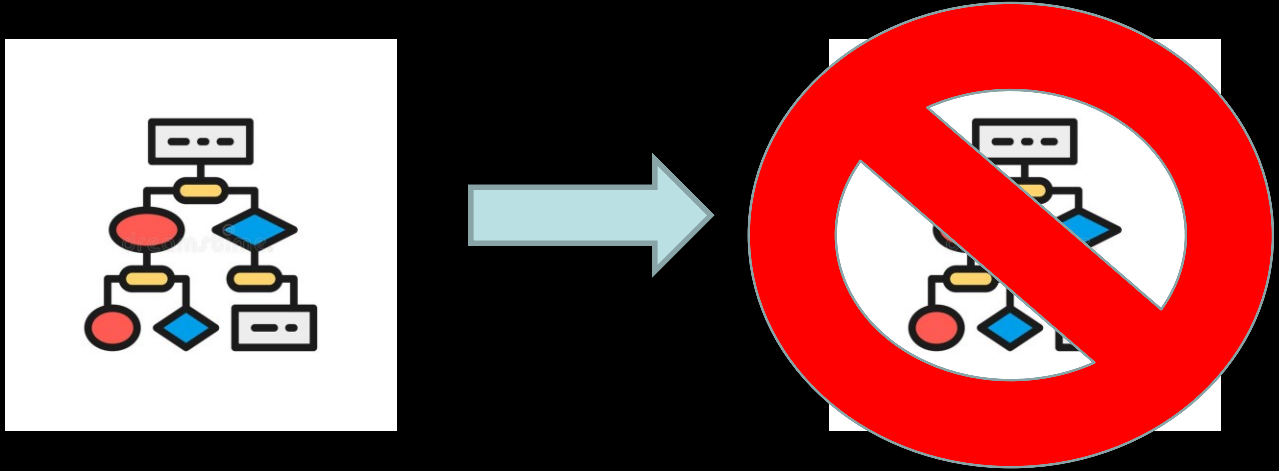
Outline

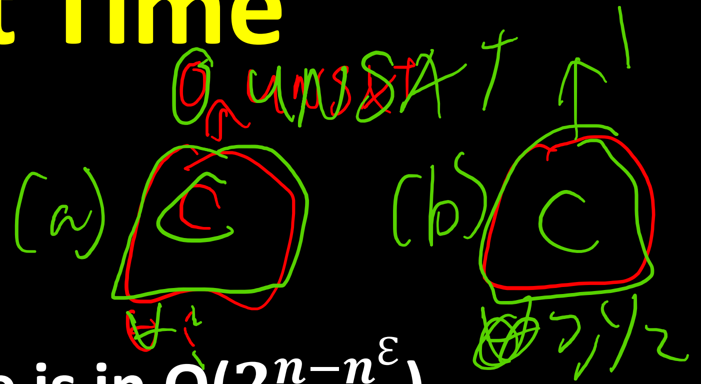
Lecture 1: Overview of Circ LBs from Algorithms
Lecture 2-3: More on Circ LBs from Algorithms
Lecture 3: The Mysteries of the Missing String
Lecture 4: The Power of Constructing Bad Inputs

How to Prove Lower Bounds With Algorithms



Lecture 2: Overview of Circuit Lower Bounds From Circuit-Analysis Algorithms

Picking Up From Last Time



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^ε} size is in O(2^{n-n^ε}) 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[$n^{\log^k n}$] \ NTIME[o($n^{\log^k n}$)]

Proof Ideas of Theorem A

Idea. **Assume:**

(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$[o(n^{log^k n})]$

Take an L in **nondeterministic $n^{log^k n}$ time.**
Given an input x, we decide if $x \in L$, by:

(A) Guessing some witness y of $O(n^{log^k n})$ length.

(B) Checking y is a witness for x in $O(n^{log^k n})$ time.

Proof Ideas of Theorem A

Idea. **Assume:**

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

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

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

Take an L in **nondeterministic $\boldsymbol{n^{log^k n}}$ time**.

Given an input x, we **will** decide if $x \in L$, by:

(A) Guessing some witness y of o($\boldsymbol{n^{log^k n}}$) length.

