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.

∃ ∀ "Non-Trivial" Circuit Analysis Algorithm (beating brute force)

SAT? YES/NO

Inherently non-relativizing approach

∃ "interesting" f

Circuit Lower Bounds

Circuits are not “black-boxes” to algs!
Circuit-Analysis Problem #1: Generalized Circuit Satisfiability

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

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

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

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

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

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

**Gap-C-SAT:**

Given $K(x_1, ..., x_n)$ 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 $\in \text{P} \Rightarrow \text{P} = \text{RP}$

[Hirsch, Trevisan, ...] Gap-$k$SAT $\in \text{P}$ for all $k$
Nontrivially Faster $\mathcal{C}$-SAT $\implies$ Circuit Lower Bounds for $\mathcal{C}$

<table>
<thead>
<tr>
<th>Slightly Faster Circuit-SAT [R.W. ’10,’11]</th>
<th>No “Circuits for NEXP”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic algorithms for:</td>
<td>Would imply:</td>
</tr>
<tr>
<td>• Circuit SAT in $O(2^n/n^{10})$ time</td>
<td>• NEXP $\not\subset$ P/poly</td>
</tr>
<tr>
<td>with $n$ inputs and $n^k$ gates, for all $k$</td>
<td>• NEXP $\not\subset$ Poly-size formulas</td>
</tr>
<tr>
<td>• Formula SAT in $O(2^n/n^{10})$ time</td>
<td>• NEXP $\not\subset$ poly-size $\mathcal{C}$</td>
</tr>
<tr>
<td>• $\mathcal{C}$-SAT in $O(2^n/n^{10})$ time</td>
<td></td>
</tr>
<tr>
<td>• Gap-$\mathcal{C}$-SAT in $O(2^n/n^{10})$ time</td>
<td>NEXP $\not\subset$ poly-size $\mathcal{C}$</td>
</tr>
<tr>
<td>on $n^k$ size, for all $k$</td>
<td></td>
</tr>
<tr>
<td><em>(Easily solved w/ randomness!)</em></td>
<td></td>
</tr>
</tbody>
</table>

Concrete LBs:
- $\mathcal{C} = ACC$
- $\mathcal{C} = ACC$ of THR
  [W’11]
- $\mathcal{C} = ACC$ of THR
  [W’14]
Even Faster SAT $\implies$ 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:
- $\text{NTIME}[n^{\text{polylog} n}] \not\subset \text{P/poly}$
- $\text{NTIME}[n^{\text{polylog} n}] \not\subset \text{NC1}$
- $\text{NTIME}[n^{\text{polylog} n}] \not\subset C$

$C = ACC$ of THR

[MW’18]
**“Fine-Grained” SAT Algorithms [Murray-W. ’18]**

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

- Circuit SAT in $O(2^{(1-\epsilon)n})$ time on $n$ inputs and $2^{\epsilon n}$ gates
- FormSAT in $O(2^{(1-\epsilon)n})$ time
- $C$-SAT in $O(2^{(1-\epsilon)n})$ time
- Gap-$C$-SAT is in $O(2^{(1-\epsilon)n})$ time on $2^{\epsilon n}$ gates

(Implied by PromiseRP in P)

**Note:** Would refute Strong ETH!

**Strongly believed to be true...**

**No “Circuits for NP”**

Would imply:

- NP $\not\subset$ SIZE($n^k$) for all $k$
- NP $\not\subset$ Formulas of size $n^k$
- NP $\not\subset$ $C$-SIZE($n^k$) for all $k$

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 \implies \text{Average-Case Lower Bounds}

\begin{itemize}
\item [\text{Det. algorithm for some } \epsilon > 0:]
  \begin{itemize}
  \item \#\text{Circuit SAT} in \(O(2^n - n^\epsilon)\) time with \(n\) inputs and \(2^n\epsilon\) gates
  \item \#\text{Formula SAT} in \(O(2^n - n^\epsilon)\) time
  \item \#\text{C-SAT} in \(O(2^n - n^\epsilon)\) time
  \item \text{C-CAPP} in \(O(2^n - n^\epsilon)\) time
  \end{itemize}
\end{itemize}

\begin{itemize}
\item \text{No Circuits for Computing Quasi-NP on Average}
\item Would imply:
  \begin{itemize}
  \item NTIME\([n^{\text{polylog}}]\) can’t be \((1/2 + 1/\text{poly})\)-approximated in P/poly
  \item Inapproximability in NC1
  \item Inapproximability in \(C/\text{poly}\)
  \end{itemize}
\end{itemize}

\text{Given a circuit of size } s, \text{ approximate its fraction of SAT assignments to within } \pm 1/s
Faster #SAT and CAPP $\implies$ Average-Case Lower Bounds

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

No Circuits for Computing Quasi-NP on Average

Would imply:
- NTIME[$n^{\text{polylog } n}$] can’t be $\frac{1}{2} + \frac{1}{\text{poly}}$-approximated in P/poly
- Inapproximability in NC1
- Inapproximability in C/poly

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

Given a circuit of size $s$, approximate its fraction of SAT assignments to within $\pm \frac{1}{s}$
A Subtle (But Important) Issue!

