Lower Bounds from Algorithm Design: An Overview

Ryan Williams
MIT
Course Announcement
CS294-152. Lower Bounds: Beyond the Boot Camp

Soda 405
Mondays 4:00pm to 6:30pm (with a break in the middle)

*first lecture is next week*
Outline

• A High-Level View

• Algorithms versus Boolean Circuits

• Circuit Analysis => Circuit Lower Bounds

• Some Details and Some Progress:
  NQP (Quasi-NP) is not in ACC
  NP doesn’t have small depth-two neural nets
High-level view of algorithms and complexity

- Algorithm designers

- Complexity theorists

- What makes some problems easy to solve? When can we find an efficient algorithm?

- What makes other problems difficult? When can we prove that a problem is not easy?

When can we prove a Lower Bound on the resources (time/space/communication/etc) needed to solve a problem?
The tasks of the algorithm designer and the complexity theorist appear to be polar opposites.

- Algorithm designers prove upper bounds
- Complexity theorists prove lower bounds

Furthermore, it’s generally believed that Algorithm Design is easier than Lower Bounds

- In Algorithm Design: find one clever algorithm
- In Lower Bounds: must reason about “all possible” algorithms, and argue none of them work well

... but there are thousands of worst-case algorithms which analyze all possible finite objects of some kind...

My Opinion: This isn’t why lower bounds are hard!
Why are lower bounds hard to prove?

There are *many* known “no-go” theorems

- Relativization [70’s]
- Natural Properties [90’s]
- Algebrization [00’s]

**Summary:** The common proof techniques are not good enough to prove even weak lower bounds!

*Great pessimism in complexity theory*
How will we make progress?

There are *many* known “no-go” theorems

- Relativization [70’s]
- Natural Properties [90’s]
- Algebrization [00’s]

Summary: The common proof techniques are not good enough to prove even weak lower bounds!

*Great pessimism in complexity theory*

*Have to non-relativize, non-algebrize, and non-naturalize!*
One Direction for Progress: 
*Connect Algorithm Design to Lower Bounds*

Much more than *opposites*! 
There are deeper connections we are slowly uncovering.

**Thesis:** Designing Algorithms (in some sense) is equivalent to Proving Lower Bounds

A typical result in Algorithm Design:
"Here is an algorithm $A$ that solves the problem, on all possible instances of the problem"

A typical theorem from Lower Bounds:
"Here is a proof $P$ that the problem can’t be solved, by all possible algorithms of some type"

Meta-computation: 
Problems whose input is the code of an algorithm
A “Plan” For Proving Lower Bounds

Want to prove results of the form:

- Task A is impossible for computation model B

Find results showing (algorithm design $\Rightarrow$ lower bounds):

- Task A’ is possible for computation model B’
  $\Rightarrow$ Task A is impossible for computation model B

Then, use results from algorithm design to show:

- Task A’ is possible for computation model B’
Want to prove results of the form:

**Task A is impossible for computation model B**

Find results showing (algorithm design → lower bounds):

**Task A’ is possible for computation model B’**

Then, use results from algorithm design to show:

**Task A’ is possible for computation model B’**
A simple example from complexity theory:

If PSPACE = EXPTIME then PTIME ≠ PSPACE

PSPACE = problems solvable in polynomial space
PTIME = ... in polynomial time
EXPTIME = ... in exponential time

Proof: PTIME ≠ EXPTIME (time hierarchy theorem)
So PTIME = PSPACE implies PSPACE ≠ EXPTIME. QED

Many such results can be proved....
But they do not seem useful!
**Big Idea:** 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

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

Turing Machine drawing by Tom Dunne for American Scientist
Big Idea: Interesting circuit-analysis algorithms tell us about the *limitations* of circuits in modeling algorithms.

Goal: Algorithmic task A is impossible for “efficient” circuits (this is our model B).

Show: Non-trivial analysis of “efficient” circuits is possible with algorithms (model B’)

⇒ Algorithmic Task A is impossible for “efficient” circuits.

Show: Non-trivial analysis of “efficient” circuits is possible with algorithms.
Outline

• A High-Level View
• Algorithms versus Boolean Circuits
• Circuit Analysis => Circuit Lower Bounds
• Some Details and Some Progress
For every input length $n$, a circuit family has a circuit $C_n$ to be run on all inputs of length $n$

$P/poly = \{ f : \{0, 1\}^* \to \{0, 1\} \text{ computable by a circuit family } \{C_n\} \text{ such that } (\exists k \geq 1)(\forall n), \text{ the size of } C_n \text{ is at most } n^k \}$

Each circuit is “small” relative to its number of inputs

Circuit model has “programs with infinite-length descriptions”

The standard methods in computability theory are powerless...
Concrete limitations on computing within the known universe

"Any logic circuit solving most instances of my 1000-bit problem needs at least $10^{100}$ bits to be described"

Circuit Family = \{ $C_1$, $C_{10}$, $C_{100}$, $C_{1000}$ \}

\[ P/poly = \{ f : \{0, 1\}^* \rightarrow \{0, 1\} \text{ computable with a circuit family} \]
\[ \{ C_n \} \text{ such that } (\exists k \geq 1) (\forall n), \text{ the size of } C_n \text{ is at most } n^k \} \]

Why study this “infinite” model of computation?
1) Circuits could be easier to analyze than Turing machines!
2) Proving limitations on $P/poly$ is a step towards non-asymptotic complexity theory:

Universe stores $< 10^{80}$ bits  [Bekenstein ‘70s]  [Meyer-Stockmeyer ‘70s]
Algorithms versus Circuit Families

\[ P/\text{poly} = \{ f : \{0, 1\}^* \rightarrow \{0, 1\} \text{ computable with a circuit family} \{C_n\} \text{ such that } (\exists k \geq 1)(\forall n), \text{ the size of } C_n \text{ is at most } n^k \} \]

Most Boolean functions require huge circuits:

**Theorem [Shannon ‘49]** W.h.p., random \( f : \{0, 1\}^n \rightarrow \{0, 1\} \) needs circuits of size at least \( 2^n/n \)

**Theorem [Lupanov’58]** Every \( f \) has a circuit of size \( (1+o(1))2^n/n \)

Explicit (non-random) hard functions?

