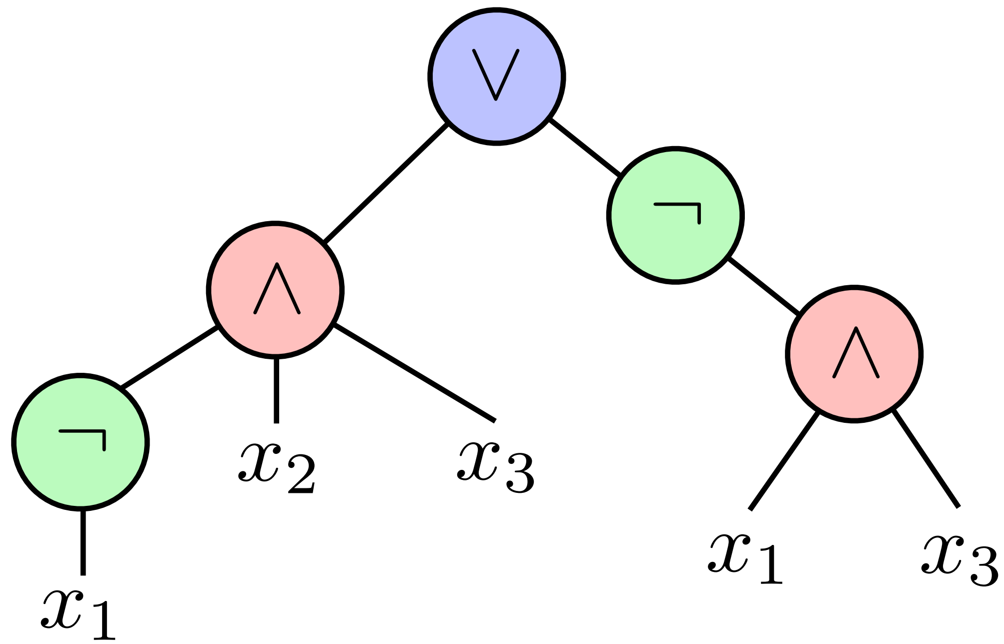
The quantum query complexity of read-many formulas

