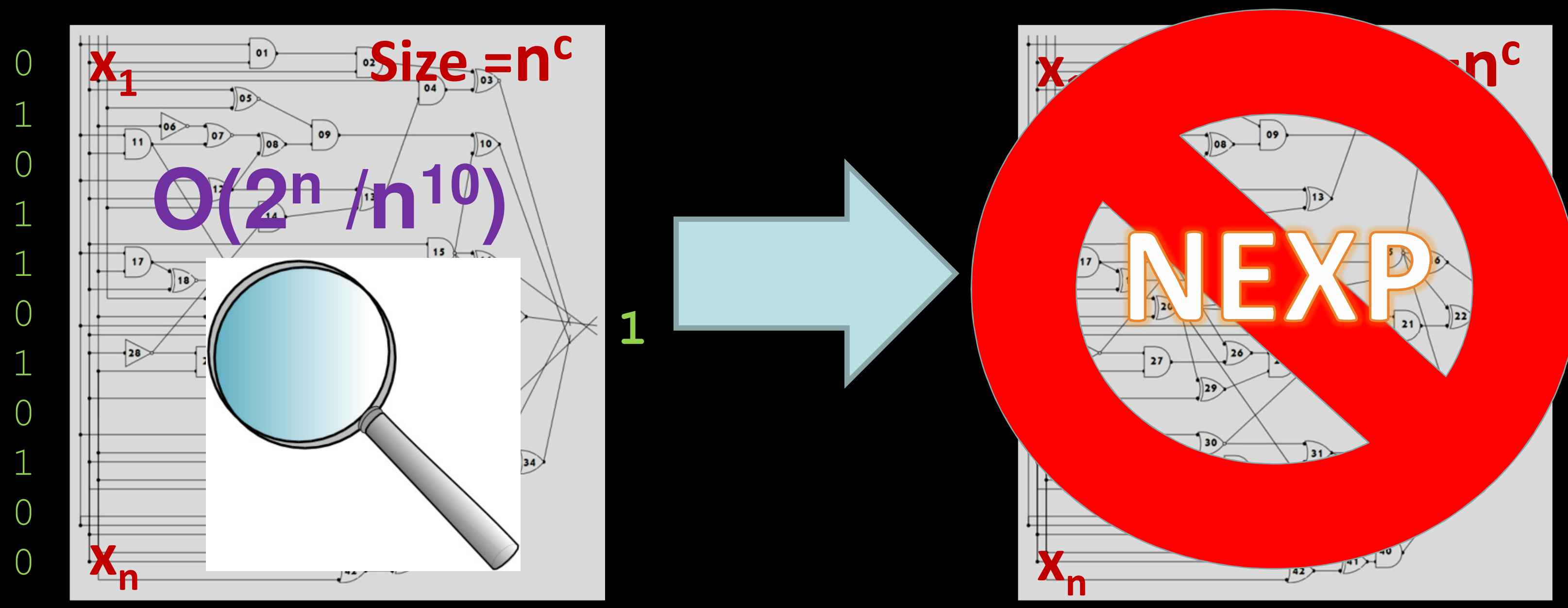
Algorithms and Lower Bounds: Some Basic Connections

Lecture 3: Circuit Complexity and Connections - PART II

Ryan Williams
Stanford University

SAT and Lower Bounds [W'10,'11,'13]

A slightly faster algorithm for $\mathcal{C}$-SAT
$\Rightarrow$ Lower bounds against $\mathcal{C}$ circuits



$\mathrm{NEXP} \not\subset \mathrm{P/poly}$

Faster Algorithms ⟹ Lower Bounds

Faster "Algorithms for Circuits"

An algorithm for:

- **Circuit SAT** in $O(2^n/n^{10})$ (n inputs and n^k gates)
- **Formula SAT** in $O(2^n/n^{10})$
- **ACC SAT** in $O(2^n/n^{10})$
- Given a circuit C that's either *UNSAT*, or has $\geq$ *2^{n-1} satisfying assignments*, determine which, in $O(2^n/n^{10})$ time **(A Promise-BPP problem)**

No "Circuits for Algorithms"

Would imply:

- **NEXP $\not\subset$ P/poly**
- **NEXP $\not\subset$ (non-uniform) NC^1**
- **NEXP $\not\subset$ ACC**

NEXP $\not\subset$ P/poly

Converse: Can interesting circuit *lower bounds* tell us something about circuit-analysis algorithms?