(B) Checking y is a witness for x in o($\boldsymbol{n^{log^k n}}$) time.

Proof Ideas of Theorem A

Idea. **Assume:**

(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$[o(n^{log^k n})]$

Take an L in **nondeterministic $n^{log^k n}$ time.**
Given an input x, we **will** decide if $x \in L$, by:

(A) Guessing some witness y of $o(n^{log^k n})$ length.

(B) Checking y is a witness for x in $o(n^{log^k n})$ time.

Guessing Short Witnesses

1. Guess a witness y of $o(n^{\log^k n})$ length.

Easy Witness Lemma [IKW'02, MW'18]:
If NEXP (Quasi-NP) has polynomial-size circuits, then all NEXP (Quasi-NP) problems have "easy witnesses"

Small circuits for solving Quasi-NP problems
→ **Small circuits encoding *solutions* to Quasi-NP problems**

Def. An NEXP/Quasi-NP problem L has easy witnesses if
∀ Verifiers V for L and $\forall x \in L$,
∃ poly($|x|$)-size circuit D_x such that V(x,Y) accepts,
where **Y = Truth Table of circuit D_x.**

$D(0^n)\ D(0^{n-1}1)\ \ldots\ D(1^n)$

1'. Guess $poly(n)$-size circuit D_x

Verifying Short Witnesses

2. Check y is a witness for x in $o(n^{\log^k n})$ time.

Assuming Quasi-NP has polynomial-size circuits, “easy witnesses” exist for ***every*** verifier V.

We choose a verifier V for $L \in NTIME[n^{\log^k n}]$ so that:

$2^{\left(\log^{k+1}(n)+4\log(n)\right)^{\varepsilon}}$

Checking V(x, y) accepts for $|x| = n$

is equivalent to

Solving UNSAT on a $\mathbb{C}$**-circuit** with $2^{m^{\varepsilon}}$ size and $m = \log^{k+1}(n) + 4\log(n)$ inputs

Then, $2^{m-m^{\epsilon}}$ time for $\mathbb{C}$-UNSAT → $o\left(n^{\log^k n}\right)$ time to decide L

Verifying Short Witnesses

Hi

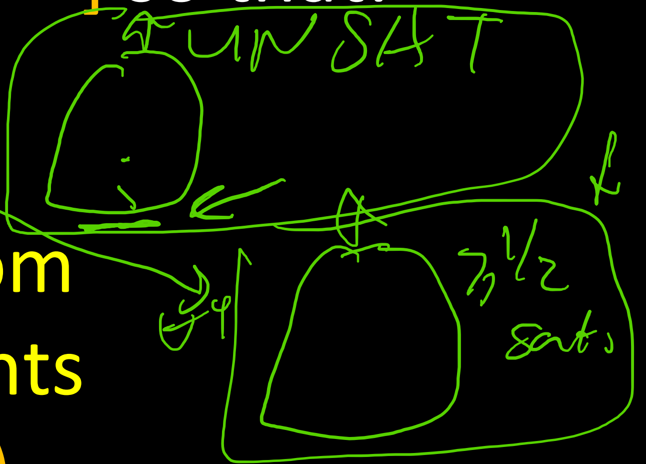
2. Check y is a witness for x in o($n^{\log^k n}$) time.

Assuming Quasi-NP has polynomial-size circuits,
"easy witnesses" exist for ***every*** verifier V.

We can also choose a verifier V for $L \in NTIME[n^{\log^k n}]$ so that:

∀y rejects

Checking $V(x, y)$ accepts ≡
Distinguishing ***unsatisfiable*** circuits from
circuits with ***many*** satisfying assignments
(Uses a version of the PCP Theorem!)



