Almost-Everywhere Circuit Lower Bounds from Circuit-Analysis Algorithms

Ryan Williams
MIT

But all “heavy-lifting” done by: Xin Lyu (Tsinghua) and Lijie Chen (MIT)

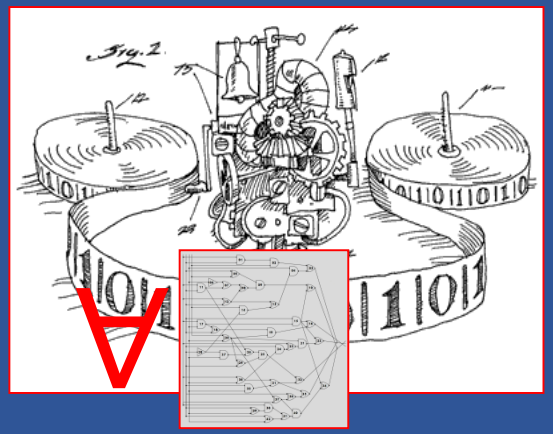
Outline

- Prior Work and a “Subtle” Issue
- What We Do
- A Little About How We Do It
- Conclusion

Algorithmic Approach to Lower Bounds:
Interesting circuit-analysis algorithms
tell us about the *limitations* of circuits in modeling algorithms

SAT? *YES/NO*

∃

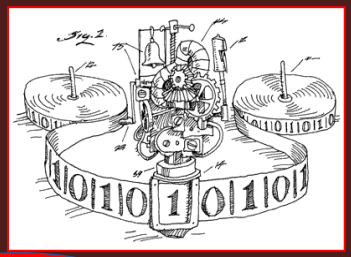


∀

“Non-Trivial” Circuit Analysis Algorithm
(beating brute force)

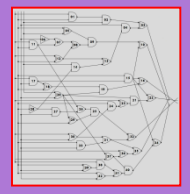
Inherently non-relativizing approach

∃ ”interesting” f

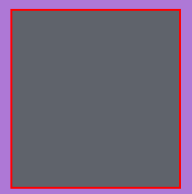


∀

Circuit Lower Bounds



≠



Circuits are **not** “black-boxes” to algs!

Turing Machine drawing by Tom Dunne for American Scientist

Circuit-Analysis Problem #1: Generalized Circuit Satisfiability

Let $\mathcal{C}$ be a class of Boolean circuits

$\mathcal{C}$ = {formulas}, $\mathcal{C}$ = {arbitrary circuits}, $\mathcal{C}$ = {3CNFs}

The $\mathcal{C}$-SAT Problem:

Given a circuit $\mathbf{K(x_1,\ldots,x_n)}$ from $\mathcal{C}$, is there an assignment $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that $\mathbf{K(a_1,\ldots,a_n) = 1}$?

A very "simple" circuit analysis problem

[CL'70s] **$\mathcal{C}$-SAT** is **NP-complete** for practically all interesting $\mathcal{C}$

$\mathcal{C}$-SAT is solvable in **$O(2^n\ |K|)$** time by brute force

Circuit-Analysis Problem #2: Gap Circuit Satisfiability

Let $\mathcal{C}$ be a class of Boolean circuits

$\mathcal{C}$ = {formulas}, $\mathcal{C}$ = {arbitrary circuits}, $\mathcal{C}$ = {3CNFs}

Gap-$\mathcal{C}$-SAT:

Given $K(x_1,\ldots,x_n)$ from $\mathcal{C}$, and the **promise** that either (a) $K \equiv 0$, or (b) $Pr_x[K(x) = 1] \geq 1/2$, **decide** which is true.