Many well-known connections between

circuit lower bounds and *derandomization*

e.g. EXP $\not\subset$ P/poly $\Rightarrow$ BPP is in SUBEXP

For *restricted* circuits, sometimes the techniques used to prove *circuit lower bounds* can be used to derive faster *SAT algorithms*

Example: Boolean formulas over AND, OR, NOT, fan-in 2

[Subbotovskaya '61] MOD2 on n bits cannot be computed with $n^{1.4999}$ size Boolean formulas with AND, OR, NOT gates

[Santhanam'11] Satisfiabiity of O(n)-size Boolean formulas with AND and OR gates can be solved in $o(2^n)$ time

Converse: Can interesting circuit *lower bounds* tell us something about circuit-analysis algorithms?

For *restricted* circuits, sometimes the techniques used to prove *circuit lower bounds* can be used to derive faster *SAT algorithms*
[CKKSZ14] "Mining circuit lower bound proofs for meta-algs"

Can "mine circuit lower bound proofs" for other algorithms!
[W'14] [AWY'15], [AW'15] applied the polynomial method of R-S to yield faster algorithms for many problems:

- **Solve all-pairs shortest paths in $\frac{n^3}{2^{\sqrt{\log n}}}$ time**
- **Find a disjoint pair of sets among a set system**
- **Compute partial match queries in batch**
- **Evaluate a CNF formula on many chosen assignments**
- **Find a longest common substring with don't cares**
- **Solve 0-1 Integer LP faster than 2^n time**
- **Find a closest pair of points in the Hamming metric**

Are interesting circuit *lower bounds* *equivalent* to interesting circuit-analysis algorithms?

[Impagliazzo-Kabanets-Wigderson'02]
There are "non-trivial" CAPP algorithms
IF AND ONLY IF
NEXP is not in P/poly

What does non-trivial mean?
We call a nondeterministic algorithm **A** **"non-trivial for CAPP"** if:

- **For every ε, A**(C) runs in $2^{n^{\varepsilon}}$ time on circuits C of size n and uses n^{ε} bits of advice
- For **infinitely many n**, there's ≥ 1 accepting computation path on all C of size n, and every accepting path outputs a value v within 1/10 of the acceptance probability of C

Are interesting circuit *lower bounds* *equivalent* to interesting circuit-analysis algorithms?

[W '13]
There are "non-trivial" algorithms for MCSP
IF AND ONLY IF
NEXP is not in P/poly

What does non-trivial mean?

We call an algorithm A **"non-trivial for MCSP"** if for all *k*,

- A(*f*) runs in poly(2^n) time on any Boolean function *f* of 2^n bits and uses *n* bits of advice
- For **infinitely many n**, there is a Boolean function *f* of 2^n bits such that A(*f*) outputs 1, and for all *f* computable with an n^k size circuit, A(*f*) outputs 0

Contrast with Natural Proofs!

Lower Bounds as Data Design

[CW'15] **New equivalence between algs + complexity**

Let $f : \{0,1\}^* \to \{0,1\}$ be a desired function.

Let $\mathcal{C}$ be some "simple" class of Boolean circuits.

Define $\mathcal{C}$-Test For *f* to be the problem:

Input: A circuit C from $\mathcal{C}$; n(C) = number of inputs to C

Decide: Does C compute *f* restricted to $\{0,1\}^{n(C)}$?

This is a very well-motivated problem from practice!

We have a specification *f* and want to verify if C meets it

Lower Bounds as Data Design

Define $\mathcal{C}$-Test For *f* to be the problem:

Input: A circuit C from $\mathcal{C}$; let n(C) = number of inputs to C

Decide: Does C compute *f* restricted to $\{0,1\}^{n(C)}$?

