining inputs have odd parity.

Parity Halving Problem (All Variants)

Given $x \in \{0,1\}^n$ such that $|x| \equiv b \pmod 2$, output $y \in \{0,1\}^n$ such that

$$|x| \equiv b \pmod 4 \implies |y| \equiv 0 \pmod 2,$$
$$|x| \equiv b+2 \pmod 4 \implies |y| \equiv 1 \pmod 2.$$

where $b \in \{0,1,2,3\}$.

Claim

All problems are equivalent.

Locality-ℓ Hardness Result

Theorem

An ℓ-local classical circuit solves PHP_n *on a random input w.p.* $\leq \frac{1}{2} + 2^{-\Omega(n/\ell^2)}$.

Proof.

- Find $\Omega(n/\ell^2)$ inputs with non-overlapping light cones. Fix the rest.
- The remaining circuit solves a variant of PHP on $\Omega(n/\ell^2)$ bits.

□

Locality-ℓ Hardness Result

Theorem

An ℓ-local classical circuit solves PHP_n *on a random input w.p.* $\leq \frac{1}{2} + 2^{-\Omega(n/\ell^2)}$.

Proof.

- Find $\Omega(n/\ell^2)$ inputs with non-overlapping light cones. Fix the rest.
- The remaining circuit solves a variant of PHP on $\Omega(n/\ell^2)$ bits.

□

Corollary

Since NC^0 *circuits have locality* $\ell = O(1)$, *they solve* PHP *w.p.* $\frac{1}{2} + 2^{-\Omega(n)}$.

AC^0 hardness

Problem

Unbounded fan-in gates make it easy to have locality n.

AC^0 hardness

Problem

Unbounded fan-in gates make it easy to have locality n.

Solution

Finally some circuit complexity theory: the Switching Lemma!!

AC^0 hardness

Problem

Unbounded fan-in gates make it easy to have locality n.

Solution

Finally some circuit complexity theory: the Switching Lemma!!

Switching Lemma (Intuition)

Consider an AC^0 circuit. With high probability, restricting a (large) random subset of bits produces a circuit with $n^{o(1)}$ locality.

Switching Lemma in Action

Theorem

AC^0 *circuits solve* PHP *w.p. at most* $\frac{1}{2} + o(1)$.

Proof.

Apply the switching lemma.
Locality is reduced, and the resulting circuit solves a variant of PHP, so hardness for local circuits implies $\frac{1}{2} + o(1)$ probability of success. □

Section 3

Relaxed Parity Halving Problem

We don't want to use an advice state, but we can't construct it ourselves.

Theorem

The state $\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$ *cannot be constructed in* QNC^0.

We don't want to use an advice state, but we can't construct it ourselves.

Theorem

The state $\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$ *cannot be constructed in* QNC^0.

But we can construct a **poor man's cat state**!

$$\frac{1}{\sqrt{2}}\left(|z\rangle + |\overline{z}\rangle\right)$$

We don't want to use an advice state, but we can't construct it ourselves.

Theorem

The state $\frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$ *cannot be constructed in* QNC^0.

But we can construct a **poor man's cat state**!

$$\frac{1}{\sqrt{2}}\left(|z\rangle + |\overline{z}\rangle\right) = X^z \frac{1}{\sqrt{2}}\left(|0^n\rangle + |1^n\rangle\right)$$

We don't want to use an advice state, but we can't construct it ourselves.

Theorem

The state $\frac{1}{\sqrt{2}}(|0^n\rangle + |1^n\rangle)$ *cannot be constructed in* QNC^0.

But we can construct a **poor man's cat state**!

$$\frac{1}{\sqrt{2}}(|z\rangle + |\overline{z}\rangle) = X^z \frac{1}{\sqrt{2}}(|0^n\rangle + |1^n\rangle)$$

A cat state with some bits flipped.

Poor Man's Cat State

Q: If we can construct $|z\rangle + |\overline{z}\rangle = X^z\,|$😹$\rangle$ in QNC^0, then why can't we apply X^z to get $|$😹$\rangle$?

Poor Man's Cat State

Q: If we can construct $|z\rangle + |\overline{z}\rangle = X^z\,|$😸$\rangle$ in QNC^0, then why can't we apply X^z to get $|$😸$\rangle$?
A: We don't know what z is!!