Even simpler! In randomized polynomial time

[Folklore?] **Gap-Circuit-SAT $\in$ P $\Rightarrow$ P = RP**

[Hirsch, Trevisan, …] **Gap-kSAT $\in$ P for all k**

Nontrivially Faster $\mathcal{C}$-SAT $\Longrightarrow$ Circuit Lower Bounds for $\mathcal{C}$

Slightly Faster Circuit-SAT
[R.W. '10,'11]

Deterministic algorithms for:

- Circuit SAT in $O(2^n/n^{10})$ time with n inputs and n^k gates, for all k
- Formula SAT in $O(2^n/n^{10})$ time
- $\mathcal{C}$-SAT in $O(2^n/n^{10})$ time
- Gap-$\mathcal{C}$-SAT in **$O(2^n/n^{10})$ time** on n^k size, for all k

(Easily solved w/ randomness!)

No "Circuits for NEXP"

Would imply:

- NEXP $\not\subset$ P/poly
- NEXP $\not\subset$ Poly-size formulas
- NEXP $\not\subset$ *poly-size* $\mathcal{C}$

NEXP $\not\subset$ *poly-size* $\mathcal{C}$

Concrete LBs:
$\mathcal{C}$ = ACC [W'11]
$\mathcal{C}$ = ACC of THR [W'14]

Even Faster SAT ⟹ Stronger Lower Bounds

Somewhat Faster Circuit SAT
[Murray-W. '18]

Det. algorithm *for some* $\epsilon > 0$:

- Circuit SAT in $O(2^{n-n^{\epsilon}})$ time with n inputs and $2^{n^{\epsilon}}$ gates
- Formula SAT in $O(2^{n-n^{\epsilon}})$ time
- C-SAT in $O(2^{n-n^{\epsilon}})$ time
- Gap-C-SAT in $O(2^{n-n^{\epsilon}})$ time on $2^{n^{\epsilon}}$ gates

No "Circuits for Quasi-NP"

Would imply:

- NTIME$[n^{polylog\, n}] \not\subset$ P/poly
- NTIME$[n^{polylog\, n}] \not\subset$ NC1
- NTIME$[n^{polylog\, n}] \not\subset C$
- NTIME$[n^{polylog\, n}] \not\subset C$

C = ACC of THR [MW'18]

Even Faster SAT ⟹ Stronger Lower Bounds

Note: Would refute Strong ETH!

Strongly believed to be true…

"Fine-Grained" SAT Algorithms [Murray-W. '18]	No "Circuits for NP"
Det. algorithm *for some* $\epsilon > 0$:	Would imply:
• Circuit SAT in $O(2^{(1-\epsilon)n})$ time on n inputs and $2^{\epsilon n}$ gates	• NP $\not\subset$ SIZE(n^k) for all k
• FormSAT in $O(2^{(1-\epsilon)n})$ time	• NP $\not\subset$ Formulas of size n^k
• C-SAT in $O(2^{(1-\epsilon)n})$ time	• NP $\not\subset$ C-SIZE(n^k) for all k
• Gap-C-SAT is in **$O(2^{(1-\epsilon)n})$ time** on $2^{\epsilon n}$ gates (Implied by **PromiseRP** in **P**)	NP $\not\subset$ C-SIZE(n^k) for all k

C = SUM of THR
C = SUM of ReLU
C = SUM of low-degree polys
[W'18]

Faster #SAT and CAPP ⟹ Average-Case Lower Bounds

[R.Chen-Oliveira-Santhanam'18, Chen-W'19, Chen'19, Chen-Ren '20]

Det. algorithm *for some* $\epsilon > 0$:

- **#Circuit SAT** in $O(2^{n-n^{\epsilon}})$ time with n inputs and $2^{n^{\epsilon}}$ gates
- **#Formula SAT** in $O(2^{n-n^{\epsilon}})$ time
- **#C-SAT** in $O(2^{n-n^{\epsilon}})$ time
- **C-CAPP** in $O(2^{n-n^{\epsilon}})$ time

Given a circuit of size **s**, approximate its *fraction* of SAT assignments to within **+- 1/s**

