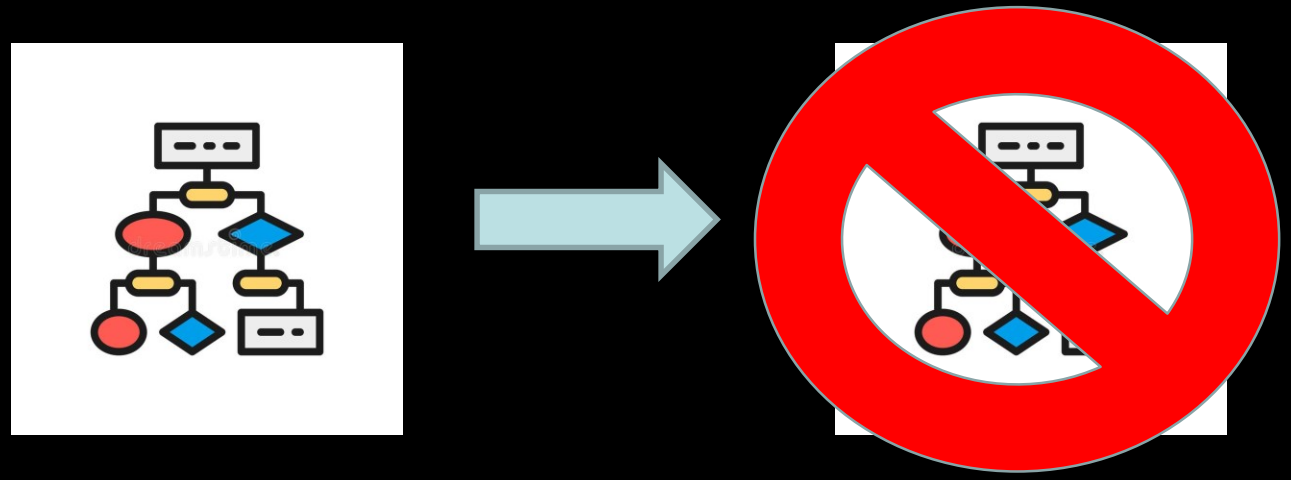
# How to Prove Lower Bounds With Algorithms



## Lecture 1: Introduction

# A view of algorithms and complexity, from 30,000 ft (9144 m)

- 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 needed to solve a problem*?)

# The tasks of the algorithm designer and the complexity theorist appear to be inherently opposite ones.

- Algorithm designers
- Complexity theorists

**Furthermore, it has been generally believed that algorithm design is somewhat "easier" than lower bounds**

- In algorithm design: you only have to find a single clever algorithm that solves a problem well
- In lower bounds: you must reason about "all possible" algorithms, and argue that 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 Proofs [90's]**
- **Algebrization [00's]**

**Summary:** The standard methods that we use to reason about generic computation cannot resolve **P** ≠ **NP** (or **P** ≠ **PSPACE**, or **EXP** ≠ **ZPP**, or **NEXP** ≠ **BPP**, etc.)

**Great pessimism in complexity theory**

# The Relativization Barrier, in One Slide

Let $O : \{0,1\}^\star \to \{0,1\}$ be arbitrary.

An algorithm $A^O$ with oracle $O$ gets to call $O$ as a sub-routine, and it takes one step.

**You can slap an oracle on practically anything in complexity theory…**

$TIME[t(n)]$={decision problems solvable in time $t(n)$}

$TIME^O[t(n)]$={decision problems solvable in time $t(n)$ with oracle $O$}

$\boldsymbol{P}$ = {decision problems solvable by some poly-time Turing machine}

$\boldsymbol{P^O}$ = {decision problems solvable in poly-time with oracle $O$}

Boolean circuits with $O$-oracle gates:

have AND, OR, NOT, and gates computing $O$ on fixed input lengths

1. Most theorems in complexity theory "relativize": still hold when oracles are added [this is very powerful! you get "uncountably many" corollaries for free!]
2. But results such as $P = NP$ (or $P \neq NP$, or $P \neq PSPACE$, or …) ***cannot*** relativize: example: there are oracles $A, B$ such that $P^A = NP^A$ and $P^B \neq NP^B$

# *How will we make progress?*

**There are *many* known "no-go" theorems**

- **Relativization [70's]**
- **Natural Proofs [90's]**
- **Algebrization [00's]**

**Summary:** The standard methods that we use to reason about generic computation cannot resolve **P** ≠ **NP** (or **P** ≠ **PSPACE**, or **EXP** ≠ **ZPP**, or **NEXP** ≠ **BPP**, etc.)