Poor Man's Cat State

Q: If we can construct $|z\rangle + |\overline{z}\rangle = X^z\,|\text{😸}\rangle$ in QNC^0, then why can't we apply X^z to get $|\text{😸}\rangle$?
A: We don't know what z is!!

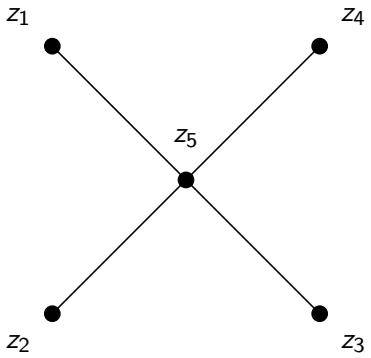
Theorem

In QNC^0 *we can*

- *construct* $\frac{1}{\sqrt{2}}\left(|z\rangle + |\overline{z}\rangle\right)$ *for some uniformly random* $z \in \{0,1\}^n$,
- *with information* $d \in \{0,1\}^{n-1}$ *from which* z *can be recovered (up to complement) in* AC^0.

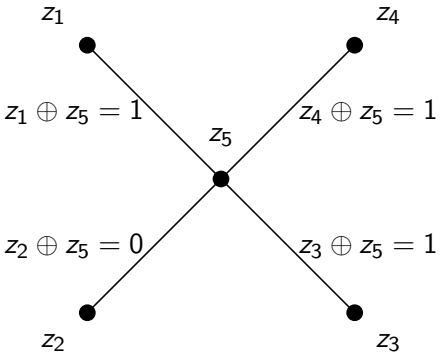
Poor Man's Cat State Example

Consider a tree $G = (V, E)$. Let there be a $|+\rangle$ qubit for each vertex.



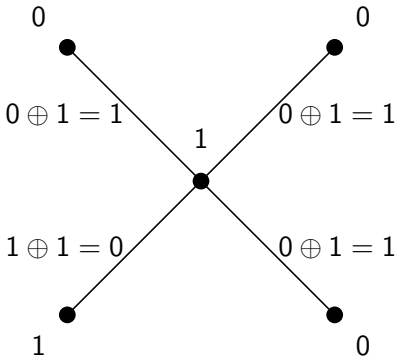
Poor Man's Cat State Example

For each edge, measure the parity of the two endpoints.



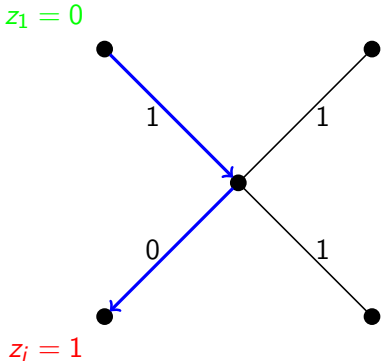
Poor Man's Cat State Example

Two vectors, z and $\overline{z}$, are consistent with these measurements.



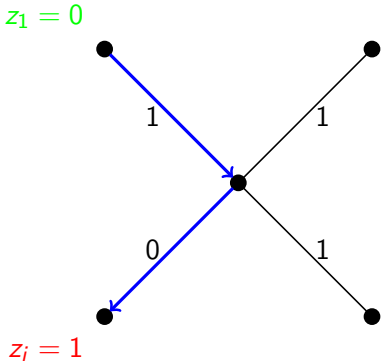
Poor Man's Cat State Example

To construct z, let z_i be the parity of the path from z_1 to z_i.



Poor Man's Cat State Example

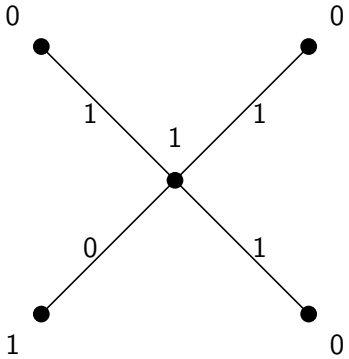
To construct z, let z_i be the parity of the path from z_1 to z_i.



$$z_2 = z_1 \oplus z_2 = (z_1 \oplus z_5) \oplus (z_2 \oplus z_5) = 1 \oplus 0 = 1$$

Poor Man's Cat State Example

Final output is state $|01001\rangle + |10110\rangle$ (vertex qubits) and $d = 1011$ (edge measurements).