No Circuits for Computing Quasi-NP on Average

Would imply:

- NTIME[$n^{polylog\, n}$] can't be *(1/2 +1/poly)-approximated* in P/poly
- Inapproximability in NC1
- Inapproximability in C/poly

C = ACC of THR [Chen-Ren'20]

Faster #SAT and CAPP ⟹ Average-Case Lower Bounds

[R.Chen-Oliveira-Santhanam'18, Chen-W'19, Chen'19, Chen-Ren '20]

Det. algorithm *for some* $\epsilon > 0$:

- **#Circuit SAT** in $O(2^{n-n^{\epsilon}})$ time with n inputs and $2^{n^{\epsilon}}$ gates
- **#Formula SAT** in $O(2^{n-n^{\epsilon}})$ time
- **#C-SAT** in $O(2^{n-n^{\epsilon}})$ time
- **C-CAPP** in $O(2^{n-n^{\epsilon}})$ time

Given a circuit of size **s**, approximate its *fraction* of SAT assignments to within **+- 1/s**

No Circuits for Computing Quasi-NP on Average

Would imply:

- NTIME[$n^{polylog\, n}$] can't be *(1/2 +1/poly)-approximated* in P/poly

There is an $f \in$ **NTIME[$n^{polylog\, n}$]** such that, for **infinitely many** n, every poly(n)-size circuit C fails to compute f_n on more than $\left(\frac{1}{2} + \frac{1}{\mathrm{poly}(n)}\right) 2^n$ inputs.

A Subtle (But Important) Issue!

When we prove statements like NEXP $\not\subset$ ACC^0 via circuit-analysis algorithms,

we end up showing that, for NEXP-complete problems such as Succinct3SAT,
there are infinitely many input lengths n such that Succinct3SAT fails to have the desired ACC circuits on length-n inputs.

Let $f: \{0,1\}^* \to \{0,1\}$ and let $f_n: \{0,1\}^n \to \{0,1\}$ be the restriction of f

An infinitely-often circuit lower bound only says "f_n doesn't have small circuits" for infinitely many n:

$f_1, f_2, f_3, f_4, \cdots, f_{100}, \cdots\cdots, f_{1000}, \cdots\cdots, f_{10000}, \cdots\cdots$

We would greatly prefer an "almost-everywhere" circuit lower bound,
which holds for all but finitely many inputs!

$f_1, f_2, f_3, f_4, \cdots, f_{100}, \cdots\cdots, f_{1000}, \cdots\cdots, f_{10000}, \cdots$

All of the classical circuit lower bounds from the 1980s
(PARITY $\not\in$ AC0, MAJORITY $\not\in$ AC0[2], etc.) yield *almost-everywhere* lower bounds.

A Subtle (But Important) Issue!

***Why* does the algorithmic approach only get infinitely-often lower bounds?**

Prior work relies on other lower bounds such as the *nondeterministic time hierarchy theorem* or *MA/1 circuit lower bounds*, and neither results are known to hold almost-everywhere.

If we knew (for example)

$$\text{NTIME}[2^n] \text{ is not } \textit{infinitely often} \text{ in } \text{NTIME}[2^n/poly(n)],$$

then we could conclude some kind of almost-everywhere lower bound.

But there are oracles relative to which NEXP is *infinitely often* in NP!
[Buhrman-Fortnow-Santhanam '09]

A Subtle (But Important) Issue!

Why **does the algorithmic approach only get infinitely-often lower bounds?**

Prior work relies on other lower bounds such as the *nondeterministic time hierarchy theorem* or *MA/1 circuit lower bounds*, and neither results are known t

If we knew

> **"There are functions in NTIME[2^n] that are so hard, no nondeterministic algorithm running in $2^n/poly(n)$ time can correctly compute the function on any *infinite sequence of* input lengths"**

NTIME[2^n] is not *infinitely often* in NTIME[$2^n/poly(n)$],

then we could conclude some kind of almost-everywhere lower bound.