Andrew Childs
Waterloo

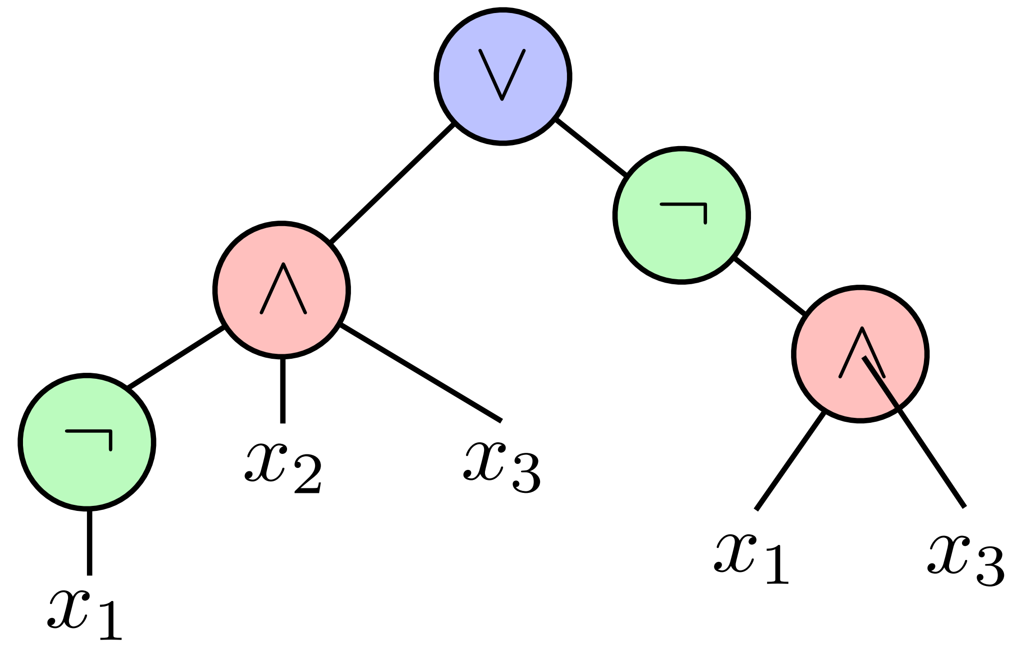
Shelby Kimmel
MIT

Robin Kothari
Waterloo

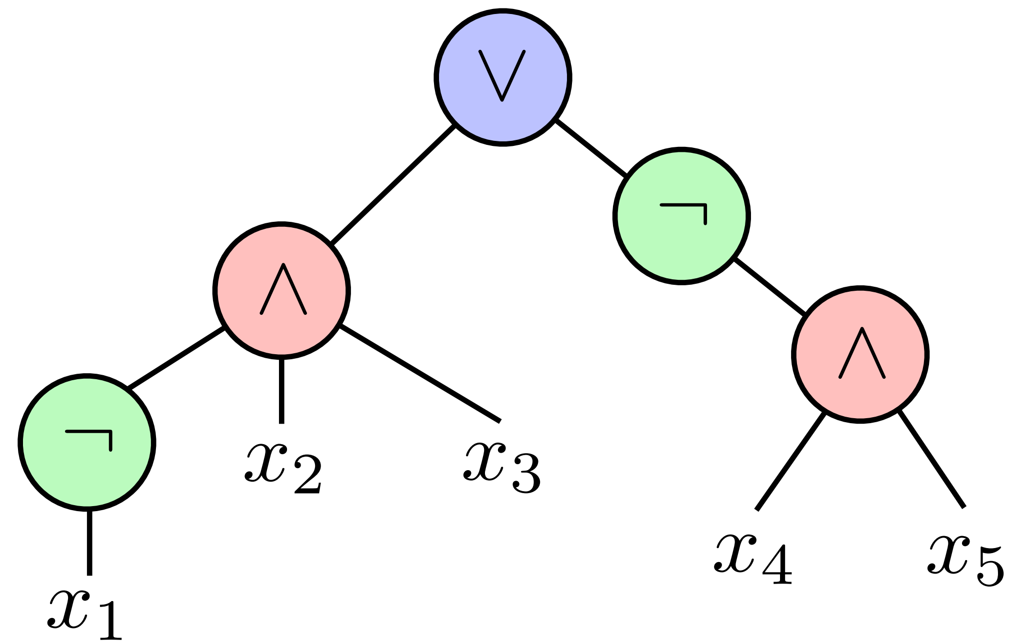
Boolean formulas



Boolean formulas



A formula is *read-once* if every input appears at most once.



Evaluating read-once formulas

Problem: Given a black box for $x \in \{0,1\}^n$, evaluate $f(x)$, where f is a fixed *read-once* formula

Evaluating read-once formulas

Problem: Given a black box for $x \in \{0,1\}^n$, evaluate $f(x)$, where f is a fixed *read-once* formula

Upper bounds:

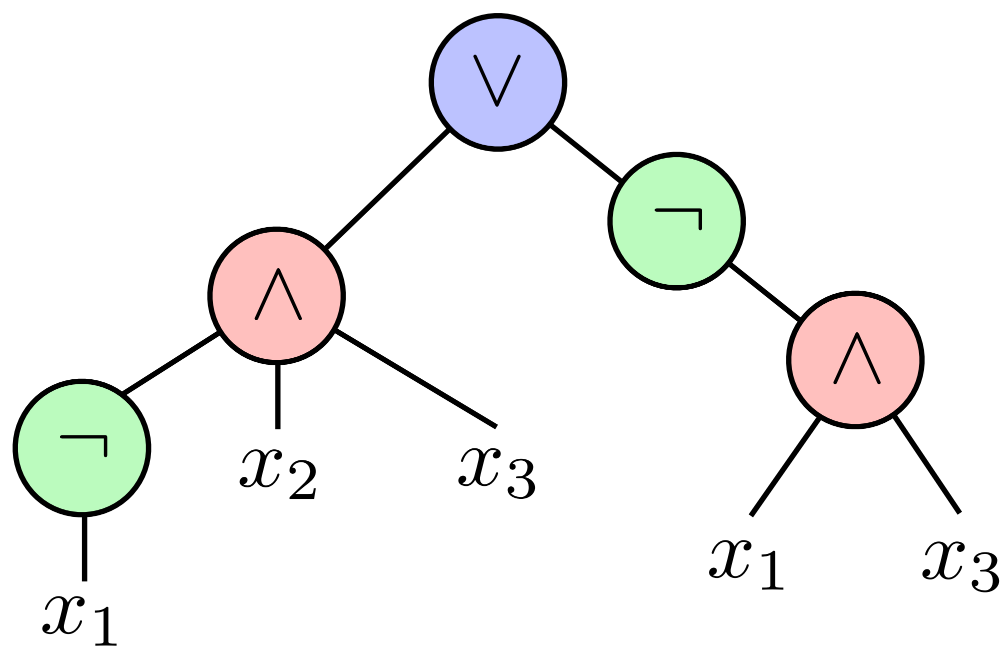
- Grover 96: $O(\sqrt{n})$ for OR
- Buhrman, Cleve, Wigderson 98: $\tilde{O}(\sqrt{n})$ for balanced, constant-depth
- Høyer, Mosca, de Wolf 03: $O(\sqrt{n})$ for balanced, constant-depth
- Farhi, Goldstone, Gutmann 07: $n^{\frac{1}{2}+o(1)}$ for balanced, binary
- Ambainis, Childs, Reichardt, Špalek, Zhang 07: $O(\sqrt{n})$ for approximately balanced formulas, $n^{\frac{1}{2}+o(1)}$ in general
- Reichardt 11: $O(\sqrt{n})$ for any formula

Lower bound:

- Barnum, Saks 04: $\Omega(\sqrt{n})$

Formula size

The size S of a formula is its total number of inputs, counted with multiplicity.



$$n = 3$$
$$S = 5$$

Every Boolean function can be computed by some formula. The formula size is a natural complexity measure.

Evaluating read-many formulas

The optimal read-once formula evaluation algorithm gives an upper bound of $O(\sqrt{S})$ for general formulas, but this can be suboptimal for read-many formulas.