**Great pessimism in complexity theory**

# One Direction for Progress:
# *Find cases where Algorithm Design can imply Lower Bounds*

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

**Designing Algorithms ≈ 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

# Simple Example: The Time Hierarchy Theorem

**Theorem:** For "reasonable" f, g where **g(n) >> f(n)**,

$$\mathbf{TIME(f(n)) \subsetneq TIME(g(n))}$$

An ***algorithm*** determines how small g(n) can be

**Proof Sketch:** Define an algorithm **N** as follows.

**N** on input $\langle M \rangle$: "Let $n = |\langle M \rangle|$. Simulate **M** on $\langle M \rangle$ for up to **f($n$)** steps. If the sim halts, output the opposite answer."

**Claim: The function computed by N cannot be in time f($n$).**

**Proof:** Assume some **D** runs in **f($n$) time**, where D is equivalent to N. By assumption, **D** on $\langle D \rangle$ runs in **f($n$) time** and outputs the *opposite* answer of **D** on $\langle D \rangle$ after **f($n$)** steps! **Contradiction!**

**Universal simulator ➔ N can be implemented in g(n) time**

# Another Simple Example

**If PSPACE = EXPTIME then PTIME ≠ PSPACE**


PSPACE = problems solvable in polynomial space

PTIME = …. in polynomial time

EXPTIME = … in exponential time

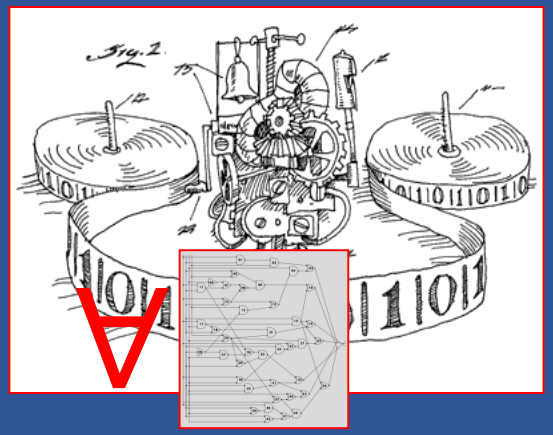
Proof: PTIME ≠ EXPTIME (time hierarchy theorem)
So PTIME = PSPACE implies PSPACE ≠ EXPTIME. QED

**Many such results can be proved….**
**But they do not seem very useful!**

# Big Idea: Interesting circuit-analysis algorithms tell us about the *limitations* of circuits in modeling algorithms

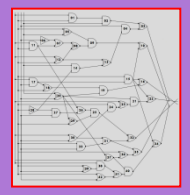
SAT? *YES/NO*

∃

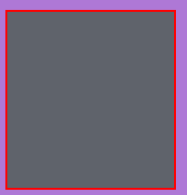


∀

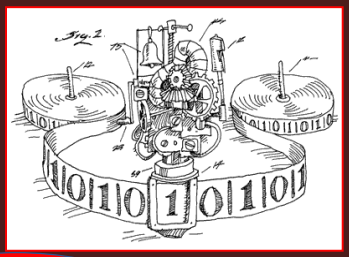
"Non-Trivial" Circuit Analysis Algorithm



≠



∃ "interesting" $f$



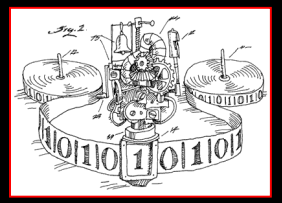
∀

Circuit Lower Bounds

Circuits are **not** "black-boxes" to algorithms!