But there are oracles relative to which NEXP is *infinitely often* in NP!
[Buhrman-Fortnow-Santhanam '09]

Outline

- Prior Work and a “Subtle” Issue
- What We Do
- A Little About How We Do It
- Conclusion

This Work:
Faster SAT $\Longrightarrow$ Almost-Everywhere Lower Bounds

[R.Chen-Oliveira-Santhanam'18, Chen-W'19, Chen'19, Chen-Ren '20]

Det. algorithm *for some* $\epsilon > 0$:

- ***C*-SAT** (or **Gap-*C*-SAT**) with n inputs and $s(n)^{O(1)}$ gates in $2^n/n^{\omega(1)}$ **time**
- **#*C*-SAT** (or ***C*-CAPP**) in **$O(2^{n-n^\epsilon})$ time** on 2^{n^ϵ} gates

Given a circuit of size **s**, approximate its *fraction* of SAT assignments to within **+- 1/s**

A.E. Circuit Lower Bounds for E^{NP} on Average

There is an $f \in$ **TIME**$[2^{O(n)}]^{\text{SAT}}$ such that, for **all but finitely many** n, every s(n)-size circuit C fails to compute f_n on more than $\left(\frac{1}{2} + \frac{1}{s(n)}\right) 2^n$ inputs.

- **E^{NP} can't be $1/2 + 1/2^{n^{o(1)}}$-*approximated* with $2^{n^{o(1)}}$ *size* *C-circuits, for a.e. n***

Almost-everywhere average-case lower bounds for ACC of THR!

This Work:
Faster SAT $\Longrightarrow$ Almost-Everywhere Lower Bounds

[R.Chen-Oliveira-Santhanam'18, Chen-W'19, Chen'19, Chen-Ren '20]	A.E. Circuit Lower Bounds for E^{NP} on Average
Det. algorithm *for some* $\epsilon > 0$: • ***C*-SAT** (or **Gap-*C*-SAT**) with n inputs and $s(n)^{O(1)}$ gates in $2^n/n^{\omega(1)}$ **time**	**Would imply:** • E^{NP} **does not have** $s(n/2)$ ***size C-circuits, for almost every*** n
• **#*C*-SAT** (or ***C*-CAPP**) in **O(2^{n-n^ϵ}) time** (2^{n^ϵ} gates	• E^{NP} **can't be** $1/2 + 1/2^{n^{o(1)}}$**-*approximated* with** $2^{n^{o(1)}}$ ***size C-circuits, for a.e.*** n

Given a circuit of size **s**, approximate its *fraction* of SAT assignments to within **+- 1/s**

C = ACC of THR
$s(n) = 2^{n^{o(1)}}$

More Almost-Everywhere Goodness

In fact, we can extend all previous "E^{NP} lower bounds" proved via the algorithmic method to the **almost-everywhere** setting.

Strong average-case ACC^0 lower bounds:
Extends [Chen-W'19], [Chen-Ren'20]
with better inapproximability parameters

Correlation bounds: For all $\varepsilon > 0$, and for **all but finitely many** n, L_n cannot be $\frac{1}{2} + \frac{1}{2^{n^{\Omega(1)}}}$ approximated by $n^{1-\varepsilon}$-degree $\mathbf{F}_2$-polynomials.
Extends [Viola'20]

Probabilistic degree lower bounds:
There is an E^{NP} language L such that, for **all but finitely many** n, L_n does not have $o(n/\log^2 n)$-degree probabilistic $\mathbf{F}_2$-polynomials. Extends [Viola'20]

Rigid matrices in P^{NP}: There is a P^{NP} algorithm $\mathcal{A}$ such that, for **all but finitely many** n, $\mathcal{A}$ on input 1^n outputs an $n \times n$ matrix M_n satisfying $\mathcal{R}_{\left(2^{\log^{1-\varepsilon} n}\right)}(M_n) = \Omega(n^2)$.