Evaluating read-many formulas

The optimal read-once formula evaluation algorithm gives an upper bound of $O(\sqrt{S})$ for general formulas, but this can be suboptimal for read-many formulas.

Trivial example: $x_1 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_2 \vee \cdots \vee x_n \vee \bar{x}_n = 1$

$S = 2n$ but no queries are required to evaluate

Evaluating read-many formulas

The optimal read-once formula evaluation algorithm gives an upper bound of $O(\sqrt{S})$ for general formulas, but this can be suboptimal for read-many formulas.

Trivial example: $x_1 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_2 \vee \cdots \vee x_n \vee \bar{x}_n = 1$

$S = 2n$ but no queries are required to evaluate

Nontrivial example: Graph collision.

Fix an n-vertex graph. Given a black box for $x \in \{0,1\}^n$. Is there an edge (v,w) of the graph with $x_v = x_w = 1$?

Evaluating read-many formulas

The optimal read-once formula evaluation algorithm gives an upper bound of $O(\sqrt{S})$ for general formulas, but this can be suboptimal for read-many formulas.

Trivial example: $x_1 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_2 \vee \cdots \vee x_n \vee \bar{x}_n = 1$

$S = 2n$ but no queries are required to evaluate

Nontrivial example: Graph collision.

Fix an n-vertex graph. Given a black box for $x \in \{0,1\}^n$. Is there an edge (v,w) of the graph with $x_v = x_w = 1$?

Upper bound of $O(n^{2/3})$ for any graph [Magniez, Santha, Szegedy 05]. Best lower bound for any particular graph is $\Omega(n^{1/2})$.

Evaluating read-many formulas

The optimal read-once formula evaluation algorithm gives an upper bound of $O(\sqrt{S})$ for general formulas, but this can be suboptimal for read-many formulas.

Trivial example: $x_1 \vee \bar{x}_1 \vee x_2 \vee \bar{x}_2 \vee \cdots \vee x_n \vee \bar{x}_n = 1$

$S = 2n$ but no queries are required to evaluate

Nontrivial example: Graph collision.

Fix an n-vertex graph. Given a black box for $x \in \{0,1\}^n$. Is there an edge (v,w) of the graph with $x_v = x_w = 1$?

Upper bound of $O(n^{2/3})$ for any graph [Magniez, Santha, Szegedy 05]. Best lower bound for any particular graph is $\Omega(n^{1/2})$.

Can be expressed by a simple formula:

$$\bigvee_{\text{edges } (v,w)} x_v \wedge x_w$$

n inputs

size $S = 2m = O(n^2)$

More parameters

To get nontrivial bounds for read-many formula evaluation, we must take other properties into account.

More parameters

To get nontrivial bounds for read-many formula evaluation, we must take other properties into account.

Gate count G: Number of AND and OR gates in the formula

(Note that $G < S$: worst case is a binary tree, with $G = S - 1$)

More parameters

To get nontrivial bounds for read-many formula evaluation, we must take other properties into account.

Gate count G: Number of AND and OR gates in the formula

(Note that $G < S$: worst case is a binary tree, with $G = S - 1$)

Depth: Length of a longest path from the output to an input (not counting NOT gates)

Results

The quantum query complexity of evaluating a formula with n inputs, size S, and G gates is $O(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$.

Results

The quantum query complexity of evaluating a formula with n inputs, size S, and G gates is $O(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$.

For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

Results

The quantum query complexity of evaluating a formula with n inputs, size S, and G gates is $O(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$.

For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

The above lower bound still holds for any fixed constant depth $k \geq 3$.

Results

The quantum query complexity of evaluating a formula with n inputs, size S, and G gates is $O(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$.

For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

The above lower bound still holds for any fixed constant depth $k \geq 3$.

There is a depth-2 circuit of linear gate count that requires $\Omega(n^{0.555})$ queries to evaluate (compare $O(n^{3/4})$, trivial lower bound of $\Omega(\sqrt{n})$).

Quantum applications

Quantum applications

$\Omega(n^{1.055})$ lower bound for checking Boolean matrix multiplication

Given $n \times n$ Boolean matrices A, B, C,

decide whether $C_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}$ for all i, j.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Quantum applications

$\Omega(n^{1.055})$ lower bound for checking Boolean matrix multiplication

Given $n \times n$ Boolean matrices A, B, C,

decide whether $C_{ij} = \bigvee_{k=1}^{n} A_{ik} \wedge B_{kj}$ for all i, j.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Constant-depth, bounded-fanout *circuits* with n inputs and G gates (i.e., circuit size G) have query complexity $\tilde{\Theta}(\min\{n, n^{1/2}G^{1/4}\})$.

Classical applications

Classical applications

Formula *gate count* lower bound of $\Omega(n^2)$ for PARITY (improving over [Khrapchenko 71]).

Classical applications

Formula *gate count* lower bound of $\Omega(n^2)$ for PARITY (improving over [Khrapchenko 71]).

Constant-depth circuit of size $O(n)$ that requires $\Omega(n^{2-\epsilon})$ gates to express as a formula.

(Best previous result we know of this kind gave a similar lower bound for formula size [Nechiporuk 66, Jukna 12], which is weaker.)