Then, $2^{n-n^{\epsilon}}$ time for Gap-ℂ-UNSAT ➔ $o\left(n^{\log^k n}\right)$ time to decide L

Now: Time for Details

Definition: ACC Circuit Family

An **ACC circuit family** $\{ C_n \}$ has the properties:

- Every C_n takes n bits of input and outputs one bit
- There is a fixed d such that every C_n has depth at most d
- There is a fixed m such that the gates of C_n are

  AND, OR, NOT, MODm (unbounded fan-in)

  MOD$m(x_1, \ldots, x_t) = 1$ iff $\sum_i x_i$ is divisible by m

Alternating
Circuits
With **C**ounters

Remarks

1. The default *size* (#gates) of C_n is **polynomial in n**
2. ***Strength:*** this is a ***non-uniform*** model of computation *(can compute some undecidable languages!)*
3. ***Weakness:*** ACC circuits can be efficiently simulated by ***constant-layer neural networks (a.k.a. TC0)***

Definition: ACC Circuit Family

An **ACC circuit family** { C_n } has the properties:

- Every C_n takes n bits of input and outputs one bit
- There is a fixed d such that every C_n has depth at most d
- There is a fixed m such that the gates of C_n are

AND, OR, NOT, MODm (unbounded fan-in)

MOD$m(x_1, \ldots, x_t) \ = \ 1$ iff $\sum_i x_i$ is divisible by m

Alternating
Circuits
With **C**ounters

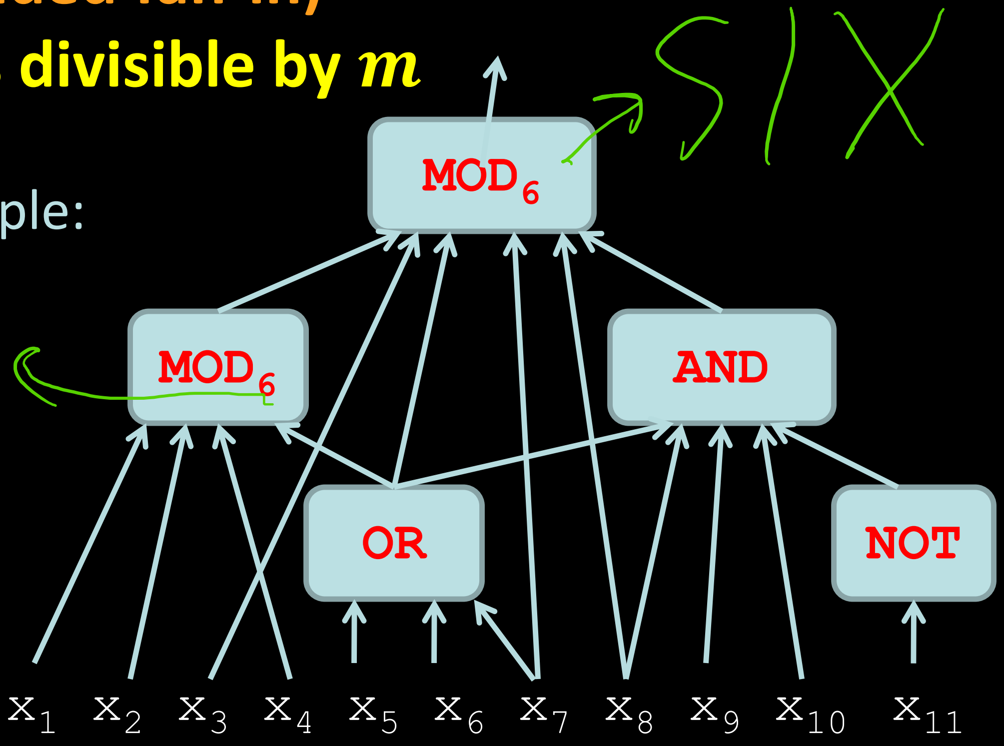
Note: These circuits become very complex, already for certain fixed d and m.

OPEN: Does every problem in EXP have polynomial-size MOD6 circuits of depth 3 (?!)

ACC does have some surprising power:

[CW'22] For ***every*** $\varepsilon > 0$, every symmetric Boolean fn has $2^{n^{\varepsilon}}$ size depth-3 ACC circuits

Example:



Where does ACC come from?

Dream of the 1980s: Prove **P ≠ NP** by proving **NP ⊄ P/poly**.

Unlike Turing Machines, logic circuits are fixed, "simple" devices.
This should make it easier to prove impossibility results.

