Algorithms and Lower Bounds: Some Basic Connections
Lecture 1: Circuit Analysis

Ryan Williams
Stanford University

A “conventional” 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 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 is generally believed that
lower bounds are *"harder"* than algorithm design

- In algorithm design, we "only" have to find a single clever algorithm that solves a problem well
- In lower bounds, we must reason about *all possible* algorithms, and argue that none of them work well

This belief is strongly reflected in the literature

"Duality" Between Circuit Analysis Algorithms and Circuit Lower Bounds

Thesis: **Algorithm design can be *as hard as* proving lower bounds.**

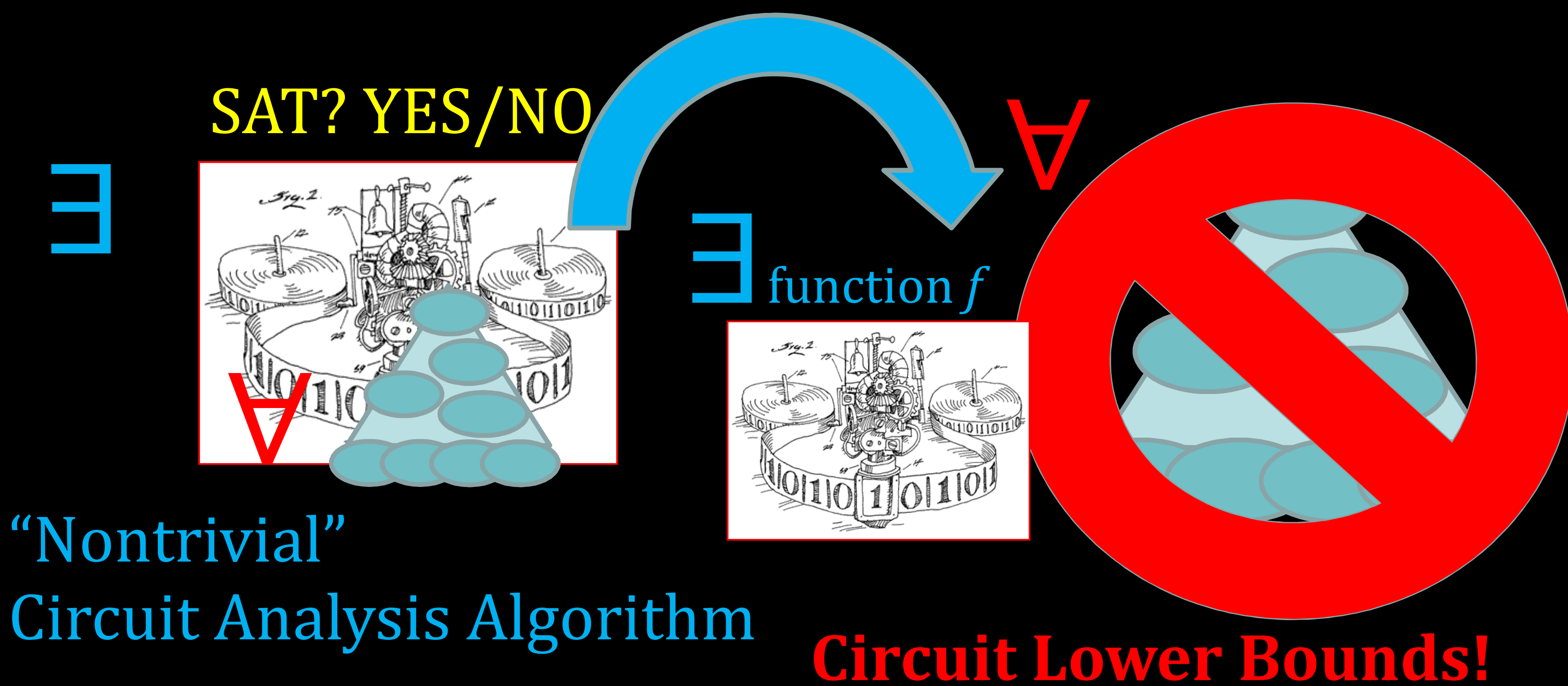
There are deep connections between the two…
so deep that they are often the "same"

A typical theorem from Algorithm Design:
"Here is an algorithm A that solves my problem,
on all possible instances of the problem"

A typical theorem from Lower Bounds:
"Here is a proof P that my problem cannot be solved,
on all possible algorithms from some class"

"Duality" Between Circuit Analysis Algorithms and Circuit Lower Bounds

Thesis: Algorithm design can be *as hard as* proving lower bounds.