Idea of the formula evaluation algorithm

Idea of the formula evaluation algorithm

Large formula size $\Rightarrow$ some inputs feed into many gates.

Idea of the formula evaluation algorithm

Large formula size $\Rightarrow$ some inputs feed into many gates.

Search for a 1 among inputs that feed into many OR gates.

If we find one, we eliminate many OR gates.

If we don't find one, we eliminate many wires.

Idea of the formula evaluation algorithm

Large formula size $\Rightarrow$ some inputs feed into many gates.

Search for a 1 among inputs that feed into many OR gates.

If we find one, we eliminate many OR gates.

If we don't find one, we eliminate many wires.

Search for a 0 among inputs that feed into many AND gates.

If we find one, we eliminate many AND gates.

If we don't find one, we eliminate many wires.

Idea of the formula evaluation algorithm

Large formula size $\Rightarrow$ some inputs feed into many gates.

Search for a 1 among inputs that feed into many OR gates.

If we find one, we eliminate many OR gates.

If we don't find one, we eliminate many wires.

Search for a 0 among inputs that feed into many AND gates.

If we find one, we eliminate many AND gates.

If we don't find one, we eliminate many wires.

Lemma: Using $O(n^{1/2}G^{1/4})$ queries, we can produce a formula of size $O(n\sqrt{G})$ with the same value on the given input.

Then apply the read-once formula evaluation algorithm.

Pruning a formula

Call an input *high-degree* if it feeds into more than $\sqrt{G}$ OR gates.

Repeatedly search for a marked high-degree input.

- We delete at least $\sqrt{G}$ OR gates each time, so we repeat $k = O(\sqrt{G})$ times.
- jth iteration takes time $O(\sqrt{n/m_j})$, where m_j is the number of marked high-degree inputs
- m_j decreases each step $\Rightarrow m_{k-j} \geq j$
- Total query complexity: $\sum_{j=1}^{O(\sqrt{G})} O\left(\sqrt{\frac{n}{j}}\right) = O(n^{1/2}G^{1/4})$

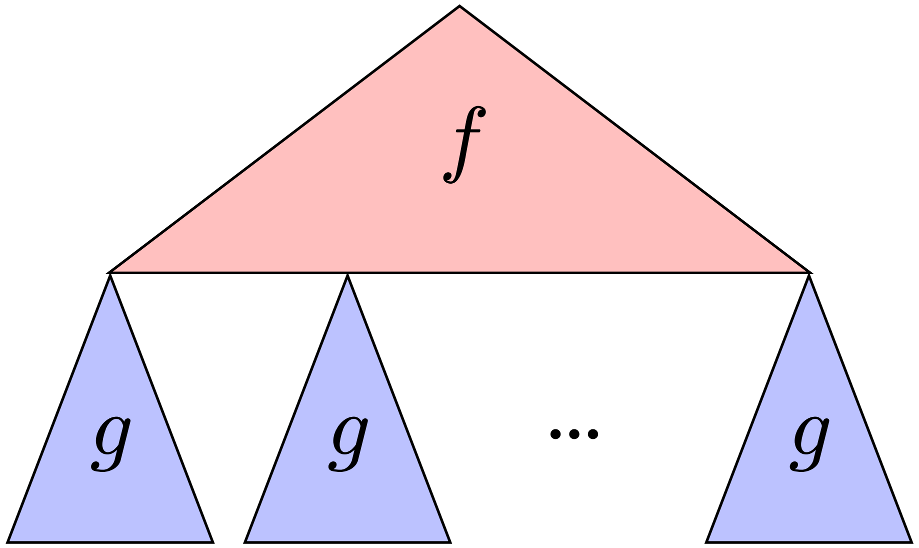
When there are no marked high-degree inputs, we can delete all wires from high-degree inputs to OR gates.

Same thing for AND gates.

Every input has degree at most $\sqrt{G} \Rightarrow$ formula size is $O(n\sqrt{G})$.

Note: No log factors in the analysis.

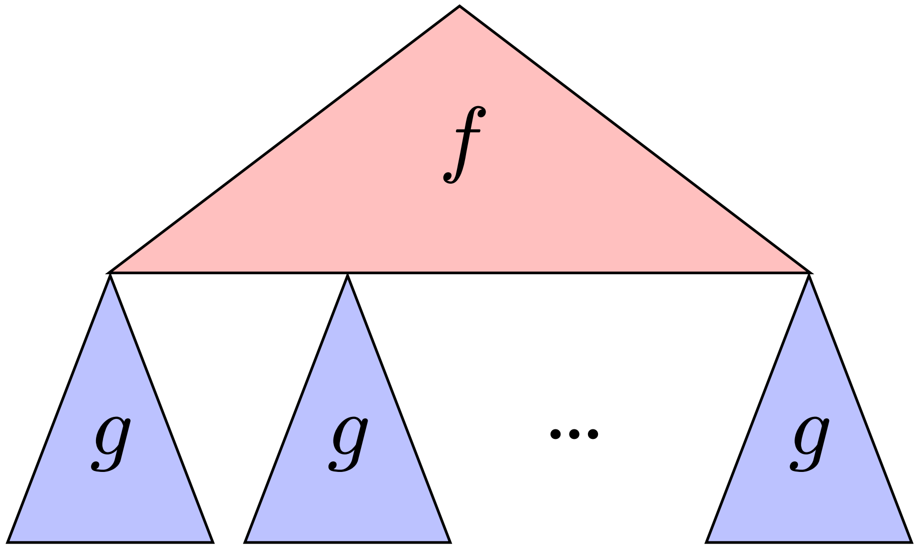
Lower bounds for composed formulas



$$Q\big(f \circ (g, \ldots, g)\big) = \Omega\big(Q(f)Q(g)\big)$$

[Reichardt 11]