Ajtai, Furst-Saxe-Sipser, Håstad (early 80's)

MOD2 ∉ AC0 [poly-size **ACC** with ***only*** AND, OR, NOT, *no* MOD*m*]

Razborov, Smolensky (late 80's)

MOD3 ∉ (AC0 with MOD2 gates)

MOD3, MOD2 ≤ MOD6

primes p ≠ q

For p ≠ q prime, **MODp ∉ (AC0 with MODq** gates)

p^k q^l

Barrington (late 80's) Suggested **ACC** as the next natural step

Conjecture Majority ∉ **ACC**

Conjecture (early 90's) **NP ⊄ ACC**

Conjecture (late 90's) **NEXP ⊄ ACC**

ACC Lower Bounds

$2^{O(n)} \rightarrow$ SAT?

EXP$^{\text{NP}}$ = Exponential Time with an NP oracle [think: SAT oracle]
NEXP = Nondeterministic Exponential Time

Theorem [W'11] There is an f ∈ **EXP$^{\text{NP}}$** such that for every d, m there is an ε > 0 such that $\boldsymbol{f}$ does not have ACC circuits with MODm gates, depth d, and size $\mathbf{2^{n^{\varepsilon}}}$ $\Leftarrow m, d$

Theorem [W'11] There is an f ∈ **NEXP** such that for all d, m, k,
$\boldsymbol{f}$ does not have $\boldsymbol{n^{\log^k n}}$ size ACC circuits of depth d with MODm gates

$2^{n^{\varepsilon}}$

Remark Compare with:
[MS 70's] **EXP$^{(\text{NP}^{\text{NP}})}$** = $EXP^{\Sigma_2 P}$ doesn't have **o(2^n/n)** size circuits
[K82] **NEXP$^{\text{NP}}$** = $\Sigma_2 E$ doesn't have $\boldsymbol{n^{\log^k n}}$-size circuits for all k

ACC Lower Bounds

TIME $2^{O(n)}$ w/ SAT oracle

Quasi-NP = Nondeterministic $n^{polylog\ n}$ Time

Theorem [MW'18] There is an f in **Quasi-NP** such that for all d, m, k, f does not have $\boldsymbol{n^k}$ size ACC circuits of depth d with MODm gates

Has since been extended in multiple ways!
(stronger circuit classes, average-case hardness, etc etc)

We'll outline a different result, and then sketch how to extend it.

Theorem There is an f in $\boldsymbol{E^{NP}} = \boldsymbol{TIME^{SAT}[2^{O(n)}]}$ such that for all d, m, there is an $\varepsilon > 0$ such that f does not have 2^{n^ε} **size** ACC circuits of depth d with MODm gates

Proof Outline

ACC ∘ THR

ACC

THR ... THR

Design a faster ACC-SAT algorithm

The Algorithm: For every d, m, there is an $\varepsilon > 0$ such that #ACC-SAT on circuits with n inputs, 2^{n^ε} size, depth d, and MODm gates is solvable in 2^{n-n^ε} time $2^n / 2^{n^\varepsilon}$

This algorithm has changed little in the past 9 years…

Show that faster ACC-SAT algorithms imply lower bounds against ACC

The LB Connection: If $\mathcal{C}$-SAT on circuits with n inputs and 2^{n^ε} size is in $O(2^n/n^{10})$ time, then **E^{NP} doesn't have 2^{n^ε} size $\mathcal{C}$-circuits.**

The connections have strengthened considerably!