We gauge the complexity of **$\mathcal{C}$-Test For *f*** by ***measuring the number of inputs needed to test if a given circuit computes f:***

The *data complexity of the* $\mathcal{C}$-Test For *f* is a function T : $\mathbb{N} \to \mathbb{N}$

T(*s*) := the minimum number of labeled examples (*x*, *f*(*x*)) necessary and sufficient to determine for all C $\in \mathcal{C}$ of size *s* whether C computes *f* on all n(C)-bit inputs

This is also well-motivated! Small data complexity means we can rapidly determine if C computes *f*

Lower Bounds as Data Design

Theorem: For every function $f : \{0,1\}^* \rightarrow \{0,1\}$,
Lower Bounds on the data complexity of $\mathcal{C}$-Test For f

are equivalent to

Upper Bounds on the $\mathcal{C}$ circuit complexity of f

Theorem: For every function $f : \{0,1\}^* \rightarrow \{0,1\}$,
Upper Bounds on the data complexity of $\mathcal{C}$-Test For f

are equivalent to

Lower Bounds on the $\mathcal{C}$ circuit complexity of f

These "duals" provide an "alternate universe" where inputs become the "computational model" and circuits become the "inputs"

Lower Bounds as Data Design

For example, the following are equivalent:

1) **NP $\not\subset$ P/poly** (resp. **NP $\not\subset$ i.o. P/poly**)
2) For every $\varepsilon > 0$ and for infinitely many s (resp. for every s), the data complexity of testing size-s circuits for SAT is at most $\boldsymbol{O\left(2^{s^{\varepsilon}}\right)}$

OPEN: Can we use data complexity to recover new proofs of old circuit lower bounds?

Lower Bounds as Data Design

Let $f: \{0,1\}^* \to \{0,1\}$, and let $S(n) \geq 2n$ for all n.

"Data Complexity of Testing Size-s Circuits for f"

= Min number of inputs needed to distinguish:

- **circuits of size s computing a slice of f**
- **circuits of size s that don't.**

Thm: If $\boldsymbol{f} \in \mathbf{SIZE}\left(\boldsymbol{S(n)}\right)$, then the data complexity of testing size-s circuits for f is $\mathbf{2}^{\boldsymbol{\Omega}\left(\boldsymbol{S^{-1}(s)}\right)}$, a.e.

Thm: If $\boldsymbol{f} \notin \mathbf{SIZE}\left(\mathbf{2}\boldsymbol{n} \cdot \boldsymbol{S(n)}\right)$, then the data complexity of testing size-s circuits for f is at most
$\boldsymbol{O}\left(\mathbf{2}^{\boldsymbol{S^{-1}(s)}} + \boldsymbol{S^{-1}(s)} \cdot \boldsymbol{s^2} \log \boldsymbol{s}\right)$ i.o.

Ideas Behind The Proofs

Thm: If $\boldsymbol{f} \in \textbf{SIZE}\big(\boldsymbol{S(n)}\big)$, then the data complexity of testing size-s circuits for f is $\mathbf{2}^{\boldsymbol{\Omega}\left(\boldsymbol{S^{-1}(s)}\right)}$, a.e.

- $\boldsymbol{f} \in \textbf{SIZE}\big(\boldsymbol{S(n)}\big)$ implies that for *every* n-bit input x, there is a circuit of size $S(n) + n$ which disagrees with f *only* at x.
- It follows that every test set for $\boldsymbol{f}$ on circuits of size $S(n) + n$ has cardinality at least 2^n.

Thm: If $\boldsymbol{f} \notin \textbf{SIZE}\big(\boldsymbol{2n \cdot S(n)}\big)$, then the data complexity of testing size-s circuits for f is at most $\boldsymbol{O}\left(\mathbf{2}^{\boldsymbol{S^{-1}(s)}} + \boldsymbol{S^{-1}(s) \cdot s^2 \log s}\right)$ i.o.

Use "small counterexample" sets: can get an O(S(n) log S(n)) size test set for all circuits of size S(n) with n inputs.

For size-s circuits where n is "too large" to compute f, we have small test sets. For size-s circuits with n "small enough", it becomes possible to compute f within size s.

General Questions To Think More About

How can **algorithms** help prove **lower bounds**?

How can properties of circuits be turned into algorithms for analyzing them?

How can **lower bounds** help design **algorithms**?

- We can make progress on both algorithms and lower bounds by studying them as a ***unit***
- ***Next, an explicit example of algorithms proving lower bounds: NEXP vs ACC***

Definition: ACC Circuits

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

- Every C_n takes n bits of input and outputs a 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)

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

Remarks

1. The default size 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***