Extends [Alman-C'19], [Bhangale-Harsha-Paradise-Tal'20]

Extension to Average Case

[Chen-Ren'20] There is a function f in $\mathrm{QuasiNP}$ such that, for **infinitely many** n, every ACC^0 circuit C of size $\mathrm{poly}(n)$ cannot $\left(\frac{1}{2}+\frac{1}{\mathrm{poly}(n)}\right)$-approximate f_n.

Theorem: There is an E^{NP} function f, such that for **all sufficiently large n**, f_n cannot be $\left(\frac{1}{2}+2^{-n^{o(1)}}\right)$-approximated by $2^{n^{o(1)}}$-size ACC^0 circuits.

"New" XOR Lemma: Suppose there is **no** $poly(s)$-size linear combination L of $\boldsymbol{C}$-circuits for f such that $E_x[|L(x)-f(x)|] < 1/10$. Then $f^{\oplus k}$ cannot be $\left(\frac{1}{2}+\frac{1}{s}\right)$-approximated by size-$s$ $\boldsymbol{C}$-circuits.

$(x_1, \ldots, x_k) \mapsto f(x_1) \oplus \cdots \oplus f(x_k)$

(Follows Levin's proof of the XOR Lemma)

Outline

- Prior Work and a “Subtle” Issue
- What We Do
- A Little About How We Do It
- Conclusion

A Little About How We Do It

- How NEXP $\not\subset$ ACC^0 Was Proved
- Another View of the Proof
- Extending to Almost-Everywhere

How **NEXP $\not\subset$ ACC^0** Was Proved

Let $\mathbb{C}$ be a "typical" circuit class (like ACC^0)

Thm A [W'11] (algorithm design → lower bounds)

If for all k, Gap-$\mathbb{C}$-SAT on n^k-size is in $O(2^n/n^k)$ time, then NEXP does not have poly-size $\mathbb{C}$-circuits.

Thm B [W'11] (algorithm)

$\exists\ \varepsilon$, #ACC^0-SAT on 2^{n^ε} size is in $O(2^{n-n^\varepsilon})$ time.

(Used a well-known representation of ACC^0 from 1990, that people long suspected should imply lower bounds)

Note that Theorem B gives a stronger algorithm than necessary in the hypothesis of Theorem A.

(This is the starting point of [Murray-W'18] which proves Quasi-NP lower bounds, and other subsequent work)

Idea of Theorem A

Let $\mathbb{C}$ be some circuit class (like ACC^0)

Thm A [W'11] (algorithm design ➔ lower bounds)
If for all k, Gap $\mathbb{C}$-SAT on n^k-size is in $O(2^n/n^k)$ time, then NEXP does not have poly-size $\mathbb{C}$-circuits.

Idea. Show that if we assume both:

(1) NEXP has poly-size $\mathbb{C}$-circuits,
AND

(2) a faster Gap $\mathbb{C}$-SAT algorithm

Then we can show NTIME$[2^n] \subseteq$ NTIME$[o(2^n)]$.
This *contradicts the nondeterministic time hierarchy: there's an L_{hard} in* NTIME$[2^n] \setminus$ NTIME$[o(2^n)]$

Proof Ideas in Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, AND

(2) there's a faster Gap $\mathbb{C}$-SAT algorithm

Show that NTIME[2^n] $\subseteq$ NTIME[o(2^n)] (contradiction)

Take any problem L in **nondeterministic 2^n time**
Given an input x, we decide L on x by:

1. Guessing a witness y of O(2^n) length.
2. Checking y is a witness for x in O(2^n) time.

Want to "speed-up" both parts 1 and 2, using the above assumptions

Proof Ideas in Theorem A

Idea. Assume

(1) NEXP has poly-size $\mathbb{C}$-circuits, AND

(2) there's a faster Gap $\mathbb{C}$-SAT algorithm

Show that NTIME[2^n] $\subseteq$ NTIME[o(2^n)]