Algorithm for SAT on ACC Circuits

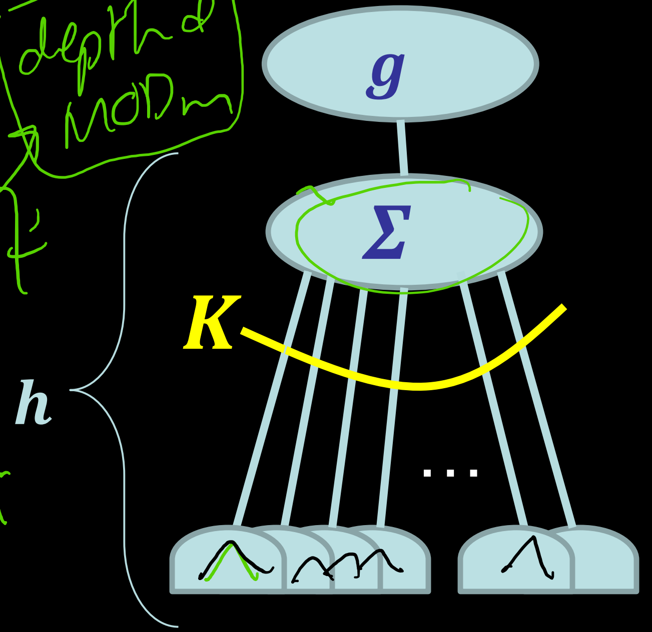
Ingredients:

1. **Old representation** [Yao'90, Beigel-Tarui'94,Green et al'95]
   For every ACC function $f : \{0,1\}^\star \to \{0,1\}$ and every $\boldsymbol{n}$, we can write $f_n : \{0,1\}^n \to \{0,1\}$ as:
   $$\boldsymbol{f_n(x_1, \dots, x_n) = g(h(x_1, \dots, x_n))}, \text{ where}$$
   - $\boldsymbol{h}$ is a multilinear polynomial of at most K monomials, $h(a) \in \{0, \dots, K\}$ for all $a \in \{0,1\}^n$
   - K is not "too large" *(quasi-polynomial in circuit size)*
   - $\boldsymbol{g} : \{0, \dots, K\} \to \{0,1\}$ is a fixed "simple" function
2. **"Fast Fourier Transform" for multilinear polynomials:**
   Given a multilinear polynomial h in its coefficient representation, the value $h(a)$ can be computed over all points $a \in \{0,1\}^n$ in $\boldsymbol{2^n\, poly(n)}$ time. $2^n \cdot 2^n$

depth d MOD_m

$\{C_n\}$

$c_S \prod x_i$, $i \in S$



[Chen-Papakonstantinou'19]

$K \leq 2^{(\log s)^{O(dr)}}$ t

where s = size, d = depth, r = # prime divisors of m

Fast Multipoint Evaluation

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

Can write: $\boldsymbol{h(x_1, \ldots, x_n) = x_1\, h_1(x_2, \ldots, x_n) + h_2(x_2, \ldots, x_n)}$