Lower bounds for composed formulas

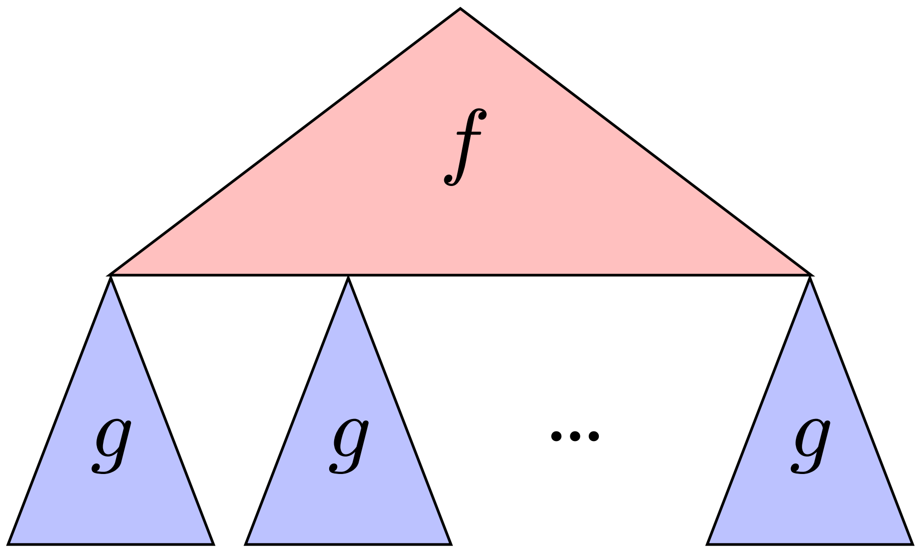


$$Q \;\; f \circ (g, \ldots, g) \;\; = \Omega \;\; Q(f)Q(g)$$

[Reichardt 11]

If the top gate of g is the same as all the bottom gates of f, then these gates can be combined, and we reduce the depth by 1.

Lower bounds for composed formulas



$$Q\left(f \circ (g, \ldots, g)\right) = \Omega\left(Q(f)Q(g)\right)$$

[Reichardt 11]

If the top gate of g is the same as all the bottom gates of f, then these gates can be combined, and we reduce the depth by 1.

Lemma: Let f, g be circuits with n_f, n_g inputs, depth k_f, k_g, size G_f, G_g. Then there exists a circuit h with $n_h = 4n_f n_g$ inputs, depth $k_h = k_f + k_g - 1$, size $G_h \leq 2G_f + 4n_f G_g$, such that $Q(h) = \Omega(Q(f)Q(g))$. Furthermore, if f is a formula and $k_g = 1$, then h is a formula of size $S_h = S_f S_g$.

Optimality of the formula evaluation algorithm

Claim: For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

Optimality of the formula evaluation algorithm

Claim: For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

If the min is n, consider PARITY:

Query complexity $\Omega(n)$ [BBCMW 98, FGGS 98]

Formula size $O(n^2)$ (use $x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$ recursively)

Optimality of the formula evaluation algorithm

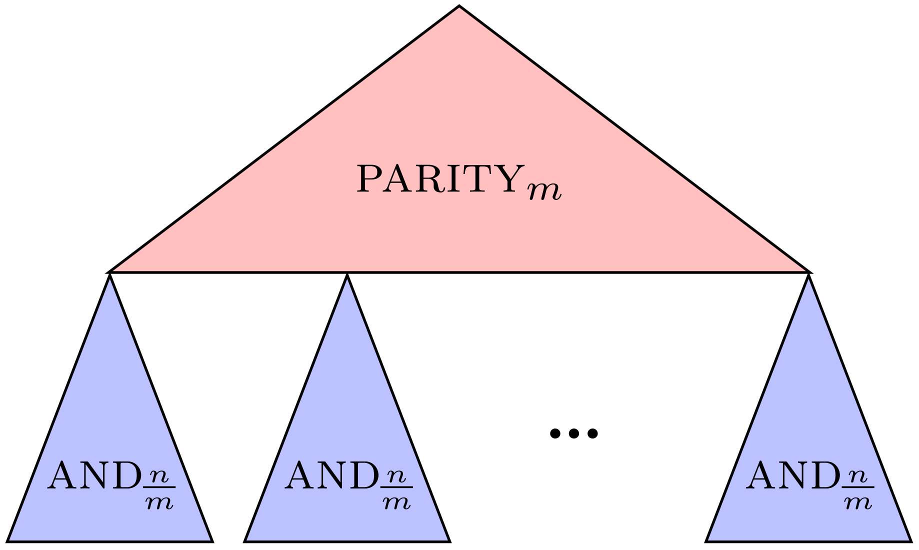
Claim: For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