What “uniform” algorithms can be simulated in \( P/\text{poly} \)?

Can huge uniform classes (like \( \text{PSPACE, EXP, NEXP} \)) be simulated with small non-uniform classes (like \( P/\text{poly} \))?

The key obstacle: Non-uniformity can be very powerful!
What “uniform” algorithms can be simulated in P/poly?
Can huge uniform classes (like PSPACE, EXP, NEXP) be simulated with small non-uniform classes (like P/poly)?

RIDICULOUSLY OPEN: Is NEXP ⊆ P/poly?
Can all problems with exponentially-long answers checkable in exponential time be solved with polynomial-size circuit families?

Conjecture: NP ∉ P/poly (harder than P ≠ NP)

OPEN: NP ∉ SIZE(O(n))?
Best known: NP ∉ SIZE(5n), SIZE(3.01n)

Now, problems like NP ∉ SIZE(O(n)) may be attackable...(?)

Algorithms versus Circuit Families
Outline

• A High-Level View
• Algorithms versus Boolean Circuits
• Circuit Analysis => Circuit Lower Bounds
• Some Details and Some Progress
Generalized Circuit Satisfiability

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

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

The $\mathbf{C}$-SAT Problem:
Given a circuit $K(x_1, ..., x_n)$ from $\mathbf{C}$, is there an assignment $(a_1, ..., a_n) \in \{0,1\}^n$ such that $K(a_1, ..., a_n) = 1$?

A very “simple” circuit analysis problem!

[CL’70s] $\mathbf{C}$-SAT is $\mathbf{NP}$-complete for practically all interesting $\mathbf{C}$
$\mathbf{C}$-SAT is solvable in $O(2^n |K|)$ time by brute force
**Gap Circuit Satisfiability**

Let $C$ be a class of Boolean circuits

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

**Gap-C-SAT:**

Given $K(x_1,...,x_n)$ from $C$, 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?] If Gap-Circuit-SAT $\in P$ then $P = RP$

[Hirsch, Trevisan, ...] Gap-kSAT is P for all $k$
Faster **C-SAT** $\implies$ Circuit Lower Bounds for **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 with $n$ inputs and $n^k$ gates</td>
<td>• NEXP $\not\subset$ P/poly</td>
</tr>
<tr>
<td>• Formula SAT in $O(2^n/n^{10})$ time</td>
<td>• NEXP $\not\subset$ Poly-size formulas</td>
</tr>
<tr>
<td><strong>C</strong>-SAT in $O(2^n/n^{10})$ time</td>
<td>• NEXP $\not\subset$ poly-size <strong>C</strong></td>
</tr>
<tr>
<td>• Gap-<strong>C</strong>-SAT is in $O(2^n/n^{10})$ time on $n^k$ size</td>
<td>NEXP $\not\subset$ poly-size <strong>C</strong></td>
</tr>
</tbody>
</table>

(Easily solved w/ randomness!)

Concrete LBs
$C = ACC$ [W’11]
$C = ACC$ of THR [W’14]
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 $2^n$ gates
- Formula SAT in $O(2^n - n^\epsilon)$ time
- $C$-SAT in $O(2^n - n^\epsilon)$ time
- Gap-$C$-SAT is in $O(2^n - n^\epsilon)$ time on $2^n$ gates

No “Circuits for Quasi-NP”
Would imply:
- $\text{NTIME}[n^{\text{polylog } n}] \not\subset \text{P/poly}$
- $\text{NTIME}[n^{\text{polylog } n}] \not\subset \text{NC1}$
- $\text{NTIME}[n^{\text{polylog } n}] \not\subset C$

$C = \text{ACC of THR}$
[MW’18]
### Even Faster SAT $\implies$ Stronger Lower Bounds

#### “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)

#### No “Circuits for NP”

Would imply:
- $NP \not\subset \text{SIZE}(n^k)$ for all $k$
- $NP \not\subset \text{Formulas of size } n^k$
- $NP \not\subset C$-$\text{SIZE}(n^k)$ for all $k$

$C = \text{SUM of THR}$
$C = \text{SUM of ReLU}$
$C = \text{SUM of POL}$

**Note:** Would refute Strong ETH!

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

[W’18]
Outline

• A High-Level View
• Algorithms versus Boolean Circuits
• Circuit Analysis => Circuit Lower Bounds
• Some Details and Some Progress
Some Lower Bounds by Algorithm Design

\( \textbf{ACC}^0 \): circuits of \textit{polynomial} size and \textit{constant} depth, with AND, OR, and MODm gates for some constant m. \( \textbf{ACC}^0 \subset \text{P/poly} \), probably a proper subset!