What kind of tree to use?

Line graph!

What kind of tree to use?

Line graph!

We want low diameter, so it is easier to compute z from d.

What kind of tree to use?

Line graph!

We want low diameter, so it is easier to compute z from d.

Star graph!

What kind of tree to use?

Line graph!

We want low diameter, so it is easier to compute z from d.

Star graph!

We want low degree, since edges incident at the same vertex cannot be simultaneously measured.

What kind of tree to use?

Line graph!

We want low diameter, so it is easier to compute z from d.

Star graph!

We want low degree, since edges incident at the same vertex cannot be simultaneously measured.

Balanced binary tree!

Max degree $\Delta = 3$, diameter $d = \Theta(\log n)$. (AC^0 can compute $O(\log n)$ size parities)

Relaxed Parity Halving Problem

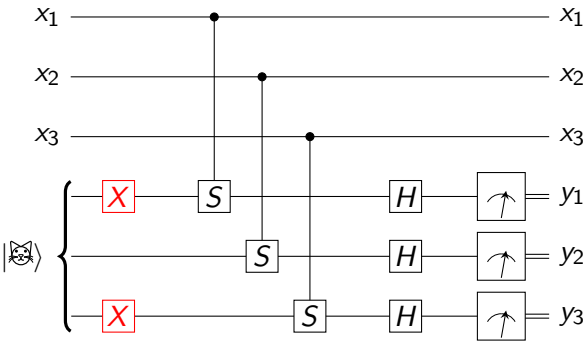
Q: What do we do with the poor man's cat state?

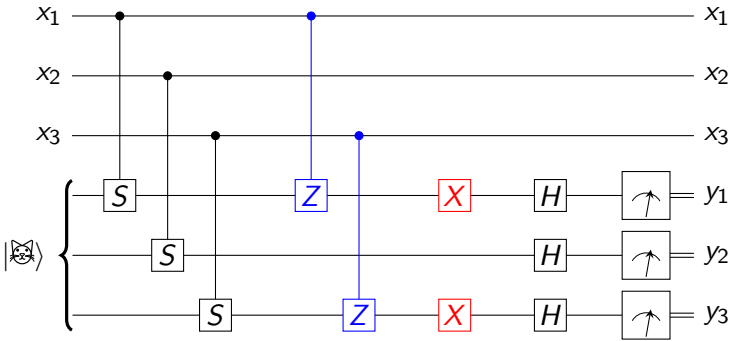
Relaxed Parity Halving Problem

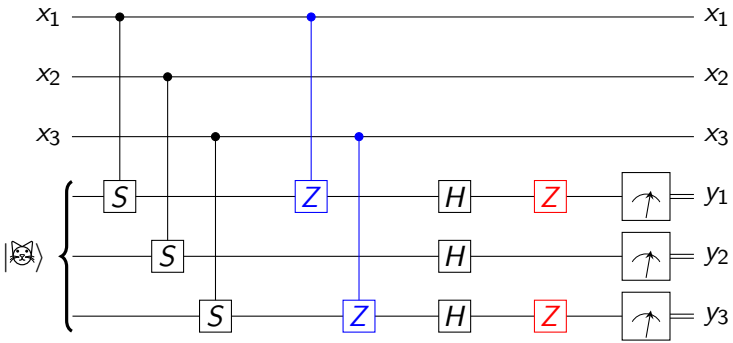
Q: What do we do with the poor man's cat state?
A: Pretend it's a cat state and run the same algorithm!!