# Circuit Complexity: A Crash Course

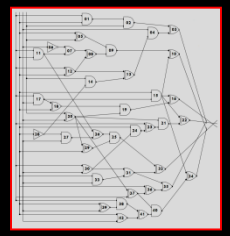
# Algorithms



Can take in **arbitrarily long inputs** and still solve the (decision) problem

$$f : \{0,1\}^* \to \{0,1\}$$

# Circuits

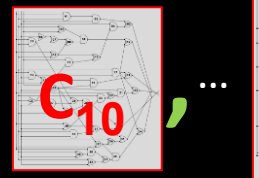
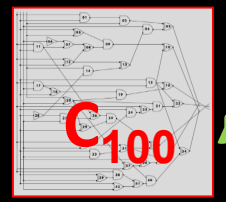
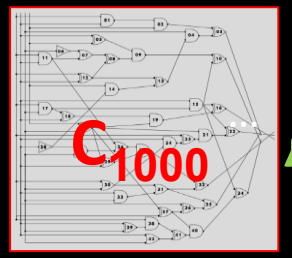


Can only take in inputs of a fixed length

$$g: \{0,1\}^n \to \{0,1\}$$

To compute functions of the form $f : \{0,1\}^* \to \{0,1\}$ with circuits, we define a

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

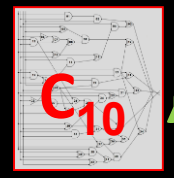
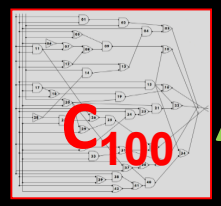
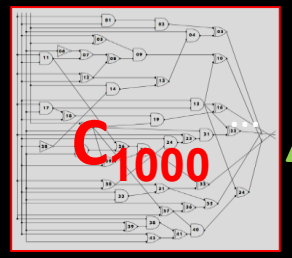





For each $n$, have a circuit $C_n$ to be run on all inputs of length $n$

A circuit family can be viewed as a "program with an ***infinite-length description***"

**P/poly** = { $f : \{0,1\}^* \to \{0,1\}$ computable by a **circuit family** $\{C_n\}$
where $\exists\ c$ s.t. for every $n$, the ***size* of $C_n$ is at most $cn^c$** }

***Each circuit is "small" relative to its number of inputs***

# Circuit Family = {   , …   , …   , …   , … }

**P/poly** = { $f : \{0,1\}^* \to \{0,1\}$ computable by a **circuit family** $\{C_n\}$ where for every $n$, the ***size* of $C_n$ is at most poly($n$)** }

**Conjecture: NP $\not\subset$ P/poly**

**Why study this model?**

One motivation: Proving limitations on circuit families is a step towards ***non-asymptotic complexity theory:***

**Concrete limitations on computing within the known universe**
**"Any computer solving most instances of this 1000-bit problem needs at least $10^{80}$ bits to be described"**

**Universe stores < $10^{80}$ bits** [Bekenstein '70s]

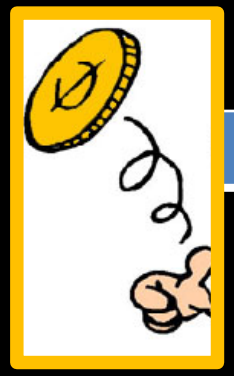
[Meyer-Stockmeyer '70s]
**Proved such a result**
**(for an EXPSPACE problem)**

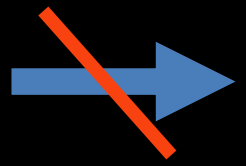
# Which Functions Have High Circuit Complexity?

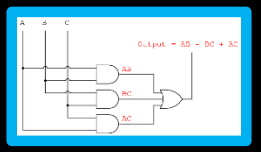
**Nearly *all* of them… "Most" functions require *huge* circuits!**

**Theorem [Shannon '49, Lupanov '58]**

With high probability,
a randomly chosen function $f : \{0,1\}^n \to \{0,1\}$
does not have circuits of size less than $\sim 2^n/n$
(and: every $f$ has a circuit of size about $\sim 2^n/n$)





***Which "natural" functions exhibit this exponential behavior?***

# Circuit Lower Bounds ⇒ Derandomization

**Thm [Nisan-Wigderson, Impagliazzo-Wigderson 90s]**

If there is an $f : \{0,1\}^\star \to \{0,1\}$
computable in $\mathbf{2^{O(n)}}$ **time**
that does not have circuits of size at most $\mathbf{2^{\varepsilon n}}$
(for all but finitely many n)

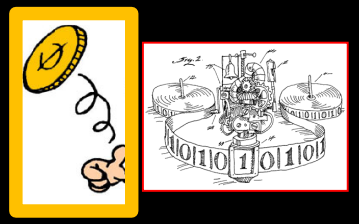
This is widely believed!