If the min is n, consider PARITY:

Query complexity $\Omega(n)$ [BBCMW 98, FGGS 98]

Formula size $O(n^2)$ (use $x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$ recursively)

Otherwise, compose PARITY with AND:



$\Theta(n)$ inputs

size $S = O(m^2(n/m)) = O(nm)$

gate count $G = O(m^2)$

query complexity $\Omega(n\sqrt{n/m}) = \Omega(\sqrt{nm})$

Optimality of the formula evaluation algorithm

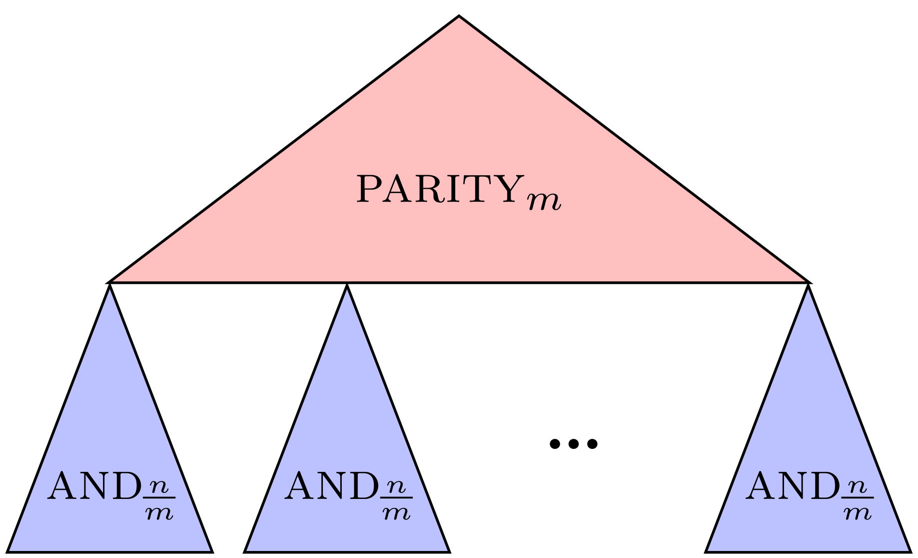
Claim: For any n, S, G, there is a formula with n inputs, size at most S, and at most G gates that requires $\Omega(\min\{n, \sqrt{S}, n^{1/2}G^{1/4}\})$ queries to evaluate.

If the min is n, consider PARITY:

Query complexity $\Omega(n)$ [BBCMW 98, FGGS 98]

Formula size $O(n^2)$ (use $x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$ recursively)

Otherwise, compose PARITY with AND:



$\Theta(n)$ inputs

size $S = O(m^2(n/m)) = O(nm)$

gate count $G = O(m^2)$

query complexity $\Omega(n\sqrt{n/m}) = \Omega(\sqrt{nm})$

Choosing m appropriately gives the desired result.
($m = S/n$ if the min is $\sqrt{S}$; $m = \sqrt{G}$ if the min is $n^{1/2}G^{1/4}$)

Constant-depth formulas (depth ≥ 3)

Constant-depth formulas (depth ≥ 3)

Constant-depth formulas for PARITY have superpolynomial size [Furst, Saxe, Sipser 84].

Constant-depth formulas (depth $\geq$ 3)

Constant-depth formulas for PARITY have superpolynomial size [Furst, Saxe, Sipser 84].

Instead, use the ONTO function [Beame, Machmouchi 10]:

$\textsc{onto}\colon X_n \to \{0,1\}$ $\qquad X_n =$ functions from $[2n-2]$ to $[n]$

$\textsc{onto}(f) = 1$ iff f is surjective

encode as a Boolean function of $N = (2n-2)\log n$ bits

Constant-depth formulas (depth $\geq$ 3)

Constant-depth formulas for PARITY have superpolynomial size [Furst, Saxe, Sipser 84].

Instead, use the ONTO function [Beame, Machmouchi 10]:

$\text{ONTO}\colon X_n \to \{0,1\}$ $\qquad X_n =$ functions from $[2n-2]$ to $[n]$

$\text{ONTO}(f) = 1$ iff f is surjective

encode as a Boolean function of $N = (2n-2)\log n$ bits

$$x^b = \begin{cases} x & \text{if } b = 1 \\ \bar{x} & \text{if } b = 0 \end{cases}$$

Depth-3 formula of size $\tilde{\Theta}(N^2)$: $\text{ONTO}(f) = \bigwedge_{j\in[n]} \bigvee_{i\in[2n-2]} \bigwedge_{\ell=0}^{\log n - 1} f(i)_\ell^{j_\ell}$

Constant-depth formulas (depth ≥ 3)

Constant-depth formulas for PARITY have superpolynomial size [Furst, Saxe, Sipser 84].