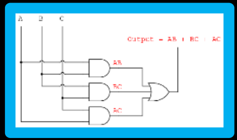


Outline of the Lectures

- Circuit Analysis (Algorithms)
- Circuit Complexity (Lower Bounds)
- Connections
- NEXP not in ACC

Circuit Analysis Problems

Circuit Analysis problems are often computational problems *on* circuits given as input:

Input: A logical circuit C = 

Output: Some property of the function computed by C

Canonical Example: Circuit Satisfiability Problem (Circuit SAT)

Input: Logical circuit C

Decide: Is the function computed by C the "all-zeroes" function?

Of course, Circuit SAT is NP-complete

But we can still ask if there are *any* algorithms solving Circuit SAT that are faster than the obvious "brute-force" algorithm which tries all 2^n input settings to the n inputs of the circuit.

Generic Circuit Satisfiability

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

$\mathcal{C}$ = {formulas}, $\mathcal{C}$ = {arbitrary circuits}, $\mathcal{C}$ = {CNF formulas}

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

$\mathcal{C}$**-SAT** is NP-complete, for essentially all interesting $\mathcal{C}$

$\mathcal{C}$**-SAT** is solvable in **$O(2^n\ |K|)$ time**
where **|K| is the size of the circuit K**

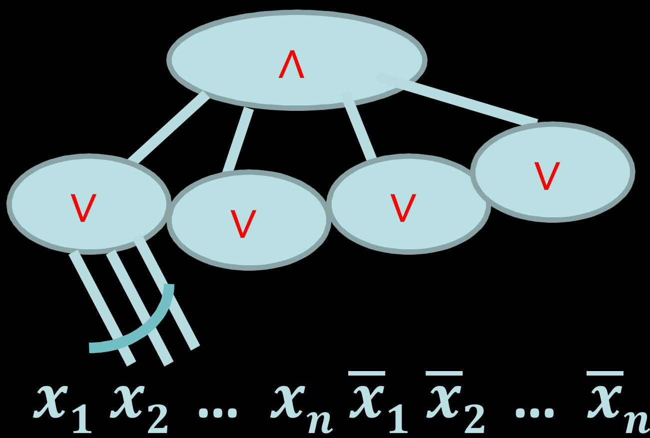
Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

- **3-SAT** 1.308^n
- **4-SAT** 1.469^n
- **k-SAT**

$2^{n - n/O(k)}$ time algorithms

[many authors …, Hertli '11]



All known c^n time algorithms for k-SAT have the property that, as k → ∞, the constant c → 2

Strong ETH: $\forall \delta < 1, \exists k \geq 3$ s.t. k-SAT requires $2^{\delta n}$ time

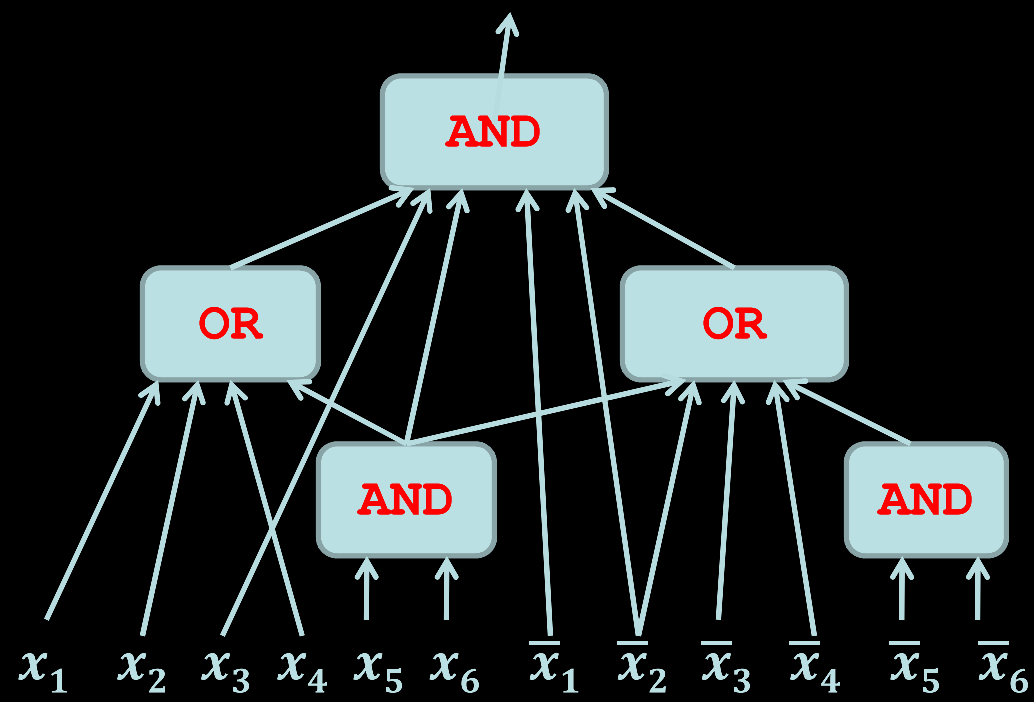
ETH: $\exists \delta > 0$ s.t. 3-SAT requires $2^{\delta n}$ time

Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

- **AC0-SAT** Constant-depth AND/OR/NOT

[IMP '12] AC0-SAT in $2^{n - n/(c \log s)^{d-1}}$ time where d = depth
s = size

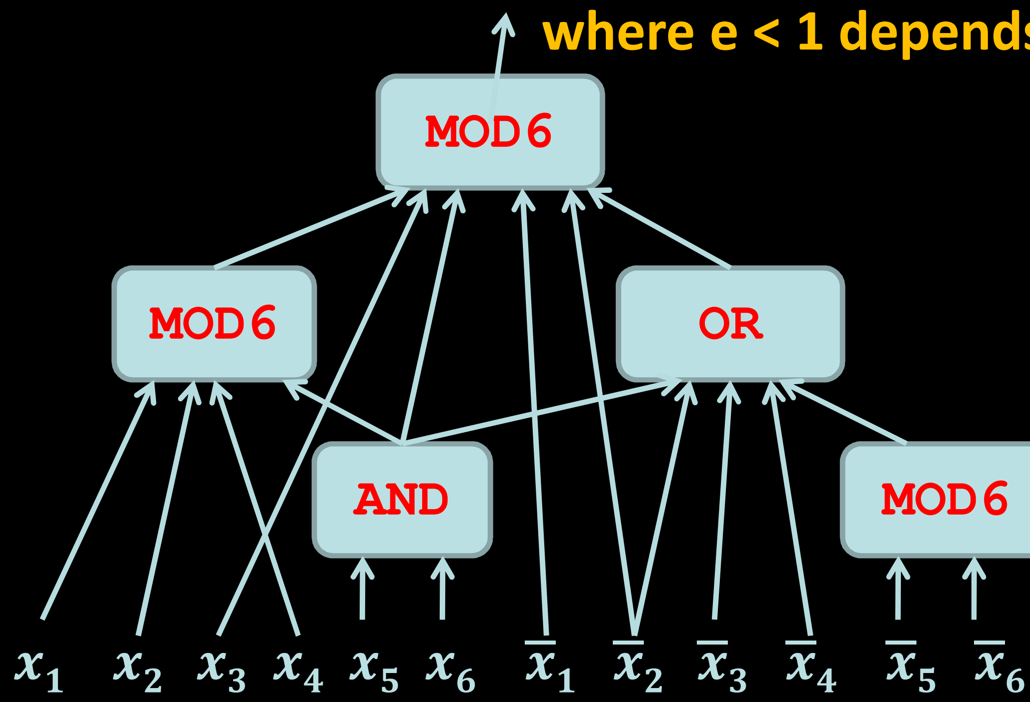


Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

- **ACC-SAT** Constant-depth AND/OR/NOT/MODm
  MOD6($x_1, \ldots, x_t$) = 1 iff $\sum_i x_i$ is divisible by 6

[W '11] ACC-SAT in $2^{n - n^e}$ time for circuits of size $2^{n^{o(1)}}$
where e < 1 depends on d and m



Ingredients for Solving ACC SAT

1. **A known representation of ACC**
   [Yao '90, Beigel-Tarui'94] Every **$f : \{0,1\}^n \rightarrow \{0,1\}$ with an ACC circuit of size s** can be expressed in the form

   $$f(x_1,\ldots,x_n) = g(h(x_1,\ldots,x_n))$$

   - **h** is a multilinear polynomial with **K** monomials, $h(x_1,\ldots,x_n) \in \{0,\ldots,K\}$ **for all** $(x_1,\ldots,x_n) \in \{0,1\}^n$
   - $K = s^{poly(\log s)}$
   - $g : \{0,\ldots,K\} \rightarrow \{0,1\}$ can be an arbitrary function

2. **"Fast Fourier Transform" for multilinear polynomials:**
   Given a multilinear polynomial h in its coefficient representation, the value h(x) can be computed over all points $x \in \{0,1\}^n$ in **2^n poly(n)** time.