\textit{Annoying Circuit Class} to prove lower bounds for, proposed in 1986 (and it is the 0\textsuperscript{th} such class)

\textbf{Thm [R.W.'11]}: \( \text{NEXP} \not\subset \text{ACC}^0 \)

\textbf{Thm [Murray-W’18]}: \( \text{NTIME}[n^{poly(\log n)}] \not\subset \text{ACC}^0 \) of \textbf{THR}

\( \textbf{ACC} \circ \textbf{THR} \): \textit{Annoying Circuits with Linear Threshold Gates} at the bottom
Progress Report

[W’14, Murray-W’18] Quasi-NP does not have ACC $\circ$ THR circuits of polynomial size

SAT algorithm uses a new depth-two representation of ACC $\circ$ THR
and fast rectangular matrix multiplication to evaluate the representation quickly

Improving the lower bounds to multiple layers of THR gates is an open frontier:

[Tamaki’16, Alman-Chan-W’16] $E^{\text{NP}}$ does not have ACC $\circ$ THR $\circ$ THR circuits of subquadratic size

Uses recent probabilistic polynomials for THR [Srinivasan’13, Alman-W’15]

Open: Quasi-NP does not have THR $\circ$ THR circuits of subquadratic size

[S.Chen-Papakonstantinou’16] Better size-depth tradeoff lower bound for NEXP vs ACC

[R.Chen-Oliveira-Santhanam’18] Average Case: NEXP doesn’t have poly-size ACC circuits

computing a $\frac{1}{2} + \frac{1}{\text{poly}(\log n)}$ fraction of $n$-bit inputs correctly

Carefully applies coding-theoretic techniques on top of the framework

[W’18] NP does not have $O(n^{100})$-size depth-two neural networks

with sign activation function, nor with ReLU activation functions

At the heart: [Horowitz-Sahni 70s] Counting subset sum solutions on $n$ items is in $\sim 2^{n/2}$ time!

New lower bounds from an old algorithm!
Progress Report

**[W’14, Murray-W’18]** Quasi-NP does not have ACC $\circ$ THR circuits of polynomial size

SAT algorithm uses a new depth-two representation of ACC $\circ$ THR and *fast rectangular matrix multiplication* to evaluate the representation quickly.

Improving the lower bounds to multiple layers of THR gates is an open frontier:

**[Tamaki’16, Alman-Chan-W’16]** $E^{NP}$ does not have ACC $\circ$ THR $\circ$ THR circuits of subquadratic size.

Uses recent probabilistic polynomials for THR [Srinivasan’13, Alman-W’15]

Open: Quasi-NP does not have THR $\circ$ THR circuits of subquadratic size

**[S.Chen-Papakonstantinou’16]** Better size-depth tradeoff lower bound for NEXP vs ACC

**[R.Chen-Oliveira-Santhanam’18]** Average Case: NEXP doesn’t have poly-size ACC circuits

computing a $\frac{1}{2} + \frac{1}{\text{poly}(\log n)}$ fraction of $n$-bit inputs correctly.

Carefully applies coding-theoretic techniques on top of the framework.

**[W’18]** NP does not have $O(n^{100})$-size depth-two neural networks with sign activation function, nor with ReLU activation functions.

At the heart: [Horowitz-Sahni 70s] Counting subset sum solutions on $n$ items is in $\sim 2^{n/2}$ time!

New lower bounds from an old algorithm!
Lower Bounds for NEXP, Quasi-NP, and NP From Nontrivial Gap-SAT Algorithms
How \( \text{NEXP} \not\subseteq \text{ACC}^0 \) Was Proved

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

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

If for all \( k \), \( \text{Gap-} \mathcal{C}\text{-SAT} \) on \( n^k \)-size is in \( O(2^n/n^k) \) time, then \( \text{NEXP} \) does not have poly-size \( \mathcal{C} \)-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 the inefficiency!

Theorem B gives a much stronger algorithm than is necessary in Theorem A.

This is exactly the starting point of [Murray-W’18]...
Idea of Theorem A

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

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

Idea. Show that if we assume both:

(1) $\text{NEXP}$ has poly-size $\mathcal{C}$-circuits, AND
(2) a faster Gap $\mathcal{C}$-SAT algorithm

Then we can show $\text{NTIME}[2^n] \subseteq \text{NTIME}[o(2^n)]$ (contradicts the nondeterministic time hierarchy!)
Proof Ideas in Theorem A

Idea. Assume

(1) NEXP has poly-size C-circuits, AND
(2) there’s a faster Gap 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 “compute” \( L \) 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 \(\mathcal{C}\)-circuits, AND
2. there’s a faster Gap \(\mathcal{C}\)-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 will “compute” \(L\) on \(x\) by:

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 \(\mathcal{C}\)-SAT**
Proof Ideas in Theorem A

Idea. Assume

(1) NEXP has poly-size $C$-circuits, AND
(2) there’s a faster Gap $C$-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 will “compute” $L$ on $x$ by:

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 $C$-SAT
Guessing Short Witnesses

1. Guess a witness $y$ of $O(2^n)$ length.

**Definition.** An NTIME[$2^n$] problem $L$ has *easy witnesses* if

\[ \exists c \geq 1, \forall \text{ Verifiers } V \text{ for } L, \text{ if } \exists y \in \{0, 1\}^{2|x|+d} \text{ s.t. } V(x, y) \text{ accepts, then} \]

\[ \exists \text{ circuit } D_x \text{ of } |x|^c \text{ size and } |x| + d \text{ inputs s.t. } V(x, tt(D_x)) \text{ accepts,} \]

where $tt(D_x) = \text{Truth Table of circuit } D_x$.

**Easy Witness Lemma [IKW’02]:**

If NEXP is in P/poly then all NEXP problems have *easy witnesses*.