Instead, use the ONTO function [Beame, Machmouchi 10]:

$\textsc{onto}\colon X_n \to \{0,1\}$ $\qquad X_n =$ functions from $[2n-2]$ to $[n]$

$\textsc{onto}(f) = 1$ iff f is surjective

encode as a Boolean function of $N = (2n-2)\log n$ bits

$$x^b = \begin{cases} x & \text{if } b = 1 \\ \bar{x} & \text{if } b = 0 \end{cases}$$

Depth-3 formula of size $\tilde{\Theta}(N^2)$: $\textsc{onto}(f) = \bigwedge_{j\in[n]} \bigvee_{i\in[2n-2]} \bigwedge_{\ell=0}^{\log n - 1} f(i)_\ell^{j_\ell}$

Proposition [BM 10]: The query complexity of $\textsc{onto}_N$ is $\Omega(N/\log N)$.

Constant-depth formulas (depth $\geq$ 3)

Constant-depth formulas for PARITY have superpolynomial size [Furst, Saxe, Sipser 84].

Instead, use the ONTO function [Beame, Machmouchi 10]:

$\textsc{onto}\colon X_n \to \{0,1\}$ $\qquad X_n =$ functions from $[2n-2]$ to $[n]$

$\textsc{onto}(f) = 1$ iff f is surjective

encode as a Boolean function of $N = (2n-2)\log n$ bits

$$x^b = \begin{cases} x & \text{if } b = 1 \\ \bar{x} & \text{if } b = 0 \end{cases}$$

Depth-3 formula of size $\tilde{\Theta}(N^2)$: $\textsc{onto}(f) = \bigwedge_{j\in[n]} \bigvee_{i\in[2n-2]} \bigwedge_{\ell=0}^{\log n - 1} f(i)_\ell^{j_\ell}$

Proposition [BM 10]: The query complexity of $\textsc{onto}_N$ is $\Omega(N/\log N)$.

Using this in place of PARITY gives the same lower bounds for depth-3 formulas, up to a log factor.

Depth-2 formulas

Element distinctness

Given $x_1, \ldots, x_n \in [n]$, does there exist $i \neq j$ with $x_i = x_j$?

Depth-2 formulas

Element distinctness

Given $x_1, \ldots, x_n \in [n]$, does there exist $i \neq j$ with $x_i = x_j$?

Query complexity $\Omega(n^{2/3})$ [Aaronson, Shi 02; Ambainis 05; Kutin 05]

Depth-2 formulas

Element distinctness

Given $x_1, \ldots, x_n \in [n]$, does there exist $i \neq j$ with $x_i = x_j$?

Query complexity $\Omega(n^{2/3})$ [Aaronson, Shi 02; Ambainis 05; Kutin 05]

Encode as a Boolean function of $N = n \log n$ bits

Depth-2 formulas

Element distinctness

Given $x_1, \ldots, x_n \in [n]$, does there exist $i \neq j$ with $x_i = x_j$?

Query complexity $\Omega(n^{2/3})$ [Aaronson, Shi 02; Ambainis 05; Kutin 05]

Encode as a Boolean function of $N = n \log n$ bits

Depth-2 circuit of size $O(n^3)$:

$$\mathrm{ED}_N(x) = \bigvee_{i,j,k \in [n]} \bigwedge_{\ell=1}^{\log n} (x_i)_\ell^{k_\ell} \wedge (x_j)_\ell^{k_\ell}$$

Depth-2 formulas

Element distinctness

Given $x_1, \ldots, x_n \in [n]$, does there exist $i \neq j$ with $x_i = x_j$?

Query complexity $\Omega(n^{2/3})$ [Aaronson, Shi 02; Ambainis 05; Kutin 05]

Encode as a Boolean function of $N = n \log n$ bits

Depth-2 circuit of size $O(n^3)$:

$$\mathrm{ED}_N(x) = \bigvee_{i,j,k \in [n]} \bigwedge_{\ell=1}^{\log n} (x_i)_\ell^{k_\ell} \wedge (x_j)_\ell^{k_\ell}$$

Using composition to produce a circuit of size n gives a lower bound of $\tilde{\Omega}(n^{5/9}) = \Omega(n^{0.555})$.

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product: $(AB)_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product: $(AB)_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$

Claim: Checking whether $C = AB$ requires $\Omega(n^{1.055})$ queries to the entries of A, B, C.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product: $(AB)_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$

Claim: Checking whether $C = AB$ requires $\Omega(n^{1.055})$ queries to the entries of A, B, C.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Matrix-vector product verification: check whether $Av = 1$ (A fixed, v given by a black box)

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product: $(AB)_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$

Claim: Checking whether $C = AB$ requires $\Omega(n^{1.055})$ queries to the entries of A, B, C.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Matrix-vector product verification: check whether $Av = 1$ (A fixed, v given by a black box)

Formula: $\bigvee_i \bigwedge_j A_{ij} v_j$ which is a generic monotone depth-2 circuit

$\Rightarrow$ lower bound of $\tilde{\Omega}(n^{5/9}) = \Omega(n^{0.555})$

Boolean matrix product verification

Boolean semiring: "sum" is OR, "product" is AND

Boolean matrix product: $(AB)_{ij} = \bigvee_k A_{ik} \wedge B_{kj}$

Claim: Checking whether $C = AB$ requires $\Omega(n^{1.055})$ queries to the entries of A, B, C.

Best known upper bound is $O(n^{3/2})$ [Buhrman, Spalek 06]; their techniques give a linear lower bound.