1. Polynomials Representing ACC

Very special cases:

1. **Writing $OR(x_1, \ldots, x_n)$ as a g of h:**
   $g(y) = 1$ iff $y > 0$, $h = x_1 + \ldots + x_n$
2. **Writing $AND(x_1, \ldots, x_n)$ as a g of h**
   $g(y) = 1$ iff $y = n$, $h = x_1 + \ldots + x_n$
3. **Writing $MODm(x_1, \ldots, x_n)$ as a g of h…**

1. Polynomials Representing ACC

A less special case:

Theorem [Razborov-Smolensky'87] For every AC0 circuit C with n inputs, size s, and depth d, there is an efficiently samplable distribution D(C) of polynomials of degree $(\log s)^{O(d)}$ over $\mathbb{F}_2$ such that

$$\text{For all } x \in \{0,1\}^n, \Pr_{p \sim D(C)}[p(x) = C(x)] > 3/4.$$

In fact can use a "small" number **S** of polynomials **(S = $n^{\text{poly}(\log s)}$)**

Can take MAJORITY value of all **S** different polynomials over $\mathbb{F}_2$.

Can write the "MAJORITY of XORs" as a symmetric Boolean function. This yields the g of h's. [Yao, Toda, Beigel-Tarui]

1. Reducing AC0[⊕] to polynomials

Theorem [Razborov-Smolensky'87] For every AC0[⊕] circuit C with n inputs, size s, and depth d, there is an efficiently samplable distribution D(C) of polynomials of degree $(\log s)^{O(d)}$ over $\mathbb{F}_2$ such that

For all $x \in \{0,1\}^n$, $\Pr_{p \sim D(C)}[p(x) = C(x)]$ > ¾.

Proof Idea: Induction on the depth d.

NOT gate: $NOT(x_i) = 1 + x_i$

XOR gate: $XOR(x_1, \ldots, x_n) = \sum_i x_i \bmod 2$.

OR gate: For all $x \in \{0,1\}^n$, observe that

$$\Pr_{r \in \{0,1\}^n}[OR(x_1, \ldots, x_n) = \sum_i r_i x_i \bmod 2] \geq ½$$

Pick $R \in \mathbb{F}_2^{k \times n}$ at random, where k = error parameter

For all $x \in \{0,1\}^n$,

$$\Pr_R\left[OR(x_1, \ldots, x_n) = 1 + \prod_j \left(1 + \sum_i R_{j,i}\, x_i\right) \bmod 2\right] \geq 1 - \frac{1}{2^k}$$

This is a degree-k polynomial simulating OR with error < $1/2^k$.

2. Fast Multipoint Evaluation

Theorem: Given the **2^n** coefficients of a multilinear polynomial **h** in **n** variables, the value **h(x)** can be computed on all points **x** $\in$ **$\{0,1\}^n$** in **2^n poly(n)** time.

Can write **$h(x_1, \ldots, x_n) = x_1 h_1(x_2, \ldots, x_n) + h_2(x_2, \ldots, x_n)$**

Want a 2^n table T that contains the value of h on all 2^n points.

Algorithm: If n = 1 then return T = [h(0), h(1)]

Recursively compute the 2^{n-1} table T_1 for the values of h_1, and the 2^{n-1} table T_2 for the values of h_2