**Then Randomized Time ≡ Deterministic Time**
**(many other results in this direction)**

Another motivation to prove circuit lower bounds!

***Idea: If f "looks random" to all circuits, then f can be used to construct a pseudorandom generator, replacing true randomness in efficient computation!***

# Algorithms vs Circuit Families



T(n) *time* on inputs of length n

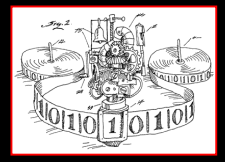
BPP is in P/poly

{ $C_1$ , $C_2$ , $C_3$ ,…}

$C_n$ has ≈ $poly$(T(n)) *size*

{ $C_1$ , $C_2$ , $C_3$ ,…}

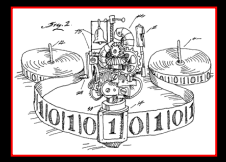
There is a family where every $C_n$ has ≈ *size n*



*No algorithm whatsoever!*

Some undecidable problems are in P/poly

Conjecture: NP ⊄ P/poly



**EXPONENTIAL TIME**

($2^n$ *steps*)

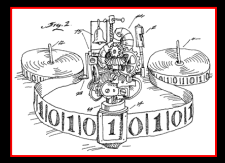
????

{ $C_1$ , $C_2$ , $C_3$ ,…}

every $C_n$ has ≈ $n^2$ *size !!*

EXP in P/poly is open!
(even NEXP in P/poly is open!!)

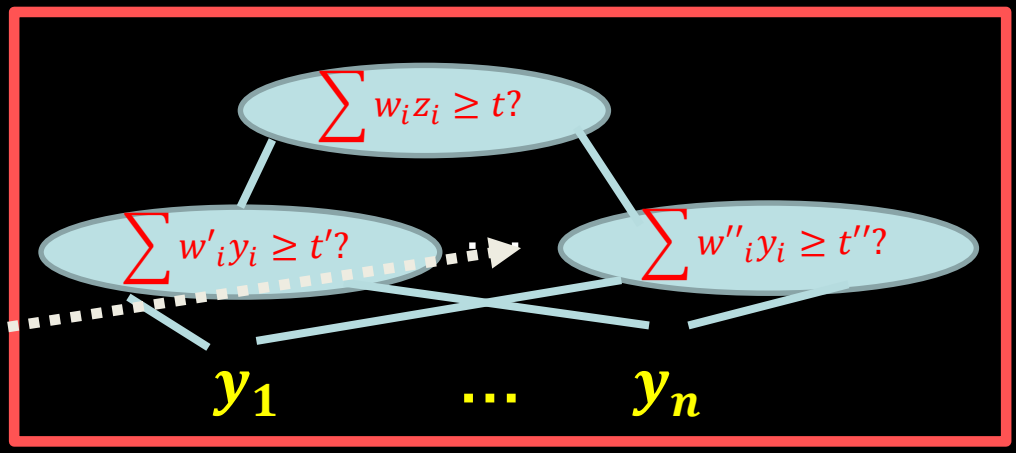
# Exponential Time Versus "Shallow" Nets?



**NONDETERMISTIC EXPONENTIAL TIME (solutions of $2^n$ *size, verified in* $2^n$ *time!*)**

???? { $C_1$ , $C_2$ , …, $C_n$ , … }

zoom in




$n^3$ "neurons"

**THIS IS OPEN!!**

We don't yet understand very simple neural networks!

*"Neural nets with one hidden layer"*
*-- this should be a very weak class!*

Depth-two threshold circuits

**Here endeth the Crash Course…**

**Now, what kinds of circuit analysis problems will we consider?**

# 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 $\mathbf{O(2^n\ |K|)}$ time by brute force

**QUESTION:** For what $\mathcal{C}$ is there a ***faster*** algorithm?

Example: for 3CNFs there is a long line of work…

Best known is about $O(1.31^n)$ *time*

# 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?]** If **Gap-Circuit-SAT ∈ P** then **P = RP**

**[Hirsch, Trevisan, …]** **Gap-kSAT ∈ P, for all k**

**Best known algorithm for Gap-Circuit-SAT:** $O(2^n |K|)$ time

**QUESTION:** For what $\mathcal{C}$ is there a ***faster*** algorithm?

# "Perfect" Circuit Analysis ➔ Circuit Lower Bounds

[Karp-Lipton-Meyer '80]
Suppose we had *extremely efficient* circuit-analysis algorithms. Then there are problems with exponential-time algorithms *that require maximum circuit complexity*

P = NP
(Circuit SAT in P)

EXP needs circuits of $\Omega(2^n/n)$ size


This is an interesting implication…
But we do not believe that the hypothesis is true,
and we believe that the consequence is true!