Take any problem L in **nondeterministic 2^n time**
Given an input x, we decide L on x in a FASTER way:

1. **Use (1) to guess a witness y of o(2^n) length**
   (**Easy Witness Lemma [IKW02]**:
   if NEXP is in P/poly, then L has "small witnesses")
2. **Use (2) to check y is a witness for x in o(2^n) time**
   Technical: Use a highly-structured PCPs for NEXP [W'10, BV'14] to reduce the check to **Gap $\mathbb{C}$-SAT**

Extend to Almost-Everywhere?

Idea. **Assume**

(1) NEXP has poly-size $\mathbb{C}$-circuits, AND

(2) there's a faster Gap $\mathbb{C}$-SAT algorithm

Show that NTIME$[2^n] \subseteq$ NTIME$[o(2^n)]$?

INFINITELY OFTEN

Even if we could prove **NTIME$[2^n] \not\subset$ io-NTIME$[o(2^n)]$**, We still don't know how to complete step 1!

Take any problem L in **nondeterministic 2^n time**
Given an input x, we decide L on x in a FASTER way:

1. **Use (1) to guess a witness y of $o(2^n)$ length**
   (Infinitely-Often Easy Witness Lemma [???]:
   if NEXP is in io-P/poly, then L has "small witnesses" **?)**
2. **Use (2) to check y is a witness for x in $o(2^n)$ time**
   Technical: Use a highly-structured PCPs for NEXP [W'10, BV'14] to reduce the check to **Gap $\mathbb{C}$-SAT**

NT$[2^n] \not\subset$ io-NT$[o(2^n)]$ and $EXP^{NP} \subset$ io-$\mathbb{C}$ would imply our desired easy witnesses. **We could infer a contradiction!**

But such an NTIME hierarchy looks very hard to prove… what to do??

A Little About How We Do It

- How NEXP $\not\subset$ ACC^0 Was Proved
- Another View of the Proof
- Extending to Almost-Everywhere

Another View of the Proof

NTIME hierarchy ⇒ There is a function $f^{hard} \in \mathbf{NTIME}[2^n] \setminus \mathbf{NTIME}[2^n/n]$

Consider a "canonical" algorithm for f^{hard}:

$\mathcal{A}^{\mathbf{hard}}(x)$:
1. Guess a witness y of $O(2^n)$ length.
2. Check y is a witness for x in $O(2^n)$ time.

Consider an algorithm that tries to **"cheat"** in the computation of f^{hard}, by ***only*** verifying witnesses that are "compressible" by small ACC^0 circuits.

$\mathcal{A}^{cheat}(x)$:
1. Guess a $2^{n^{o(1)}}$-size ACC^0 circuit $C: \{0,1\}^n \to \{0,1\}$.
2. Check the **truth-table** of C is a witness for x, in $2^n/n^{\omega(1)}$ time.

NTIME hierarchy ⇒ $\mathcal{A}^{\mathrm{cheat}}$ **fails** to compute f^{hard} on infinitely many inputs

⇒ **There are infinitely many x such that $\mathcal{A}^{cheat}(x) = 0$ and $\mathcal{A}^{hard}(x) = 1$**

For each such x, every valid witness for $\mathcal{A}^{\mathrm{hard}}(x)$ is a hard function:
it **cannot** be computed by **small ACC^0 circuits**!

Another View of the Proof

There are infinitely many x such that $\mathcal{A}^{cheat}(x) = 0$ and $\mathcal{A}^{hard}(x) = 1$

For each such x, every valid witness for $\mathcal{A}^{\text{hard}}(x)$ is a hard function:
it cannot be computed by **small $\mathbf{ACC^0}$ circuits**!