**Small circuits for solving NEXP problems**\n
$\Rightarrow$ **Small circuits for solutions to NEXP problems**

Replace 1 with: 1’. Guess poly($|x|$)-size circuit $D_x$
Proof Sketch of Theorem A

Idea. Assume

(1) NEXP has poly-size $C$-circuits, and
(2) there’s a faster Gap $C$-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 compute $L$ on $x$ by:

1. Guessing a circuit $D_x$ of $\text{poly}(|x|)$ size
   (Easy Witness Lemma, using (1))

2. Using (2) to check $D_x$ encodes a witness for $x$ in $o(2^n)$ time (Nice PCPs for $L$)
Improving Theorem A [MW’18]

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

**Thm A+ [MW18]** If there is an $\varepsilon > 0$ such that
\[
\text{Gap-$\mathcal{C}$-SAT on } 2^{n^\varepsilon} \text{-size circuits is in } O(2^{n-n^\varepsilon}) \text{ time}
\]
then $\text{NTIME}[2^{(\log n)^{O(1)}}]$ doesn’t have poly-size $\mathcal{C}$-circuits

**Thm A++ [MW18]** If there is an $\varepsilon > 0$ such that
\[
\text{Gap-$\mathcal{C}$-SAT on } 2^{\varepsilon n} \text{-size circuits is in } O(2^{n - (1 - \varepsilon) n}) \text{ time}
\]
then for all $k$, NP doesn’t have $n^k$-size $\mathcal{C}$-circuits
and $\text{NTIME}[n^{\log^* n}]$ doesn’t have poly-size $\mathcal{C}$-circuits [Tell’18]
Proof of Theorem A++?

Approach: Want to show that given

(1) **NP has** $n^k$-size $C$-circuits, and

(2) **Gap-$C$-SAT** algorithm running in $2^{(1-\varepsilon)n}$ time

Then $\text{NTIME}[n^d] \subseteq \text{NTIME}[o(n^d)]$ for some $d$

Let $L \in \text{NTIME}[n^d]$. To solve $L$ faster on input $x$,

1. **Guess a witness circuit** $C_x$ of $o(n^d)$ size

2. **Check** $C_x$ encodes witness for $x$ in $o(n^d)$ time
   (Use nice PCP; this still works, if part 1 works)

   Easy Witness Lemma only works for NEXP!
New Easy Witness Lemma [MW’18]

\[ \text{NTIME}[t(n)] \text{ has } s(n)\text{-size witness circuits if} \]
\[ \forall L \in \text{NTIME}[t(n)], \ \forall \text{Verifiers } V, \ \forall x \in L, \]
\[ \exists \ s(n)\text{-size circuit } D_x \text{ such that } V(x, \text{tt}(D_x)) \text{ accepts}. \]

Old Easy Witness Lemma [IKW02]:

If every problem in \( \text{NEXP} \) has \( \text{poly}(n) \)-size circuits, then \( \text{NEXP} \) has \( \text{poly}(n) \)-size witness circuits.

New Easy Witness Lemma (Special Case of [MW’18]):

If every problem in \( \text{NP} \) has \( n^k \)-size circuits, then \( \text{NP} \) has \( n^{O(k^3)} \)-size witness circuits.

Similar statement for \( \text{NTIME}[n^{\text{polylog } n}] \).
Proof of Theorem A++?

Approach: Want to show that given

1. **NP** has **n**\(^k\)-size **C**-circuits, and

2. **Gap-**\(\mathbb{C}\)-SAT algorithm for **2**\(^{\epsilon n}\) size, in **2**\(^{n(1-\epsilon)}\) time

Then **NTIME**[\(n^{k^4}\)] \(\subseteq\) **NTIME**[\(o(n^{k^4})\)]

Let \(L \in **NTIME**[n^{k^4}]\). To solve \(L\) faster on input \(x\),

1. **Guess circuit** \(C_x\) of **O**\((n^{k^3})\) size with \(k^4 \log n\) inputs, encoding witness \(y\) of length \(n^{k^4}\)
   (Use (1) and New Easy Witness Lemma)

2. **Check** \(C_x\) encodes witness for \(x\) in **o**\((n^{k^4})\) time
   (Use (2) and nice PCP)

**Contradiction!**
IKW’s Easy Witness Lemma

**Easy Witness Lemma [IKW02]:**
\( \text{NTIME}[2^n] \subseteq \text{SIZE}[n^k] \) for some \( k \)
\[ \implies \] \( \text{NTIME}[2^n] \) has \( n^c \)-size witness circuits for some \( c \).

**Strategy:** Assume the negation, prove a contradiction!

(1) \( \exists k \) \( \text{NTIME}[2^n] \subseteq \text{SIZE}[n^k] \) and

(2) \( \forall c, \text{NTIME}[2^n] \) DOESN’T have \( n^c \)-size witness circuits

IKW start with \( L_{\text{hard}} \in \text{SPACE}[n^{k+1}] / \text{i.o.-SIZE}[n^k] \)

and show how assumptions (1) and (2) imply:

\( \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \)

Merlin-Arthur protocols

infinitely often, with \( n \) bits of advice
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \text{ NTIME}[2^n] \subseteq \text{SIZE}[^n \leq k] \) and
(2) \( \forall c, \text{NTIME}[2^n] \text{ DOESN’T have } n^c \text{-size witness circuits} \)

Show how assumptions (1) and (2) imply:
\( \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \)

**MA:** Merlin-Arthur = NP with probabilistic verification

L is in MA means there’s a polytime V such that
\( x \in L \) \( \Rightarrow \) there is a y such that \( V(x,y) \) always accepts
\( x \notin L \) \( \Rightarrow \) for every y, \( V(x,y) \) rejects with prob > \( \frac{3}{4} \)

Merlin  Arthur
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \ \text{NTIME}[2^n] \subset \text{SIZE}[n^k] \) and
(2) \( \forall c, \ \text{NTIME}[2^n] \text{ DOESN’T have } n^c \text{-size witness circuits} \)

Show how assumptions (1) and (2) imply:

\[
\text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k]
\]

(1) \( \text{NTIME}[2^n] \subset \text{SIZE}[n^k] \)