*Aside: Could you use this to separate P from NP??*

# Faster Algorithms ⟹ Lower Bounds

*C needs some closure properties*

| Slightly Faster Circuit-SAT [R.W. '10,'11] | No "Circuits for NEXP" |
|---|---|
| ***Deterministic* algorithms for:** | **Would imply:** |
| • **Circuit SAT** in $O(2^n/n^{10})$ time with n inputs and $n^k$ gates | • **NEXP $\not\subset$ P/poly** |
| • **Formula SAT** in $O(2^n/n^{10})$ | • **NEXP $\not\subset$ Poly-size Formulas** |
| • $\mathcal{C}$**-SAT** in $O(2^n/n^{10})$ | • **NEXP $\not\subset$ *poly-size* $\mathcal{C}$** |
| • Gap-$\mathcal{C}$-SAT is in **$O(2^n/n^{10})$ time** on $n^k$ size | **NEXP $\not\subset$ poly-size $\mathcal{C}$** |
| *(Easily solved w/ randomness!)* | |

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

# Even Faster ⟹ "Easier" Functions

**Better "Algorithms for Circuits"**

**[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 $O(2^{n-n^{\epsilon}})$
- $\boldsymbol{C}$**-SAT** in $O(2^{n-n^{\epsilon}})$
- Gap-$\boldsymbol{C}$-SAT is in $\mathbf{O(2^{n-n^{\epsilon}})}$ **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**$[n^{polylog\, n}] \not\subset$ **psize** $\boldsymbol{C}$

**NTIME**$[n^{polylog\, n}] \not\subset$ **psize** $\boldsymbol{C}$

# Even Faster ⟹ "Easier" Functions

| | Fine-Grained SAT Algorithms [Murray-W. '18] | No "Circuits for NP" |
|---|---|---|
| | **Det. algorithm for some $\epsilon > 0$:** | **Would imply:** |
| Note: Not currently believed… | • **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 $k$** |
| | • **Formula SAT** in $O(2^{(1-\epsilon)n})$ | • **NP $\not\subset$ Formula-SIZE($n^k$)** |
| | • **$C$-SAT** in $O(2^{(1-\epsilon)n})$ | • **NP $\not\subset$ $C$-SIZE($n^k$) for all $k$** |
| Strongly believed to be true… | • 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 ⟹ Average-Case Lower Bounds

(and for larger circuit classes!)

[R.Chen-Oliveira-Santhanam'18, Chen-W'19, Chen'19, Chen-Ren '20, Chen-Lyu-Williams '20, Chen'23]

**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

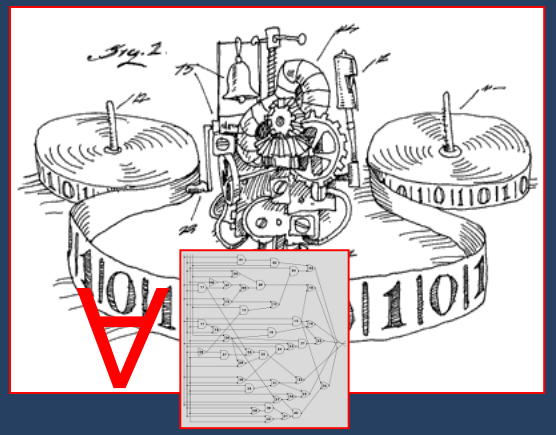
There is an $f \in$ **NTIME[$n^{polylog\, n}$]** such that, for **infinitely many** $n$, every poly($n$)-size

There's $f \in$ **NTIME[$n^{polylog\, n}$]** such that, for **infinitely many** $n$, every poly($n$)-size circuit in "MAJ of SUM of $C$" fails to compute $f_n$ [Chen-Lu-Lyu-Oliveira'22]