Can use this to construct an E^{NP}/n algorithm with no small $\mathbf{ACC^0}$ circuits:

Input: an n-bit index $\boldsymbol{i} \in \{0,1\}^n$.
Advice: an n-bit string $\boldsymbol{x_n}$ such that $\mathcal{A}^{\text{cheat}}(\boldsymbol{x_n}) = 0, \mathcal{A}^{\text{hard}}(x_n) = 1$.
Output: Repeatedly call an NP oracle to find the lexicographically first witness $\boldsymbol{y}$ such that $\mathcal{A}^{hard}(\boldsymbol{x_n}) = 1$, and output the i-th bit of $\boldsymbol{y}$.

Finally, we can "remove" the advice by just considering an E^{NP} algorithm that takes $(\boldsymbol{i}, \boldsymbol{x})$ as input. This will also have no small $\mathbf{ACC^0}$ circuits.

What was gained by this perspective??? (We already had $\mathbf{NEXP}$ not in $\mathbf{ACC^0}$)

Vague Idea: Can we use another hierarchy? Can we "construct" these bad x_n?

A Little About How We Do It

- How NEXP $\not\subset$ ACC^0 Was Proved
- Another View of the Proof
- Extending to Almost-Everywhere

Extending to Almost-Everywhere

Recall: It is open if there is an $f \in \mathrm{NTIME}[2^n] \setminus \text{io-}\mathrm{NTIME}[o(2^n)]$

Idea: Start from a *restricted* almost-everywhere NTIME hierarchy

$\mathrm{NTIMEGUESS}[T(n), g(n)]$: languages that can be decided by nondeterministic algorithms running in $O(T(n))$ time and guessing at most $g(n)$ bits of witness.

Theorem [Fortnow-Santhanam 2016]

$$\mathrm{NTIME}[T(n)] \not\subseteq \text{io-}\mathrm{NTIMEGUESS}[o(T(n)), o(n)]$$

For time-constructible $T(n)$, there's a function decidable in $O(T(n))$ nondeterministic time that cannot be decided, even infinitely often, by any $o(T(n))$-time algorithm using $o(n)$ bits of guessing.

Does it Just Work??

[FS'16] There is a function $f^{hard} \in \mathbf{NTIME}[n^k] \setminus \mathbf{io\text{-}NTIMEGUESS}[o(n^k), o(n)]$

Consider a "canonical" algorithm for f^{hard}:

$\mathcal{A}^{\mathbf{hard}}(x)$:
1. Guess a witness y of $O(n^k)$ length.
2. Check y is a witness for x in $O(n^k)$ time.

As before, we consider an algorithm that tries to **"cheat"** to compute f^{hard}…

Let $m = k\log(n)$.

$\mathcal{A}^{cheat}(x)$:

1. Guess a $2^{m^{o(1)}}$-size ACC^0 circuit $C: \{0,1\}^m \to \{0,1\}$.
2. Check the **truth-table** of C is a witness for x, in $o(2^m)$…

$2^{m^{o(1)}} \leq o(n)$

$o(2^m) \leq o(n^k)$

[FS'16] $\Rightarrow$ for a.e. n, $\mathcal{A}^{\mathrm{cheat}}$ **fails** to compute f^{hard} on some input of length n

$\Rightarrow$ For a.e. n, there's an $x \in \{0,1\}^n$ such that $\mathcal{A}^{cheat}(x) = 0$ and $\mathcal{A}^{hard}(x) = 1$

For each such x, every valid witness for $\mathcal{A}^{\mathrm{hard}}(x)$ is a hard function:
it cannot be computed by **small $\mathbf{ACC}^0$ circuits**!

Does it Just Work??

For a.e. n, there's an $x \in \{0,1\}^n$ such that $\mathcal{A}^{cheat}(x) = 0$ and $\mathcal{A}^{hard}(x) = 1$

For each such x, every valid witness for $\mathcal{A}^{\text{hard}}(x)$ is a hard function:
it cannot be computed by **small $\mathbf{ACC^0}$ circuits**!

