Algorithms as Lower Bounds

Lecture 2: Circuits for Algorithms, Connections, NEXP vs ACC

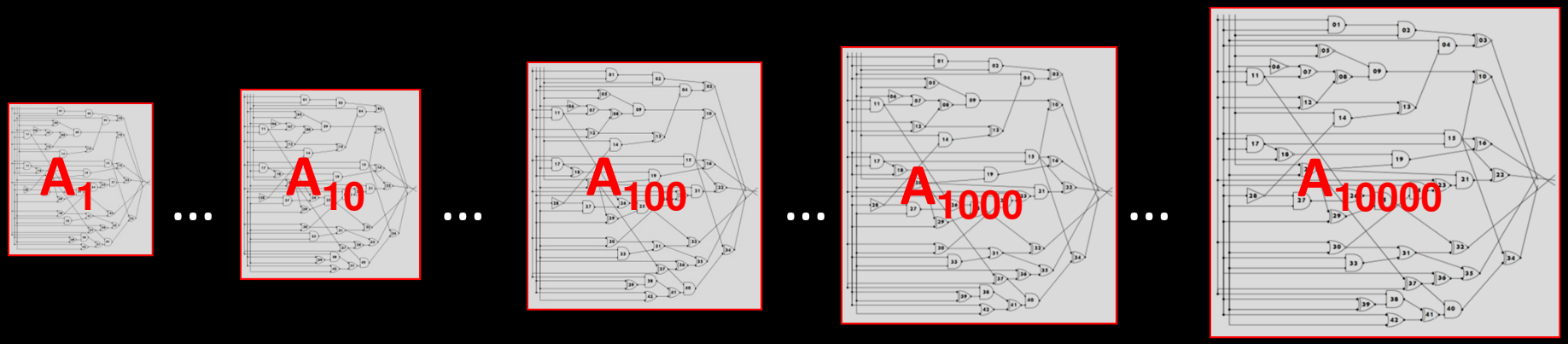
Ryan Williams
Stanford University

Outline

- **Circuit Analysis Algorithms (Last Time)**
- **Circuit Complexity (Today)**
- **Connections**
- **NEXP vs ACC**

Circuit Complexity of Infinite Languages

Allow a distinct logical circuit $\mathbf{A}_n$ to run on inputs of length n



P/poly = Class of problems solvable with a **circuit family** $\{\mathbf{A}_n\}$ **such that** $(\exists k \geq 1)(\forall n)$, the ***size* of $\mathbf{A}_n$ is at most n^k**

This is an ***infinite computational model***

$\{1^n$ | the nth Turing machine halts on blank tape$\}$ $\in$ P/poly

The usual techniques of computability theory are essentially powerless for understanding **P/poly**

Circuits for Algorithms

P/poly = Problems solvable with a **circuit family** $\{A_n\}$ where the ***number of gates*** **of** $A_n \leq n^k$

Most Boolean functions require huge circuits!

Theorem [Shannon '49] W.h.p., a randomly chosen function $f : \{0,1\}^n \to \{0,1\}$ requires a circuit of size at least $2^n/n$

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)?

The key obstacle: Non-uniformity can be **very** powerful!

Circuits for Algorithms

P/poly = Problems solvable with a **circuit family** $\{A_n\}$ where the ***number of gates*** **of** $A_n \leq n^k$

Most Boolean functions require huge circuits!

Theorem [Shannon '49] W.h.p., a randomly chosen function $f : \{0,1\}^n \to \{0,1\}$ requires a circuit of size at least $2^n/n$

What "uniform" algorithms can be simulated in P/poly?

OPEN PROBLEM: Is **NEXP** $\subset$ **P/poly?**

Can all problems with ***exponentially long*** solutions be solved with ***polynomial size*** circuit families?

Given "infinite" preprocessing time, can one construct small-size circuits solving NEXP problems?

Circuits for Algorithms

P/poly = Problems solvable with a **circuit family** $\{A_n\}$ where the ***number of gates*** **of** $A_n \leq n^k$

What "uniform" algorithms can be simulated in P/poly?

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

In other words, the SAT problem cannot be in P/poly

The proof of a theorem is the first step to concrete numerical tradeoffs between *sizes of inputs* and *sizes of computations*.