When we prove statements like $\text{NEXP} \not\subset \text{ACC}^0$ 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 $n$:

\[
f_1, f_2, f_3, f_4, \ldots, \text{\ding{55}}, \ldots, \text{\ding{55}}, f_{100}, \ldots, \text{\ding{55}}, f_{1000}, \ldots, \text{\ding{55}}, f_{10000}, \ldots, \text{\ding{55}}
\]

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

\[
f_1, f_2, f_3, f_4, \ldots, f_{100}, \ldots, f_{1000}, \ldots, f_{10000}, \ldots
\]

All of the classical circuit lower bounds from the 1980s (PARITY $\not\in \text{AC0}$, MAJORITY $\not\in \text{AC0}[2]$, 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)

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

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

But there are oracles relative to which \( \text{NEXP} \) is *infinitely often* in \( \text{NP} \)!

[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 to hold almost everywhere.

If we knew (for example) $\text{NTIME}[2^n]$ is not infinitely often in $\text{NTIME}[2^n / \text{poly}(n)]$, then we could conclude some kind of almost-everywhere lower bound.

But there are oracles relative to which $\text{NEXP}$ 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 ⇒ Almost-Everywhere Lower Bounds

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 $2^n$ gates\n
A.E. Circuit Lower Bounds for $E^{NP}$ on Average

There is an $f \in \text{TIME}[2^{O(n)}]^{SAT}$ 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 \implies \text{Almost- Everywhere Lower Bounds}

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

\textbf{Given a circuit of size } s, \textbf{approximate its } fraction \textbf{ of SAT assignments to within } +1/s
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 $\epsilon > 0$, and for all but finitely many $n$, $L_n$ cannot be $\frac{1}{2} + \frac{1}{2^{n\Omega(1)}}$ approximated by $n^{1-\epsilon}$-degree $F_2$-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 $F_2$-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 $\mathcal{R}_{2^{\log^{1-\epsilon} n}}(M_n) = \Omega(n^2)$.
Extends [Alman-C’19], [Bhangale-Harsha-Paradise-Tal’20]
Theorem: There is an $E^{NP}$ function $f$, such that for all sufficiently large $n$, $f_n$ cannot be approximated by $2^{n^{o(1)}}$-size $ACC^0$ circuits.

"New" XOR Lemma: Suppose there is no $poly(s)$-size linear combination $L$ of $C$-circuits for $f$ such that $E_x[|L(x) - f(x)|] < 1/10$. Then $f \oplus k$ cannot be approximated by size-$s$ $C$-circuits.
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 \( \text{NEXP} \not\subset \text{ACC}^0 \) Was Proved
• Another View of the Proof
• Extending to Almost-Everywhere
How $\text{NEXP} \not\subset \text{ACC}^0$ Was Proved

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

**Thm A [W’11]** (algorithm design $\rightarrow$ lower bounds)

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

**Thm B [W’11]** (algorithm)

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

(Used a well-known representation of $\text{ACC}^0$ 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 $\text{ACC}^0$)

**Thm A** [W’11] (algorithm design $\Rightarrow$ 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 $\text{NTIME}[2^n] \subset \text{NTIME}[o(2^n)]$.

This contradicts the nondeterministic time hierarchy: there’s an $L_{\text{hard}}$ in $\text{NTIME}[2^n] \setminus \text{NTIME}[o(2^n)]$.
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 $\text{NTIME}[2^n] \subseteq \text{NTIME}[o(2^n)]$ (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 \( \text{NTIME}[2^n] \subseteq \text{NTIME}[o(2^n)] \)

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 $\text{NTIME}[2^n] \subseteq \text{NTIME}[o(2^n)]$?

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
   (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

Even if we could prove $\text{NTIME}[2^n] \not\subseteq \text{io-NTIME}[o(2^n)]$,
We still don’t know how to complete step 1!