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

Algorithm: If $n = 1$ then return $T = [\boldsymbol{h(0), h(1)}]$

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

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

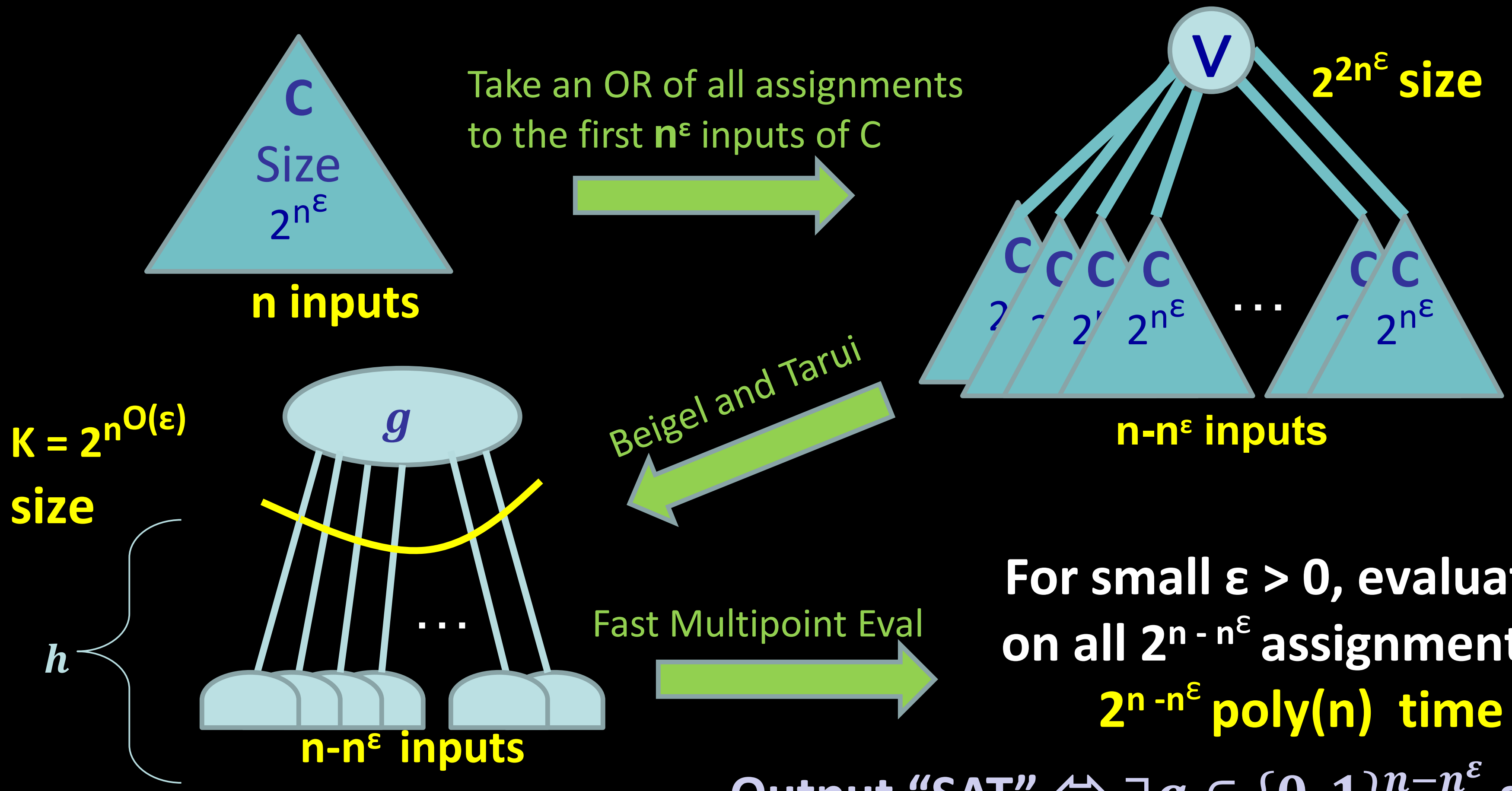
Running time has the recurrence $\boldsymbol{R(2^n) \leq 2 \cdot R(2^{n-1}) + 2^n\, poly(n)}$

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

ACC Satisfiability Algorithm

Theorem: For every d, m, there is an $\varepsilon > 0$ such that ACC-SAT on circuits with n inputs, $2^{n^{\varepsilon}}$ size, depth d, and MODm gates is solvable in $2^{n-n^{\varepsilon}}$ time

Proof:



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

Output "SAT" ⇔ $\exists a \in \{0,1\}^{n-n^{\varepsilon}}$ s.t. $g(h(a)) = 1$

The LB Connection: If *ACC*-SAT on circuits with n inputs and $2^{n^{\varepsilon}}$ size is in $O(2^n/n^{10})$ time, then **E^{NP} doesn't have $2^{n^{\varepsilon}}$ size *ACC*-circuits.**

Given circuit $\boldsymbol{C} : \{\mathbf{0},\mathbf{1}\}^{\boldsymbol{n}} \to \{\mathbf{0},\mathbf{1}\}$, let $\boldsymbol{tt}(\boldsymbol{C})$ be its truth table: the output of $\boldsymbol{C}$ on all 2^n assignments, in lexicographical order

n inputs
n^{10} size

Succinct 3SAT: *Given a circuit C, does $tt(C)$ encode a satisfiable 3CNF?*

Key Idea: Succinct 3SAT is NEXP-complete, in a very strong way…

Lemma 1 Succinct 3SAT for AC0 circuits of n inputs and n^{10} size is solvable in nondeterministic $2^n\, poly(n)$ time but ***not*** in nondeterministic $\frac{2^n}{n^5}$ time.