# Why on Earth would it be true?

SAT? YES/NO

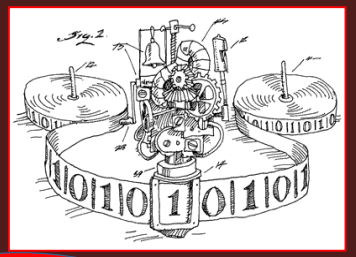
∃



∀

**“Non-Trivial” Circuit Analysis Algorithm**

∃ ”interesting” $f$



∀

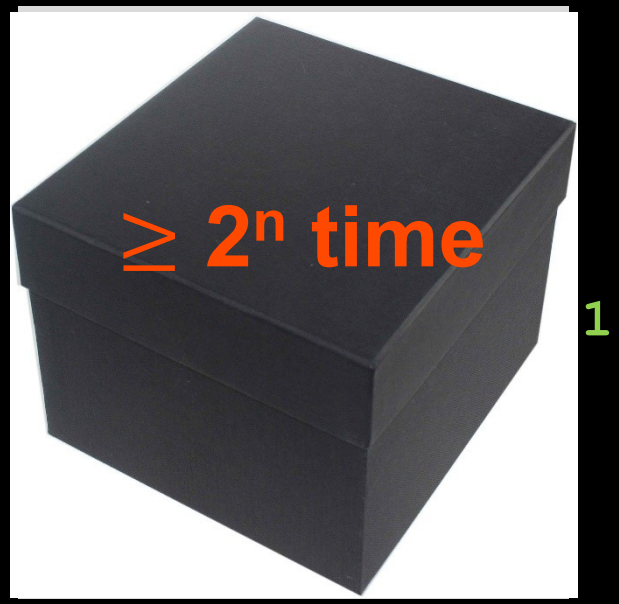
**Circuit Lower Bounds**

# *Some More Intuition*

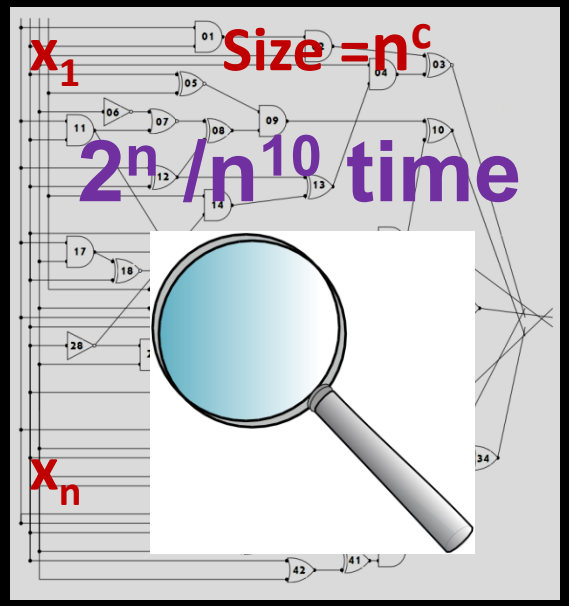
Faster Circuit-SAT algorithms reveal a ***weakness*** of small circuits