Return the table $T = (T_2)(T_1 + T_2)$ of 2^n entries

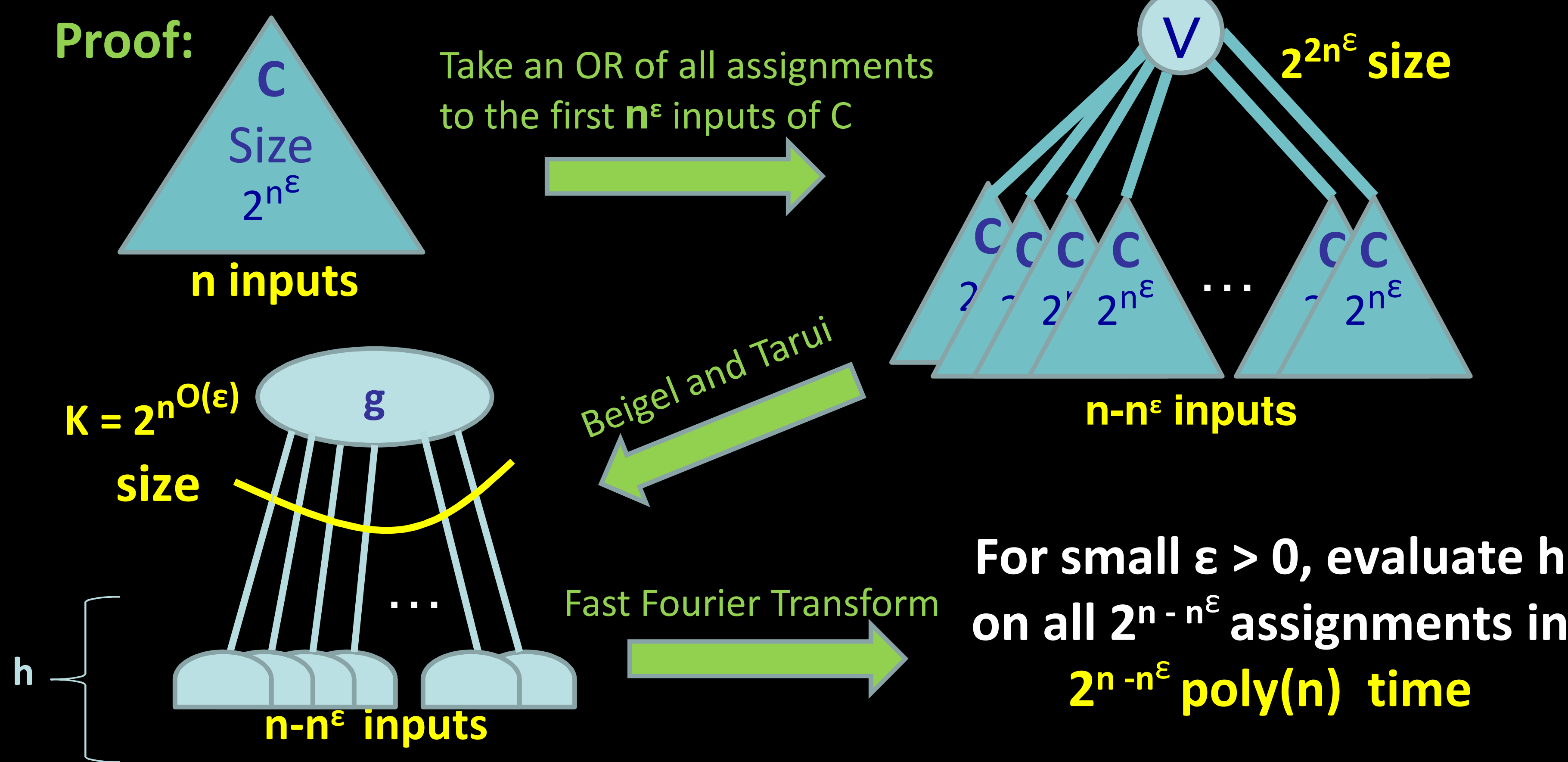
Running time has the recurrence **$R(2^n) \leq 2R(2^{n-1}) + 2^n \text{poly}(n)$**

Corollary: We can compute g of h on all x $\in$ $\{0,1\}^n$ in only 2^n poly(n) time

ACC Satisfiability Algorithm

Theorem For all d, m there's an $\varepsilon > 0$ such that ACC[m] SAT with depth d, n inputs, 2^{n^ε} size can be solved in $2^{n - \Omega(n^\varepsilon)}$ time