\( \Rightarrow \) \( \text{SPACE}[O(n)] \subset \text{P/poly} \)

\( \Rightarrow \) \( \text{PSPACE} \subset \text{P/poly} \)

\( \Rightarrow \) \text{PSPACE} = \text{MA} \quad \text{[BFNW’93]}

Use the fact that \( \text{PSPACE} = \text{IP} \) \text{[Shamir]}:

Guess a small circuit encoding the prover’s strategy, then run the interactive protocol with that circuit
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \) NTIME\([2^n]\) \(\subseteq\) SIZE\([n^k]\) \(\text{ and}\)

(2) \( \forall c, \) NTIME\([2^n]\) DOESN’T have \(n^c\)-size witness circuits

Show how assumptions (1) and (2) imply:

\[ \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \]

(1) NTIME\([2^n]\) \(\subseteq\) SIZE\([n^k]\)

\(\Rightarrow\) i.o.-NTIME\([2^n]/n \subseteq\) i.o.-SIZE\([n^k]\)

(Hard-code the advice in the circuit)
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \text{ NTIME}[2^n] \subseteq \text{SIZE}[n^k] \) and

(2) \( \forall c, \text{NTIME}[2^n] \text{ DOESN’T have } n^c \text{-size witness circuits} \)

Show how assumptions (1) and (2) imply:

\( \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \)

(2) NTIME[2^n] DOESN’T have \( n^c \)-size witness circuits:

\( \neg \left( \forall L \in \text{NTIME}[2^n], \forall \text{Verifiers } V, \text{ for all but finitely many } x \in L, \right. \)
\( \left. \exists y \text{ s.t. } V(x, y) \text{ accepts and (Circuit complexity of } y) \leq n^c \right) \)
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \ \text{NTIME}[2^n] \subseteq \text{SIZE}[n^k] \) and
(2) \( \forall c, \text{NTIME}[2^n] \) DOESN’T have \( n^c \)-size witness circuits

Show how assumptions (1) and (2) imply:
\( \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \)

(2) \( \text{NTIME}[2^n] \) DOESN’T have \( n^c \)-size witness circuits:
\( \exists L \in \text{NTIME}[2^n], \exists \text{Verifier V}, \exists \text{infinitely many } x \in L, \)
\( \text{such that } \forall y [ V(x, y) \text{ accepts } \Rightarrow (\text{Circuit complexity of } y) > n^c ] \)

*Given a ‘bad’ input \( x \) as advice, can use verifier V to guess-and-check a function with circuit complexity > \( n^c \) in \( O(2^n) \) time*

*Can nondeterministically generate hard functions!*
Proof of IKW’s Easy Witness Lemma

(1) \( \exists k \) NTIME\([2^n] \subseteq \text{SIZE}[n^k] \) and
(2) \( \forall c \), NTIME\([2^n] \) DOESN’T have \( n^c \)-size witness circuits

Show how assumptions (1) and (2) imply:

\[ \text{SPACE}[n^{k+1}] \subseteq \text{MA} \subseteq \text{i.o.-NTIME}[2^n]/n \subseteq \text{i.o.-SIZE}[n^k] \]

(2) NTIME\([2^n] \) DOESN’T have \( n^c \)-size witness circuits:
\( \exists L \in \text{NTIME}[2^n], \exists \text{Verifier } V, \exists \text{ infinitely many } x \in L, \)

such that \( \forall y \ [ V(x, y) \text{ accepts } \Rightarrow (\text{Circuit complexity of } y) > n^c ] \)

**Thm [Hardness-to-PRGs]** There’s an \( \alpha > 0 \) and \( O(2^n) \)-time computable \( F \) such that, given a string \( y \) with circuit complexity \( > n^c \),

\( F \) outputs a set of \( O(2^n) \) strings which “fool” all circuits of size \( n^{\alpha c} \)

Use \( F \) to derandomize \( n^{O(c)} \)-time Merlin-Arthur protocols in \( O(2^n) \) time, on infinitely many input lengths, with \( n \) bits of advice
Scaling Down to NP?

**New Easy Witness Lemma (Special Case)**
If $\mathsf{NP}$ has $n^k$-size circuits,
then $\mathsf{NP}$ has $n^{O(k^3)}$-size witness circuits.

**Idea:** Derive a contradiction from assuming that

$$\mathsf{NP} \subset \mathsf{SIZE}[n^k]$$
and

$$\forall c, \mathsf{NP} \text{ does NOT have } n^c \text{-size witness circuits.}$$
Scaling Down to NP?

What happens when we try to follow the IKW proof?
We want to derive something like:

\[ \text{PSPACE} \subseteq \text{MA} \subseteq \text{i.o.NP}_/n \subseteq \text{i.o.SIZE}[n^k] \]

These two inclusions are OK!

They follow from \( \text{NP} \subseteq \text{SIZE}[n^k] \)
and

NP does NOT have \( n^c \)-size witness circuits
Scaling Down to NP?

What happens when we try to follow the IKW proof?
We want to derive something like:

\[ \text{PSPACE} \subseteq \text{MA} \subseteq \text{i.o.} \text{NP}_n \subseteq \text{i.o.} \text{SIZE}[n^k] \]

**Problem:** Can’t conclude PSPACE is in MA from assuming NP \( \subseteq \text{SIZE}[n^k] \) and NP does NOT have \( n^c \)-size witness circuits!

**Possible fix:** Use another circuit lower bound?

\[ \text{Thm [San07]} \; \text{MA}_{/1} \not\subseteq \text{SIZE}[n^k] \]
Scaling Down to NP?