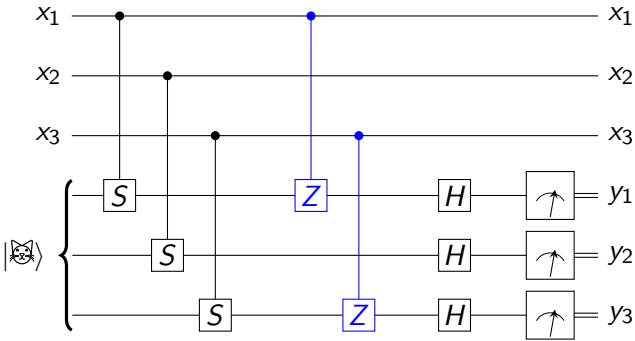
Relaxed Parity Halving Problem

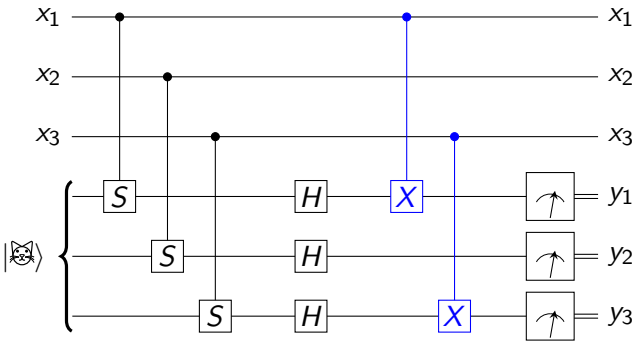
Q: What do we do with the poor man's cat state?
A: Pretend it's a cat state and run the same algorithm!!


x_1
x_2
x_3
$|$ $\rangle$
X
S
S
S
X
H
H
H
x_1
x_2
x_3
y_1
y_2
y_3


x_1
x_2
x_3
$|$ $\rangle$
S
S
S
Z
Z
X
X
H
H
H
x_1
x_2
x_3
y_1
y_2
y_3


x_1
x_2
x_3
$|$ $\rangle$
S
S
S
Z
Z
H
H
H
Z
Z
x_1
x_2
x_3
y_1
y_2
y_3


x_1
x_2
x_3
$|$ $\rangle$
S
S
S
Z
Z
H
H
H
x_1
x_2
x_3
y_1
y_2
y_3


x_1
x_2
x_3
S
S
S
H
H
H
X
X
x_1
x_2
x_3
y_1
y_2
y_3

Relaxed Parity Halving Problem

Given an even parity input $x \in \{0,1\}^n$, output $y \in \{0,1\}^n$ such that

$$|y| \equiv \frac{1}{2}|x| + \langle x, z \rangle \pmod 2$$

.

Relaxed Parity Halving Problem

Given an even parity input $x \in \{0,1\}^n$, output $y \in \{0,1\}^n$ and $d \in \{0,1\}^{n-1}$ such that

$$|y| \equiv \frac{1}{2}|x| + \langle x, z \rangle \pmod 2$$

where $z \in \{0,1\}^n$ is either vector consistent d.

Relaxed Parity Halving Problem

Given an even parity input $x \in \{0,1\}^n$, output $y \in \{0,1\}^n$ and $d \in \{0,1\}^{n-1}$ such that

$$|y| \equiv \frac{1}{2}|x| + \langle x, z \rangle \pmod 2$$

where $z \in \{0,1\}^n$ is either vector consistent d.

RPHP $\in$ QNC^0 is clear.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Proof.

- Suppose we have a circuit for RPHP.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Proof.

- Suppose we have a circuit for RPHP.
- In AC^0, we can compute z from d because each z_i is an $O(\log n)$-bit parity.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Proof.

- Suppose we have a circuit for RPHP.
- In AC^0, we can compute z from d because each z_i is an $O(\log n)$-bit parity.
- "Correct" for z. Remove $\langle x, z \rangle$.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Proof.

- Suppose we have a circuit for RPHP.
- In AC^0, we can compute z from d because each z_i is an $O(\log n)$-bit parity.
- "Correct" for z. Remove $\langle x, z \rangle$.
  - Compute $w_i = x_i z_i$ for all i.
  - XOR in corrections: $y'_i := y_i \oplus w_i$.
  - Note $|y'| = |y| + \langle x, z \rangle = \frac{1}{2}|x| \pmod 2$.

Classical Hardness

Theorem

Any AC^0 *circuit for* RPHP *succeeds with probability* $< \frac{1}{2} + o(1)$.

Proof.

- Suppose we have a circuit for RPHP.
- In AC^0, we can compute z from d because each z_i is an $O(\log n)$-bit parity.
- "Correct" for z. Remove $\langle x, z \rangle$.
  - Compute $w_i = x_i z_i$ for all i.
  - XOR in corrections: $y'_i := y_i \oplus w_i$.
  - Note $|y'| = |y| + \langle x, z \rangle = \frac{1}{2}|x| \pmod 2$.
- New circuit solves PHP (with the same probability)

□

Section 4

Extensions

Extensions – Better Parameters

State of the art switching lemma results (Hastad and Rossman) give

Theorem

AC^0 *circuits solve* RPHP *on a random input w.p. at most* $\frac{1}{2} + 2^{-n^{0.999}}$.