Upper bound: Evaluate the AC0 circuit on all 2^n inputs, get a 2^n-length 3CNF instance, guess and check a SAT assignment, in $2^n\, poly(n)$ time

Lower bound: **[JMV'13]** Every $L \in NTIME[2^n]$ can be reduced in poly-time to a Succinct 3SAT instance which is AC0, $m = n + 4\log(n)$ inputs, n^{10} size

So, if Succinct3SAT is in $2^m/m^5$ time, then L can be decided in time $o(2^n)$

Contradicts the nondeterministic time hierarchy theorem!

The LB Connection: If $\mathbf{E^{NP}}$ **has** $2^{n^{\varepsilon}}$ **size** ***ACC*-circuits** and *ACC*-SAT on circuits with n inputs and $2^{n^{\varepsilon}}$ size is in $O(2^n/n^{10})$ time, then *contradiction*

Succinct 3SAT: ***Given a circuit C, does $tt(C)$ encode a satisfiable 3CNF?***

Lemma 1 Succinct 3SAT for ACC circuits of n inputs and n^{10} size is solvable in nondeterministic $2^n\, poly(n)$ time but ***not*** in nondeterministic $\frac{2^n}{n^5}$ time.

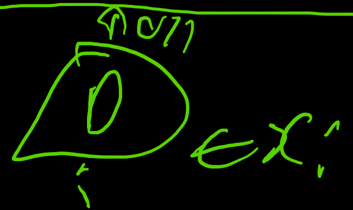
Goal: Use ACC circuits for E^{NP} & the ACC-SAT algorithm, to solve Succinct 3SAT faster.

Say that **Succinct 3SAT has "succinct" SAT assignments** if

for **every** $\boldsymbol{C}$ (of n inputs and n^{10} size) such that $\boldsymbol{tt(C)}$ encodes a satisfiable 3CNF $\boldsymbol{F}$,

there is an ACC circuit $\boldsymbol{D}$ of $\mathbf{2^{n^{10\varepsilon}}}$ **size** such that

$\boldsymbol{tt(D)}$ encodes a variable assignment $\boldsymbol{A}$ that satisfies $\boldsymbol{F}$.



(Imagine F has variables $x_1, \ldots, x_{2^n}$. Then $D(i)$ outputs a 0-1 assignment to variable x_i in F)

If a succinct SAT assignment exists, we only have to guess a witness of length $\mathbf{2^{n^{10\varepsilon}}}$

Lemma 2 If $\mathbf{E^{NP}}$ has $2^{n^{\varepsilon}}$ size ACC circuits then

Succinct 3SAT has "succinct" SAT assignments

The LB Connection: If **E^NP has $2^{n^{\varepsilon}}$ size *ACC*-circuits** and *ACC*-SAT on circuits with n inputs and $2^{n^{\varepsilon}}$ size is in $O(2^n/n^{10})$ time, then *contradiction*

Succinct 3SAT: *Given a circuit C, does $tt(C)$ encode a satisfiable 3CNF?*

Lemma 2 If **E^NP** has $2^{n^{\varepsilon}}$ size ACC circuits then **Succinct 3SAT has "succinct" SAT assignments**

Proof The following is an **E^NP** procedure: → i'th variable

$\leq |C| + n$

On input (C, i), where $i \in \{1, \dots, 2^n\}$, C has n inputs & n^{10} size

Compute $F = tt(C)$, think of F as a 3CNF formula. ← $2^{O(n)}$

Use a SAT oracle and search-to-decision for SAT, to find the lexicographically first SAT assignment to F.

Output the i-th bit of this assignment. ←

E^NP has $2^{n^{\varepsilon}}$ size ACC circuits ⇒ there is a $2^{|C|^{\varepsilon}} \leq 2^{n^{10\varepsilon}}$ size ACC circuit $\boldsymbol{D(C, i)}$ which outputs the $\boldsymbol{i}$-th bit of a satisfying assignment to $\boldsymbol{F = tt(C)}$.

Now for any circuit **C' of $\boldsymbol{n^{10}}$** size, define the circuit **E(i) := D(C', i)**

Then **E** has $2^{n^{10\varepsilon}}$ size, and the assignment **tt(E)** satisfies **tt(C')**