What happens when we try to follow the IKW proof? We want to derive something like:

\[ \text{MA}_1 \subseteq \text{i.o.NP}_{/n+1} \subseteq \text{i.o.SIZE}[n^k] \]

**New problem:** We only know \( \text{MA}_1 \not
subseteq \text{SIZE}[n^k] \)

**Don’t know if** \( \text{MA}_1 \not
subseteq \text{i.o.SIZE}[n^k] \)

**Possible fix:** Prove a stronger MA lower bound? Turns out we don’t need an “almost-everywhere” lower bound...
New Lower Bound for Merlin-Arthur Protocols

**Thm [MW’18]** For all $k$, there is an $L \in \text{MA-TIME}[n^{k^2}]/O(\log n)$ such that for all but finitely many input lengths $n$,

- either $L_n$ has circuit complexity at least $n^k$
- or $L_{n^k}$ has circuit complexity at least $n^{k^2}$

Our proof of the new EWL shows:

If every problem in NP has $n^k$-size circuits and some NP problem doesn’t have $n^{O(k^3)}$-size witnesses, then the above Merlin-Arthur lower bound is contradicted!
Sketch of the New Easy Witness Lemma

Start with \( L \in \text{MA-TIME}[n^{k^2}]_{/O(\log n)} \) from our new circuit lower bound.

Assuming some NP problem doesn’t have \( n^{O(k^3)} \)-size witnesses, we derive a partial derandomization of the MA protocol for \( L \):

For infinitely many \( n \), there is an \( \text{NP}_{/O(n)} \) algorithm computing \( L \) correctly on all inputs of length \( n \) AND of length \( n^k \).

Assuming \( \text{NP has } n^k \)-size circuits, we can derive:

For infinitely many \( n \), \( L_n \) has an \( n^k \)-size circuit AND \( L_{n^k} \) has an \( n^{k^2} \)-size circuit.

This directly contradicts our lower bound for \( L \)!
More Details on Derandomizing MA

Assume: NP does NOT have $n^{k^3}$-size witness circuits. Let V be a “bad” verifier (for inf. many $x$, every witness for $x$ is not easy)

How to derive $\text{MA}_{/O(\log n)} \subseteq \text{i.o.} \text{NP}_{/n+O(\log n)}$

Given a ‘bad’ $x_w$ as advice,

Guess a ‘bad’ $y$ such that $V(x_w,y)$ accepts

// $y$ encodes a function with circuit complexity $> n^{k^3}$

Stick $y$ into a PRG that fools $n^{\Omega(k^3)}$-size circuits

Use PRG to derandomize an $m$-time MA protocol

(Guess Merlin’s message, construct a circuit of size $m^2$ that takes Arthur’s message as input)

This works as long as $m^2 << n^{O(k^3)}$
More Details on Derandomizing MA

**How to derive** $\text{MA} \subseteq \text{i.o. NP}$

Given a ‘bad’ $x_w$ as advice,

1. Guess a ‘bad’ $y$ such that $V(x_w, y)$ accepts
   
   // $y$ encodes a function with circuit complexity $> n^{k^3}$

2. Stick $y$ into a PRG that fools $n^{\Omega(k^3)}$-size circuits

3. Use PRG to derandomize an $m$-time MA protocol
   (Guess Merlin’s message, construct a circuit of size $m^2$ that takes Arthur’s message as input)

   This works as long as $m^2 << n^{O(k^3)}$

If NP does not have $n^{k^3}$-size witness circuits, the *same* advice $x_w$ can be used to derandomize MA

for *all* running times up to $m = n^{O(k^3)}$
Lower Bounds for NP Against Some Depth-Two Classes
Let $\mathcal{C}$ be a class of “simple” functions (take Boolean inputs, but need not be Boolean-valued).

Which “interesting” functions $f$ can (not) be represented by “short” $\mathbb{R}$-linear combinations of functions from $\mathcal{C}$?

$$f : \{0,1\}^n \to \{0,1\} \equiv \sum$$

If $\mathcal{C}$ spans the vector space of all functions $f : \{0,1\}^n \to \mathbb{R}$ then there is always some $\sum \circ \mathcal{C}$ circuit of $\leq 2^n$ size...
The $\mathbb{R}$-linear Representation Problem

Which “interesting” functions $f$ can (not) be represented by “short” $\mathbb{R}$-linear combinations of functions from $C$?

If $C$ is the class of $2^n$ AND functions on $n$ variables:
$$\sum \circ AND \equiv 0/1 \text{ polynomials over } \mathbb{R}$$

If $C$ is the class of $2^n$ PARITY functions on $n$ variables:
$$\sum \circ PARITY \equiv -1/1 \text{ polynomials over } \mathbb{R}$$

(Fourier analysis of Boolean functions)

These are well-understood:
$C$ is a basis for the vector space of functions $f : \{0,1\}^n \to \mathbb{R}$
$\implies$ the $\mathbb{R}$-linear representation of $f$ is unique,
so the “shortest” is also the “longest”…

More interesting cases: representations are not unique
[W’18] Three Simple Classes

1. Linear Threshold Functions \([LTF]\)
2. Rectified Linear Units \([ReLU]\)
3. \(GF(p)\)-Polynomials of Degree-\(d\) \([POLYd[p]]\)
   \(\text{(p prime and } d \geq 2)\)

For all three classes:

- There are \(\gg 2^n\) functions on \(n\) variables, so \(\mathbb{R}\)-linear representations are not unique.
- \(2^{\Theta(n^2)}\) LTFs, \(p^{\Theta(n^d)}\) degree-\(d\) polys, \(\infty\) ReLU functions