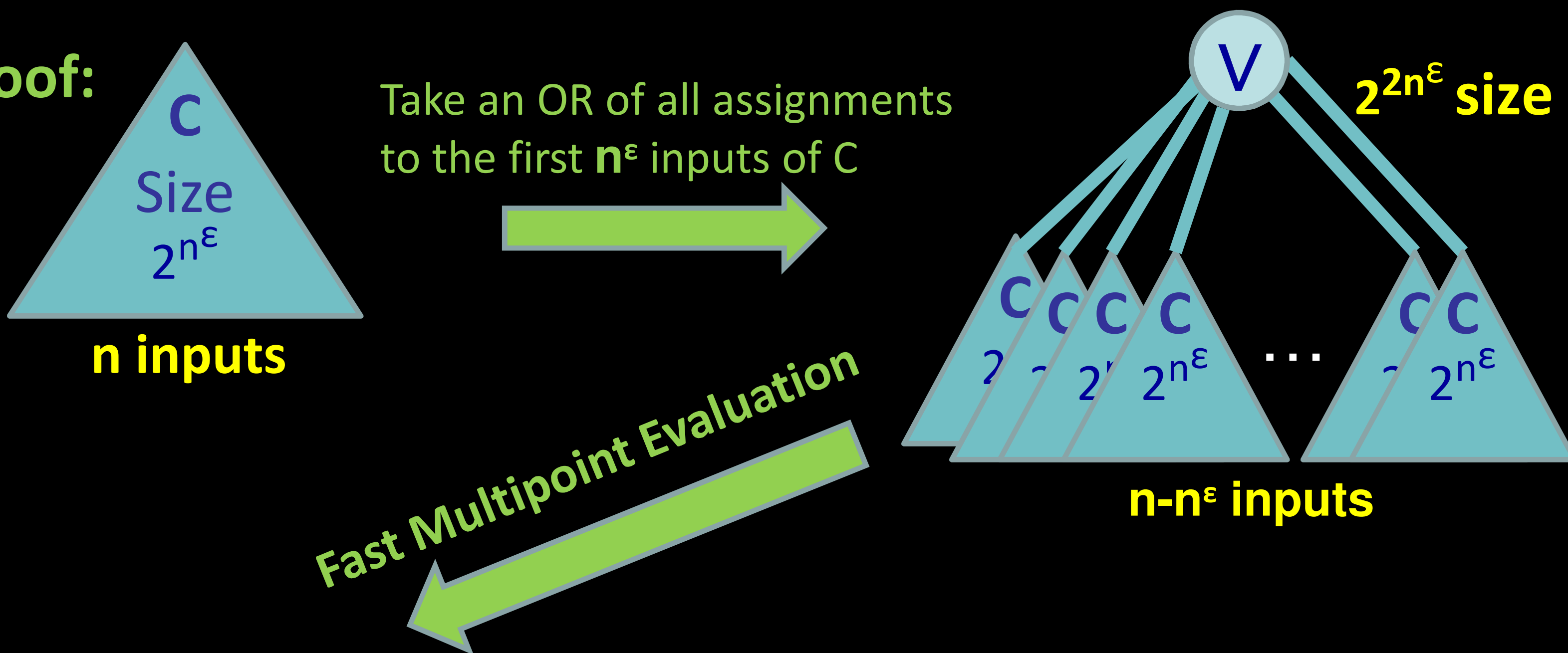
Proof:



Fast Multipoint Circuit Evaluation $\Rightarrow$ Circuit SAT algorithms

Theorem If we can **evaluate a circuit of size s on all 2^n inputs** in **2^n poly(n) + poly(s)** time, then **Circuit-SAT is in $o(2^n)$ time**

Proof:



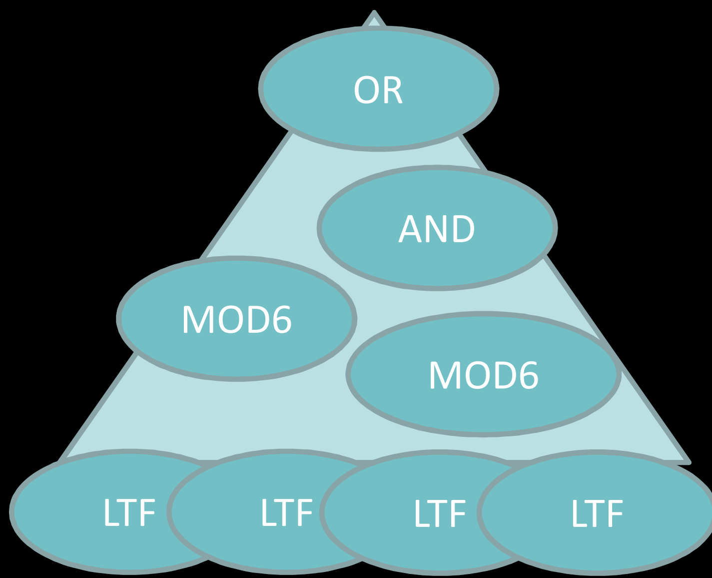
For small ε > 0, can evaluate on all
$2^{n - n^{\varepsilon}}$ assignments in
$2^{n - n^{\varepsilon}}$ poly(n) + poly($2^{2n^{\varepsilon}}$) time

Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

- **ACC-THR-SAT** Constant-depth AND/OR/NOT/MODm with a layer of linear threshold fns at the bottom

[W '14] ACC-THR-SAT is in $2^{n - n^e}$ time for circuits of size $2^{n^{o(1)}}$



[IPS'13] THR-THR-SAT in $2^{n(1 - e)}$ time for circuits with O(n) wires

Circuit SAT Algorithms

- **DeMorgan-Formula-SAT**
  Formulas over AND/OR/NOT, each gate has fan-in at most 2
  [Santhanam '10, CKKSZ '14]
  DM-Formula-SAT is in 2^{n-n^e} time for formulas of size < $n^{2.99}$

- Formulas over AND/OR/NOT/XOR with fan-in two
  [Seto-Tamaki '12, CKKSZ '14]
  Formula-SAT is in 2^{n-n^e} time for formulas of size < $n^{1.99}$

- **Circuit-SAT** Generic circuits over AND/OR/NOT, fan-in 2

OPEN: *Can we improve on $O(2^n s)$ time ??*

Circuit Approximation Probability Problem

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

$\mathcal{C}$-CAPP:

Given a circuit **K**(**x**$_1$,…,**x**$_n$) ∈ $\mathcal{C}$, output ***v*** such that

$$|\boldsymbol{v} - \Pr_x[K(\mathbf{x}) = 1]| < 1/10$$

Related to Pseudorandom Generators and Derandomization

[AW'85, Nisan'91, TX'13] AC0-CAPP is in $n^{\tilde{O}(\log^{d+4} s)}$ time

(n = inputs, s = size, d = depth)

[GMR'12] CNF-CAPP is in ~ $n^{O(\log\log n)}$ time for poly(n) clauses

[IMZ'12] DM-Formula-CAPP: $2^{n^{e}}$ time for formulas of size < $n^{2.99}$

Formula-CAPP: $2^{n^{e}}$ time for formulas of size < $n^{1.99}$

Uses old techniques from *lower bounds!*

Circuit Analysis Problems

Circuit Analysis problems can also analyze functions *directly*:

Canonical Example:

Minimum Circuit Size Problem (MCSP) [Yablonski '59, KC'00]

Input: 2^n-bit truth table of $f : \{0,1\}^n \rightarrow \{0,1\}$, $s \in \{1,\ldots,2^n\}$,

Decide: Is the minimum size of a circuit computing *f* at most *s*?

(Note: MCSP is in NP)

It is widely conjectured that MCSP is *not* in P

If in P: Would contradict conventional wisdom in cryptography

Known: [Masek'79, AHMPS'08] DNF Minimization is NP-complete

(uses lower bounds on DNF!)

Is the MCSP problem NP-complete? [MW'15]

Open: Find *any* improvement over exhaustive search

Circuit Analysis Problems

Circuit Minimization (MCSP) [Yablonski '59, KC'00]

Input: Truth table of a Boolean function f, parameter s

Decide: Is the minimum size of a circuit computing f at most s?

[ABKvMR '06] Factoring is in $\mathbf{ZPP}^{\text{Circuit Min}}$

[ABKvMR '06] Discrete Log is in $\mathbf{BPP}^{\text{Circuit Min}}$

[Allender-Das '14] Graph Iso is in $\mathbf{RP}^{\text{Circuit Min}}$

Some open problems:

- Find interesting problems in $\mathrm{P}^{\mathrm{MCSP}}$
- In $\mathrm{P}^{\mathrm{MCSP}}$ can we *produce* a min-size circuit, given a truth table?
- How hard is MCSP for AC0 circuits?

Exponential Time Algorithms

This topic of “Algorithms for Circuits” is one tiny part of the growing area of

Exact algorithms for NP-hard problems

This is a very active research area with many cool open problems.

Outline of the Lectures

- Circuit Analysis (Algorithms)
- Circuit Complexity (Lower Bounds)
- Connections
- NEXP not in ACC

End of Lecture 1