***Small circuits cannot "obfuscate" the all-zeroes function as well as a black-box can!***

**Black-Box SAT:**




**Circuit SAT:**



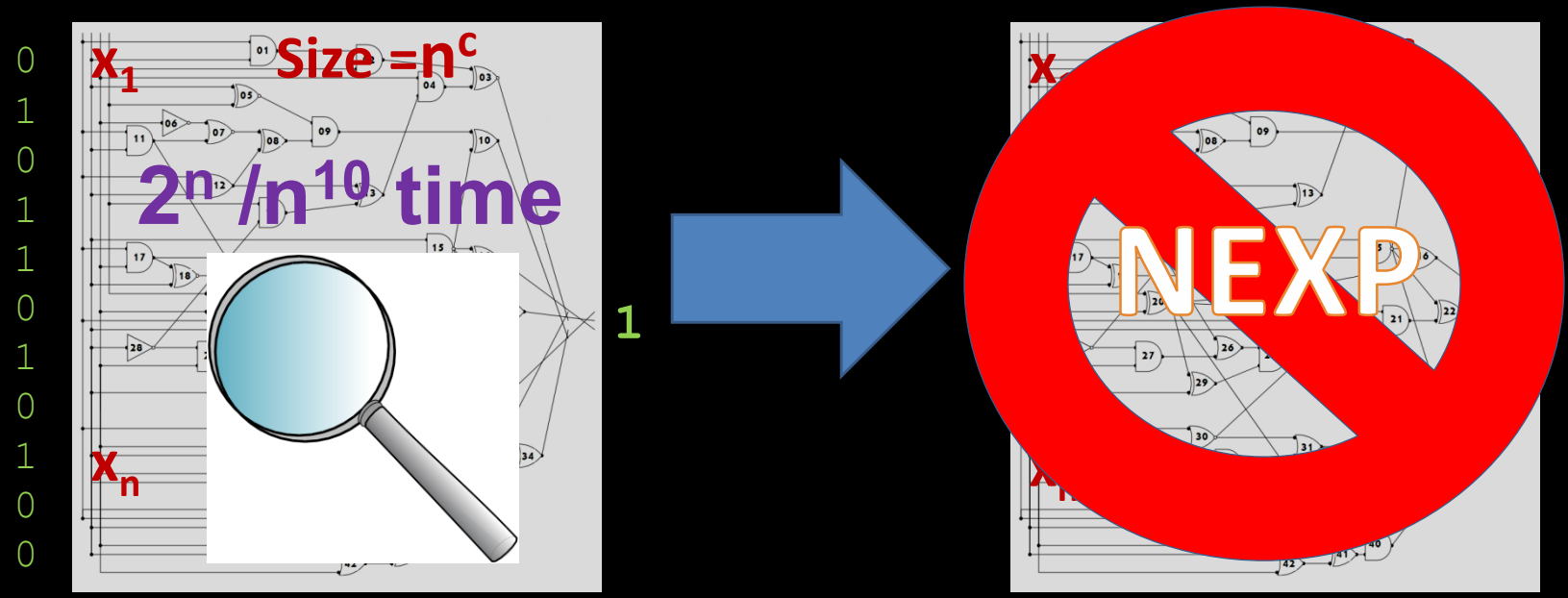

**<u>Proposition:</u> For every algorithm *A* computing SAT on black-boxes, there is a box *B* such that *A* must call *B* for Ω(2^n) times!**

***Therefore: a faster Circuit SAT algorithm demonstrates a concrete difference between a "white-box" circuit problem and a "black-box" problem***

# *Some More Intuition*

Faster Circuit-SAT algorithms show a ***strength*** of "*faster than* $2^n$" algorithms!

***A "quicker" algorithm can tell when a given circuit computes the all-zeroes function!***




**Therefore, Faster-Than-$2^n$ time Algorithms are "strong" and Small Circuits are "weak"…**

***so we can construct an "algorithmically-defined function" which doesn't have small circuits***

# A Concrete Lower Bound From Algorithms

**Thm [Murray-W'18]: Quasi-NP $\not\subset$ ACC$^0$**

Quasi-NP = NTIME[$2^{\log^{O(1)} n}$]

ACC$^0$: ~~polynomial size, constant depth circuits with AND, OR, and MOD[m] gates for some constant m.~~

**A simple but Annoying Circuit Class to prove lower bounds for (proposed in 1986 by Barrington)**

# How Quasi-NP $\not\subset$ $ACC^0$ Was Proved

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

**Thm A [MW'18] (algorithm design → lower bounds)**

**If for some $\varepsilon > 0$, Gap-$\mathbb{C}$-SAT on $2^{n^\varepsilon}$ size is in $O(2^{n-n^\varepsilon})$ time, then Quasi-NP does not have poly-size $\mathbb{C}$-circuits.**

**Thm B [W'11] (algorithm)**

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

*[Uses a representation theorem for $ACC^0$ from 1990, that people long suspected should imply lower bounds!]*

# More on Theorem A

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

**Thm A [MW'18]:**

**If for some $\varepsilon > 0$, Gap-$\mathbb{C}$-SAT on $2^{n^\varepsilon}$ size is in $O(2^{n-n^\varepsilon})$ time, then Quasi-NP does not have poly-size $\mathbb{C}$-circuits.**

**Idea. Show that if we assume both:**

**(1) Quasi-NP has poly-size $\mathbb{C}$-circuits,**

**and**

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

**Then show $\exists k$ NTIME$[n^{\log^k n}] \subseteq$ NTIME$[o(n^{\log^k n})]$**

***Contradicts the nondeterministic time hierarchy:***

***there is $L_{hard}$ in*** **NTIME$[n^{\log^k n}] \setminus$ NTIME$[o(n^{\log^k n})]$**