- \(\mathbb{R}\)-linear Representations have been studied!
  \[ \sum \circ LTF = \text{Special Case of Depth-2 Threshold Circuits} \]
  \[ \sum \circ ReLU = \text{“Depth-2 Neural Net with ReLU activation”} \]
  \[ \sum \circ POLYd[p] = \text{“Higher-Order” Fourier Analysis for } d \geq 2 \]
Sums of Linear Threshold Functions

**Def.** $f_n : \{0,1\}^n \rightarrow \{0,1\}$ is an LTF if $\exists w_1, \ldots w_n, t \in \mathbb{R}$ such that
\[
\forall (x_1, \ldots, x_n) \in \{0,1\}^n, \quad f(x_1, \ldots, x_n) = \textup{1} \iff \sum_i w_i x_i \geq t
\]

Depth-Two LTF Circuits ($\textup{LTF} \circ \textup{LTF}$): Major problem to find “nice” functions without $n^k$-gate $\textup{LTF} \circ \textup{LTF}$ circuits, for all $k$

[Hajnal et al.’91] $\exp(n)$ depth-two lower bounds for small $w_i$’s

[Roychowdhury-Orlitsky-Siu’94] What about $\sum \circ \textup{LTF}$?

**Special case of $\textup{LTF} \circ \textup{LTF}$:**
the linear form for output LTF must always evaluate to 0 or 1

Still, no $n^{1.5}$-gate lower bounds were known for $\sum \circ \textup{LTF}$!

We prove:

\[\text{Thm} \forall k, \exists f_k \in \text{NP} \text{ without } n^k\text{-size } \sum \circ \text{LTF}\]

\[\text{Thm} \exists f \in \text{NTIME}[n^{\log^* n}] \text{ without } \text{poly}(n)\text{-size } \sum \circ \text{LTF}\]

Note: It is a major open problem to prove
\[\exists f \in \text{NP} \text{ without } n^k\text{-size (unrestricted) circuits}\]
Sums of ReLUs

Def. \( f_n : \mathbb{R}^n \to \mathbb{R}^+ \) is a ReLU if \( \exists w_1, \ldots, w_n, t \in \mathbb{R} \) such that 
\[ \forall (x_1, \ldots, x_n) \in \mathbb{R}^n, \quad f(x_1, \ldots, x_n) = \max(0, \sum_i w_i x_i + t) \]

\[ \Sigma \circ \text{ReLU} \] generalizes \[ \Sigma \circ \text{LTF} \]

\[ \Sigma \circ \text{ReLU} \] = “Depth-Two Neural Nets with ReLU Activations”

Very widely studied, thousands of references

Several recent references [see paper] give lower bounds for some “weird” \( f : \mathbb{R}^n \to \mathbb{R} \) which vary sharply / sensitive

No lower bounds known for discrete-domain / Boolean functions (note: “most sensitive” Boolean fn PARITY has \( O(n) \)-size \( \Sigma \circ \text{LTF} \))

We can generalize the \( \Sigma \circ \text{LTF} \) limits to \( \Sigma \circ \text{ReLU} \):

Thm \( \forall k, \exists f_k \in \text{NP} \) without \( n^k \)-size \( \Sigma \circ \text{ReLU} \)

Thm \( \exists f \in \text{NTIME}[n^{\log^* n}] \) without \( \text{poly}(n) \)-size \( \Sigma \circ \text{ReLU} \)
Compelling Conjecture ["Degree-Two Uncertainty Principle"]: linear combination of \( f : \{0,1\}^n \rightarrow \{0,1,\ldots,p-1\} \) where for every \( f \) there is a degree-\( d \) polynomial \( q(x) \) such that 
\[
\forall x \in \{0,1\}^n, f(x) = q(x) \mod p
\]
Case of \( d = 2, p = 2 \) is already very interesting!

Compelling Conjecture ["Degree-Two Uncertainty Principle"]: 
\( \text{AND} \) (on \( n \) inputs) requires \( n^{\omega(1)} \)-size \( \sum \circ \text{POLY2}[2] \)

\textbf{Known}: \( \text{AND} \) requires \( \Omega(2^n) \)-size \( \sum \circ \text{POLY1}[2] \)

\( \text{AND} \) has \( O(2^{n/2}) \)-size \( \sum \circ \text{POLY2}[2] \)

No non-trivial lower bounds were known for \( \sum \circ \text{POLY2}[p] \)

We prove:

\textbf{Thm} \( \forall d, k, \forall p \text{ prime}, \exists f_k \in \text{NP} \) without \( n^k \)-size \( \sum \circ \text{POLYd}[p] \)

\textbf{Thm} \( \exists f \in \text{NTIME}[n^{\log^*n}] \) without \( \text{poly}(n) \)-size \( \sum \circ \text{POLYd}[p] \) for all fixed \( d \) and fixed prime \( p \)
Key Theorem

A new instance of “Circuit Analysis Algorithms ⇒ Circuit Lower Bounds”

Key Theorem: Let \( \mathcal{C} \) be a class of functions \( f : \{0, 1\}^n \to \mathbb{R} \).
Assume: there is an \( \epsilon > 0 \) and an algorithm \( A \) so that
for any given \( f_1, \ldots, f_4 \in \mathcal{C} \), \( A \) can compute the “sum-product”

\[
\sum_{a \in \{0, 1\}^n} \prod_{i=1}^4 f_i(a)
\]

in \( 2^{n(1-\epsilon)} \) time.

Then: \( \forall k, \exists f \in \mathbf{NP} \) without \( n^k \)-size \( \sum^\circ \mathcal{C} \), and
\( \exists f \in \mathbf{NTIME}[n^{\log^* n}] \) without \( \text{poly}(n) \)-size \( \sum^\circ \mathcal{C} \)

Applies new Easy Witness Lemma [Murray-W’18]

We show how to compute sum-products in \( 2^{n(1-\epsilon)} \) time
for LTFs, ReLUs, and low-degree polynomials.
Major Ideas in the Key Theorem

Assume: (1) There is a $2^{n(1-\varepsilon)}$-time sum-product algorithm $A$ for $C$
(2) For some fixed $k$, all $f \in NP$ have $n^k$-size $\sum \circ C$

Goal: Derive a contradiction.