What happens when we try the same E^{NP} algorithm again?

Input: an m-bit index $\boldsymbol{i \in \{0,1\}^m}$, recall $\boldsymbol{m = k\log(n)}$
Advice: an n-bit string $\boldsymbol{x_n}$ such that $\mathcal{A}^{hard}(x_n) = 1$.
Output: Repeatedly call an **NP** oracle to find the lexicographically first witness $\boldsymbol{y}$ such that $\mathcal{A}^{hard}(x_n) = 1$, and output the i-th bit of $\boldsymbol{y}$.

Advice has length $n = 2^{m/k}$ (!!)

Now the advice is *insanely* long! We can't just remove it, as before!
(And of course there's a function in $E^{NP}/2^{n/k}$ without small ACC circuits…)

But *now,* the *construction* of such inputs x_n becomes an important problem!

If we could construct these "bad" x_n in E^{NP} (given input 1^m) we'd be done!

One More (New!) Ingredient

Theorem: [Fortnow-Santhanam 2016]
There's an $f^{\text{hard}} \in \text{NTIME}[T(n)] \setminus \text{io-NTIMEGUESS}\left[o\left(T(n)\right), o(n)\right]$

Theorem: There is a $\mathbf{DTIME}[\boldsymbol{n\,T(n)}]^{\mathbf{NP}}$ algorithm $\boldsymbol{R}$ (a **refuter**) such that for every $\text{NTIMEGUESS}\left[o\left(T(n)\right), o(n)\right]$ algorithm $\mathcal{A}$, $\boldsymbol{R}(1^n, \mathcal{A})$ outputs an n-bit x_n such that $f^{hard}(x_n) \neq \mathcal{A}(x_n)$, for **every sufficiently large** n.

Rough Idea: Using a variation on the proof of this time hierarchy, $\boldsymbol{R}$ does a "binary search" with its NP oracle, making $O(n)$ calls with queries of length about $O(T(n))$, to find a bad input x_n.

One More Try…

For a.e. n, there's an $x \in \{0,1\}^n$ such that $\mathcal{A}^{cheat}(x) = 0$ and $\mathcal{A}^{hard}(x) = 1$

For each such x, every valid witness for $\mathcal{A}^{\text{hard}}(x)$ is a hard function:
it cannot be computed by **small $\mathbf{ACC^0}$ circuits**!

The $\mathbf{E^{NP}}$ algorithm computing an almost-everywhere hard function:

Input: m-bit index $\boldsymbol{i \in \{0,1\}^m}$, recall $\boldsymbol{m = k\log(n)}$
Algorithm: Set $n \approx 2^{m/k}$ and run refuter $\boldsymbol{R(1^n, \mathcal{A}^{cheat})}$ in $\mathbf{E^{NP}}$, obtaining (for all but finitely many n) an n-bit string $\boldsymbol{x_n}$ such that $\mathcal{A}^{\text{cheat}}(\boldsymbol{x_n}) \neq \mathcal{A}^{\text{hard}}(\boldsymbol{x_n})$.
Repeatedly call an $\mathbf{NP}$ oracle to find the lexicographically first witness $\boldsymbol{y}$ such that $\boldsymbol{\mathcal{A}^{hard}(x_n) = 1}$, and output the i-th bit of $\boldsymbol{y}$.

Conclusion: $\mathbf{E^{NP} \not\subset io\text{-}ACC^0}$

Conclusion

We have managed to prove several almost-everywhere lower bounds for functions in E^{NP}, even for the average case.

What about NEXP? Or Quasi-NP? Or NP?

Can we prove $\mathbf{NEXP} \not\subset \mathbf{io\text{-}ACC^0}$?

What other lower bounds can be made a.e.?
(e.g. $\Sigma_2 P \not\subset SIZE(n^k)$)

Thanks for watching!