Extensions – Better Parameters

State of the art switching lemma results (Hastad and Rossman) give

Theorem

AC^0 *circuits solve* RPHP *on a random input w.p. at most* $\frac{1}{2} + 2^{-n^{0.999}}$.

Theorem

AC^0 *circuits of depth* d *and size* $\exp(n^{1/2d})$ *solve* RPHP *w.p. at most* $\frac{1}{2} + 2^{-n^{0.999}}$.

Extensions – Better Parameters

State of the art switching lemma results (Hastad and Rossman) give

Theorem

AC^0 *circuits solve* RPHP *on a random input w.p. at most* $\frac{1}{2} + 2^{-n^{0.999}}$.

Theorem

AC^0 *circuits of depth* d *and size* $\exp(n^{1/2d})$ *solve* RPHP *w.p. at most* $\frac{1}{2} + 2^{-n^{0.999}}$.

Note: RPHP or PHP are solved exactly by $\exp(n^{1/d})$ size AC^0 circuits.

Parallel Copies

Parallel Parity Halving Problem

Given inputs $x_1, \ldots, x_k \in \{0,1\}^n$, output $y_1, \ldots, y_k \in \{0,1\}^n$ such that for all i,

$$|x_i| \equiv \frac{1}{2}|y_i| \pmod 2.$$

In other words, make the circuit solve k copies of the problem at once.

Parallel Copies

Parallel Parity Halving Problem

Given inputs $x_1, \ldots, x_k \in \{0,1\}^n$, output $y_1, \ldots, y_k \in \{0,1\}^n$ such that for all i,

$$|x_i| \equiv \frac{1}{2}|y_i| \pmod 2.$$

In other words, make the circuit solve k copies of the problem at once.

Theorem

AC^0 *circuits of depth* d *and size* $\exp(n^{1/2d})$ *solve* Parallel-RPHP *w.p. at most* $2^{-n^{0.999}}$.

2D Locality

- Any tree in the grid has diameter $\Omega(\sqrt{n})$.
- A different reduction works for trees with diameter $d = o(n)$.

Theorem

There exists a constant-depth quantum circuit for Grid-RPHP *which is local on a 2D grid.*

2D Locality

- Any tree in the grid has diameter $\Omega(\sqrt{n})$.
- A different reduction works for trees with diameter $d = o(n)$.

Theorem

There exists a constant-depth quantum circuit for Grid-RPHP *which is local on a 2D grid.*

Parallel-Grid-RPHP . . .

Extensions – HLF and Geometric Locality

RPHP reduces to HLF

Theorem

RPHP $\leq$ HLF.
There is no AC^0 *circuit for* HLF.

Extensions – HLF and Geometric Locality

RPHP reduces to HLF

Theorem

RPHP $\leq$ HLF.
There is no AC^0 *circuit for* HLF.

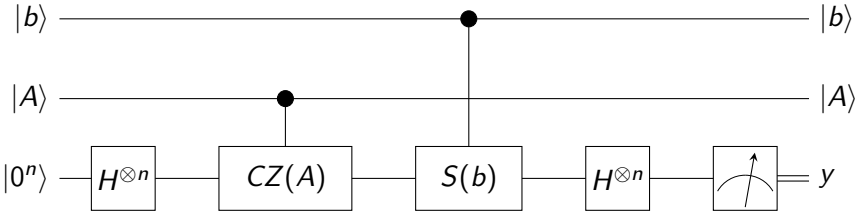
Theorem

Parallel-Grid-RPHP $\leq$ 2DHLF

HLF Problem

Hidden Linear Function Problem

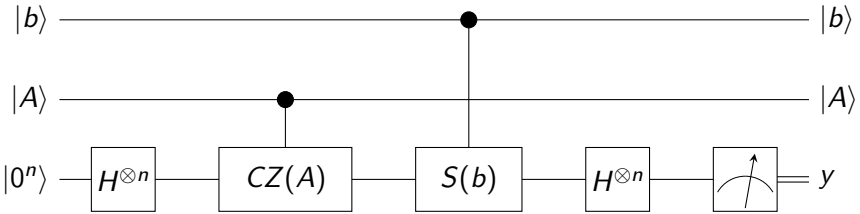
Given a symmetric matrix $A \in \{0,1\}^{n \times n}$ and vector $b \in \{0,1,2,3\}^n$, output any string y that may be output by the following circuit.