(1) and (2) $\Rightarrow$ Given (unrestricted) Boolean circuit $T$ with $n$ inputs and $m$ size,
we can guess-and-check an $m^k$-size $\sum \circ C$ computing $T$, in $2^{n(1-\varepsilon)}m^O(1)$ time

Notes: (a) Checking that a given $\sum \circ C$ is Boolean-valued is the hardest part.
(b) In order to guess the $\sum \circ C$ circuit, we need that the coefficients in our
linear combinations have “small” bit complexity, WLOG

(1) $\Rightarrow$ Can solve #Circuit-SAT in nondeterministic $2^{n(1-\varepsilon)}m^O(1)$ time

Idea: given (unrestricted) circuit $T$, guess-and-check an equivalent $m^k$-size
$\sum \circ C$ computing $T$. Then, $\#SAT(T)$ is equiv. to $\sum_{a \in \{0,1\}^n} (\sum \circ C (a)) = \sum \sum_a C(a)$.

[Murray-W’18] $+$ #Circuit-SAT algorithm $\Rightarrow \forall k, \exists f \in NP$ without $n^k$-size unrestricted circuits

Contradicts (2) when $\sum \circ C$ can be simulated by Boolean circuits!

The proof crucially relies on the $\sum \circ C$ circuit computing an arbitrary circuit exactly
Sum-Product Algorithm for LTF

Uses (old) fact that #Subset-Sum is solvable in $\text{poly}(n) \cdot 2^{n/2}$ time!

Thm [HS’76] #Subset-Sum on $n$ numbers is in $\text{poly}(n) \cdot 2^{n/2}$ time

Proof Given $w_1, \ldots, w_n, t$, we want to know the number of $S \subseteq [n]$ such that $\sum_{i \in S} w_i = t$

1. Enumerate all possible $2^{n/2}$ subsets $S$ of $\{w_1, \ldots, w_{n/2}\}$.
   Make a list $L_1$ of the $2^{n/2}$ subset sums, and SORT all sums in $L_1$

2. Enumerate all possible $2^{n/2}$ subsets $T$ of $\{w_{n/2+1}, \ldots, w_n\}$.
   For each $T$ summing to a value $v$,
   BINARY SEARCH for a value $v'$ in $L_1$ such that $v + v' = t$

3. To compute the total number of subsets summing to $t$:
   For each sum value $v'$ appearing in $L_1$,
   store the number $n_{v'}$ of subsets in $L_1$ which have value $v'$.
   Later, if value $v'$ is found in the binary search,
   add $n_{v'}$ to a running sum.

Takes $\text{poly}(n) \cdot 2^{n/2}$ time in total
Sum-Product Algorithm for LTF

Uses (old) fact that 
#Subset-Sum is solvable in \( poly(n) \cdot 2^{n/2} \) time!

**Thm** For any \( f_1, \ldots, f_4 \in LTF \), we can compute

\[
\sum_{a \in \{0,1\}^n} \prod_{i=1}^{4} f_i(a) \quad \text{in } poly(n) \cdot 2^{n/2} \text{ time.}
\]

**Proof** An Exact LTF (ELTF) \( g \) has the form \( g(x) = 1 \iff \sum_i w_i x_i = t \)

#Subset-Sum in \( poly(n) \cdot 2^{n/2} \) time \( \Rightarrow \sum_a g(a) \) in \( poly(n) \cdot 2^{n/2} \) time

[HP’10]: Every LTF on \( n \) inputs can be written as \( \sum_{poly(n)} \) ELTF

So we can write

\[
\sum_{a \in \{0,1\}^n} \prod_{i=1}^{4} f_i(a) = \sum_{a \in \{0,1\}^n} \prod_{i=1}^{4} \left( \sum_{poly(n)} g_{i,j}(a) \right) \text{ for ELTFs } g_{i,j}
\]

Simple algebra:

\[
= \sum_{a \in \{0,1\}^n} \sum_{poly(n)} \prod_{i=1}^{4} g_{i,j}(a) = \sum_{poly(n)} \sum_{a \in \{0,1\}^n} \prod_{i=1}^{4} g_{i,j}(a)
\]

Each \( \prod_{i=1}^{4} g_{i,j}(x) = h(x) \) for some ELTF \( h \)

Can compute in \( poly(n) \cdot 2^{n/2} \) time!
Open Problems

Know: For each $k$, there is an $f \in \text{NTIME} \left[ n^{O(k^4)} \right]$ without $n^k$-size $\Sigma \circ \text{LTF}$

Show $\text{SAT}$ requires $n^k$-size $\Sigma \circ \text{LTF}$, for all $k$

Show Quasi-NP does not have THR $\circ$ THR circuits of subquadratic size

Show there’s a function in $E^{NP}$ without $6n$ size circuits

I know how to solve \#SAT for $\Sigma \circ \text{POLY2}[2]$ in poly-time. Thus this class should not even represent CNF. Prove that!

If $\text{SAT} \in P$, then $\text{TIME}(n^{\log n})$ is not in $P/poly$.
If $\text{SAT}$ is in $n^{\text{polylog} n}$ time, then Quasi-P is not in $P/poly$.
Is such a connection true for Gap-Circuit-SAT?

[IW97] $\text{TIME}[2^{O(n)}]$ not in $2^{n/100}$ size) $\Rightarrow$ Gap-Circuit-SAT is in P
Thank you!