Matrix-vector product verification: check whether $Av = 1$ (A fixed, v given by a black box)

Formula: $\bigvee_i \bigwedge_j A_{ij} v_j$ which is a generic monotone depth-2 circuit

$\Rightarrow$ lower bound of $\tilde{\Omega}(n^{5/9}) = \Omega(n^{0.555})$

$AB = J$ is the logical AND of n instances of the above problem

$\Rightarrow$ lower bound of $\tilde{\Omega}(\sqrt{n} \cdot n^{5/9}) = \tilde{\Omega}(n^{19/18}) = \Omega(n^{1.055})$

Formula gate count lower bounds

Formula gate count lower bounds

Read-many formula evaluation algorithm:

The quantum query complexity of evaluating a formula f with n inputs and G gates is $Q(f) = O(n^{1/2}G^{1/4})$.

Formula gate count lower bounds

Read-many formula evaluation algorithm:

The quantum query complexity of evaluating a formula f with n inputs and G gates is $Q(f) = O(n^{1/2}G^{1/4})$.

Corollary: Any formula representing a function f with n inputs requires $\Omega(Q(f)^4/n^2)$ gates.

Formula gate count lower bounds

Read-many formula evaluation algorithm:

The quantum query complexity of evaluating a formula f with n inputs and G gates is $Q(f) = O(n^{1/2} G^{1/4})$.

Corollary: Any formula representing a function f with n inputs requires $\Omega(Q(f)^4/n^2)$ gates.

For example, any formula for PARITY must have $\Omega(n^2)$ gates.

Formula gate count lower bounds

Read-many formula evaluation algorithm:
The quantum query complexity of evaluating a formula f with n inputs and G gates is $Q(f) = O(n^{1/2}G^{1/4})$.

Corollary: Any formula representing a function f with n inputs requires $\Omega(Q(f)^4/n^2)$ gates.

For example, any formula for PARITY must have $\Omega(n^2)$ gates.

Since $G < S$, this improves the classic result that the formula size of PARITY is $\Omega(n^2)$ [Khrapchenko 71].

Lower bounds on formula gate count of AC^0

Problem: How efficiently can we reexpress a given constant-depth circuit as a formula?

Lower bounds on formula gate count of AC^0

Problem: How efficiently can we reexpress a given constant-depth circuit as a formula?

Prior work [Nechiporuk 66, Jukna 12]: There is a constant-depth circuit of linear size such that any formula for the same function has size at least $n^{2-o(1)}$.

Lower bounds on formula gate count of AC^0

Problem: How efficiently can we reexpress a given constant-depth circuit as a formula?

Prior work [Nechiporuk 66, Jukna 12]: There is a constant-depth circuit of linear size such that any formula for the same function has size at least $n^{2-o(1)}$.

We show that there is a constant-depth circuit of linear size that requires $\Omega(n^{2-\epsilon})$ *gates* to express as a formula.

Lower bounds on formula gate count of AC^0

Problem: How efficiently can we reexpress a given constant-depth circuit as a formula?

Prior work [Nechiporuk 66, Jukna 12]: There is a constant-depth circuit of linear size such that any formula for the same function has size at least $n^{2-o(1)}$.

We show that there is a constant-depth circuit of linear size that requires $\Omega(n^{2-\epsilon})$ *gates* to express as a formula.

Main idea:

- ONTO has query complexity $\tilde{\Omega}(n)$, circuit size $\tilde{O}(n^2)$
- Recursively composing ONTO with itself gives a circuit with smaller size but nearly the same query complexity

Open problems

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Possible candidate for an improved lower bound:

$$\bigvee_{\ell \in L} \bigwedge_{i \in \ell} x_i \qquad L = \text{set of lines in a finite projective plane}$$

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Possible candidate for an improved lower bound:

$$\bigvee_{\ell \in L} \bigwedge_{i \in \ell} x_i \qquad L = \text{set of lines in a finite projective plane}$$

Formula evaluation upper/lower bounds taking other properties into account (beyond number of inputs, size, gate count, depth)

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Possible candidate for an improved lower bound:

$$\bigvee_{\ell \in L} \bigwedge_{i \in \ell} x_i \qquad L = \text{set of lines in a finite projective plane}$$

Formula evaluation upper/lower bounds taking other properties into account (beyond number of inputs, size, gate count, depth)

Circuit evaluation

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Possible candidate for an improved lower bound:

$$\bigvee_{\ell \in L} \bigwedge_{i \in \ell} x_i \qquad L = \text{set of lines in a finite projective plane}$$

Formula evaluation upper/lower bounds taking other properties into account (beyond number of inputs, size, gate count, depth)

Circuit evaluation

Upper/lower bounds as a function of number of inputs, size, depth

Open problems

Tighter bounds for evaluating depth-2 formulas (= circuits)

Possible candidate for an improved lower bound:

$$\bigvee_{\ell \in L} \bigwedge_{i \in \ell} x_i \qquad L = \text{set of lines in a finite projective plane}$$

Formula evaluation upper/lower bounds taking other properties into account (beyond number of inputs, size, gate count, depth)

Circuit evaluation

Upper/lower bounds as a function of number of inputs, size, depth

Graph collision as a depth-2 circuit of quadratic size or a depth-3 circuit of linear size