NT$[2^n] \not\subseteq \text{io-NT}[o(2^n)]$ and $\text{EXP}^\text{NP} \subseteq \text{io-}\mathbb{C}$
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 $\text{NEXP} \not\subseteq \text{ACC}^0$ Was Proved

• Another View of the Proof

• Extending to Almost-Everywhere
Another View of the Proof

**NTIME hierarchy** \(\Rightarrow\) There is a function \(f^\text{hard} \in \text{NTIME}[2^n] \setminus \text{NTIME}[2^n/n]\)

Consider a “canonical” algorithm for \(f^\text{hard}^\):  

\(\mathcal{A}^\text{hard}(x):\)
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 \(f^\text{hard}\), by **only** verifying witnesses that are “compressible” by small \(\text{ACC}_0\) circuits.

\(\mathcal{A}^\text{cheat}(x):\)
1. Guess a \(2^{n^{o(1)}}\)-size \(\text{ACC}_0\) circuit \(C: \{0,1\}^n \rightarrow \{0,1\}\).
2. Check the **truth-table** of \(C\) is a witness for \(x\), in \(2^n/n^{o(1)}\) time.

**NTIME hierarchy** \(\Rightarrow\) \(\mathcal{A}^\text{cheat}\) fails to compute \(f^\text{hard}\) on infinitely many inputs 

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

For each such \(x\), every valid witness for \(\mathcal{A}^\text{hard}(x)\) is a hard function: it **cannot** be computed by **small** \(\text{ACC}_0\) circuits!
Another View of the Proof

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

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

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

**Input:** an $n$-bit index $i \in \{0, 1\}^n$.

**Advice:** an $n$-bit string $x_n$ such that $A^{\text{cheat}}(x_n) = 0, A^{\text{hard}}(x_n) = 1$.

**Output:** Repeatedly call an NP oracle to find the lexicographically first witness $y$ such that $A^{\text{hard}}(x_n) = 1$, and output the $i$-th bit of $y$.

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

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

Vague Idea: Can we use another hierarchy? Can we “construct” these bad $x_n$?
A Little About How We Do It

• How \( \text{NEXP} \not\subset \text{ACC}^0 \) Was Proved
• Another View of the Proof
• Extending to Almost-Everywhere
Extending to Almost-Everywhere

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

Idea: Start from a restricted almost-everywhere NTIME hierarchy

\( \text{NTIMEGUESS}[T(n), g(n)]: \) 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]**

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

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

Consider a “canonical” algorithm for \( f^{\text{hard}} \):

\[ \mathcal{A}^{\text{hard}}(x): \]
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 \( f^{\text{hard}} \)...

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

\[ \mathcal{A}^{\text{cheat}}(x): \]
1. Guess a \( 2^{m^{o(1)}} \)-size ACC\(^0\) circuit \( C: \{0,1\}^m \rightarrow \{0,1\} \).
2. Check the truth-table of \( C \) is a witness for \( x \), in \( o(2^m) \) time.

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

\Rightarrow \text{For a.e. } n, \text{ there’s an } x \in \{0,1\}^n \text{ such that } \mathcal{A}^{\text{cheat}}(x) = 0 \text{ and } \mathcal{A}^{\text{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 ACC\(^0\) circuits!
Does it Just Work??

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

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

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

**Input:** an \( m \)-bit index \( i \in \{0, 1\}^m \), recall \( m = k \log(n) \)

**Advice:** an \( n \)-bit string \( x_n \) such that \( \mathcal{A}^{\text{cheat}}(x_n) = 0 \) and \( \mathcal{A}^{\text{hard}}(x_n) = 1 \)

**Output:** Repeatedly call an \( \text{NP} \) oracle to find the lexicographically first witness \( y \) such that \( \mathcal{A}^{\text{hard}}(x_n) = 1 \), and output the \( i \)-th bit of \( y \).

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 \( \text{ACC} \) 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!
Rough Idea: Using a variation on the proof of this time hierarchy, $R$ 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 $x_n$. 

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

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

Rough Idea: Using a variation on the proof of this time hierarchy, $R$ 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 $x_n$. 
For a.e. $n$, there’s an $x \in \{0, 1\}^n$ such that $A^{\text{cheat}}(x) = 0$ and $A^{\text{hard}}(x) = 1$

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

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

**Input:** $m$-bit index $i \in \{0, 1\}^m$, recall $m = k \log(n)$

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

**Conclusion:** $E^{\text{NP}} \not\subseteq \text{io-ACC}^0$
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 $\text{NEXP} \not\subseteq \text{io-ACC}^0$?

What other lower bounds can be made a.e.?

(e.g. $\Sigma_2P \not\subseteq \text{SIZE}(n^k)$)

Thanks for watching!