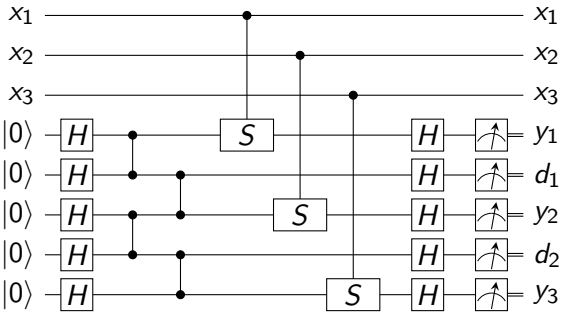
HLF Problem

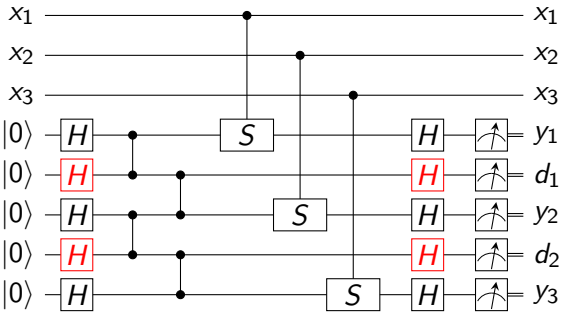
Hidden Linear Function Problem

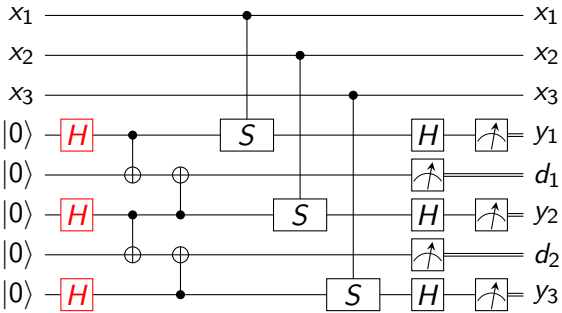
Given a symmetric matrix $A \in \{0,1\}^{n \times n}$ and vector $b \in \{0,1,2,3\}^n$, output any string y that may be output by the following circuit.

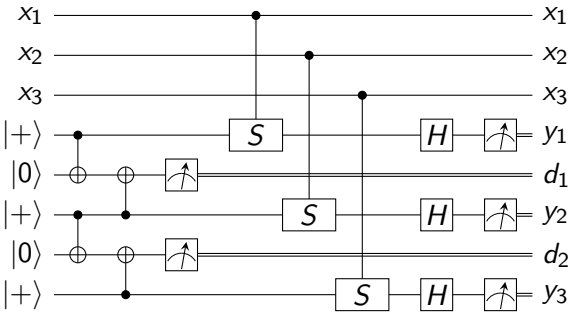


Suppose we fix all of A, part of b.


x_1
x_2
x_3
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
S
x_1
x_2
x_3
y_1
d_1
y_2
d_2
y_3


x_1
x_2
x_3
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
S
x_1
x_2
x_3
y_1
d_1
y_2
d_2
y_3


x_1
x_2
x_3
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
$|0\rangle$
H
H
H
S
S
S
H
H
H
x_1
x_2
x_3
y_1
d_1
y_2
d_2
y_3


x_1
x_2
x_3
$|+\rangle$
$|0\rangle$
$|+\rangle$
$|0\rangle$
$|+\rangle$
S
S
S
H
H
H
x_1
x_2
x_3
y_1
d_1
y_2
d_2
y_3

Open Problems

- Can improve the classical hardness to more powerful circuit classes? $AC^0[2]$, TC^0, NC^1
  - Problem: Best circuit lower bounds stop around TC^0. Would need to be conditional.
  - Partial result: QNC^0/qpoly vs. $AC^0[2]$.
- Can we get the same separation with 1D local circuits?

Thank You!