Circuits for Algorithms

P/poly = Problems solvable with a **circuit family** $\{A_n\}$ where the ***number of gates*** **of** $A_n \leq n^k$

What "uniform" algorithms can be simulated in P/poly?

Kolmogorov's Hypothesis:
P has O(n)-size circuits

This would be remarkable…

In fact, if this could be proved true,
then a proof of P ≠ NP would follow!
(If P=NP then P does not have O(n)-size circuits.)

Circuits for Algorithms

The "circuits for algorithms" questions have interesting consequences, regardless of how they're resolved.

[Karp-Lipton-Meyer '80] **EXP $\subset$ P/poly $\Rightarrow$ P $\neq$ NP**

Folklore Theorem

If every problem in $2^{O(n)}$ time has circuits *smaller* than 1.99^n size for infinitely many input lengths, then **P $\neq$ NP**

[BFNW '90] **EXP $\not\subset$ P/poly $\Rightarrow$ Pseudorandom generators**

Theorem [Impagliazzo-Wigderson '97]

If *some* problem in $2^{O(n)}$ time needs circuits *larger* than 1.99^n for almost all input lengths, then **P = BPP**

Theorem [IKW '01] **NEXP $\not\subset$ P/poly $\Rightarrow$**

Can simulate MA in NSUBEXP

Outline

- **Circuit Analysis Algorithms (Last Time)**
- **Circuit Complexity (Today)**
- **Connections**
- **NEXP vs ACC**

Connections

Algorithms for Circuits (Circuit Analysis):

Designing faster circuit-analysis algorithms

Circuits for Algorithms (Circuit Complexity):

Designing small circuits to simulate complex algorithms

Can we use one of these tasks to inform the other task?

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

[Karp-Lipton-Meyer '80]
Suppose we had extremely efficient circuit-analysis algorithms.
Then we could prove that there are problems
solvable by an algorithm in 2^n time that are not in P/poly

P = NP ⇒ There are problems in EXP
(Circuit SAT in P) which are not in P/poly
(Circuit Minimization in P)

This is an interesting conditional statement, but it has limited utility, since we do not believe the hypothesis is true!

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

[Kabanets-Cai '00]

Studied consequences of MCSP in P

Given: Truth table of a Boolean function ***f*,** parameter **s**

Question: Does *f* have a circuit of size at most s?

If MCSP is in P, then

1. EXP^{NP} requires maximum circuit complexity
   (new circuit lower bounds)
2. BPP = ZPP
3. Discrete Log, Factoring, Graph Iso **[AD'14]** are in BPP
4. No strong pseudorandom functions (or PRGs)

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

The Natural Proofs Barrier [Razborov-Rudich '94]

Suppose while proving a circuit lower bound,
you construct a polytime algorithm that can:
*distinguish **many** functions **not** computable with the circuits*
*from all "easy" functions that **are** computable with the circuits*
(MCSP is "kind of" in P)
Then these circuits are too weak to support pseudorandom fns.

If we *believe* it's possible to prove lower bounds which are strong enough for crypto, then we must also believe that "natural proofs" cannot establish results like P ≠ NP

Unfortunately, most known arguments for strong circuit lower bounds can be "naturalized"

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

[Kabanets-Impagliazzo '04] Arithmetic complexity

Arithmetic formulae: **Analogous to Boolean formulae, except operations are + and * over $\mathbb{Z}$ instead of OR and AND over {0,1}**

Polynomial Identity Testing (PIT): Given two arithmetic formulas F and G, do F and G represent the *same* polynomial?

Examples: $(x + y)^2 = x^2 + y^2 + 2xy$

$(x^2 + a^2)\cdot(y^2 + b^2) = (x\cdot y - a\cdot b)^2 + (x\cdot b + a\cdot y)^2$

There are efficient ***randomized*** algorithms for PIT, but no efficient **deterministic** algorithms are known

Can interesting circuit-analysis algorithms tell us something about the *limitations* of circuits?

[Kabanets-Impagliazzo '04] **Arithmetic complexity**

Polynomial Identity Testing (PIT): Given two arithmetic formulas F and G, do F and G represent the *same* polynomial?

Theorem [KI'04]

Deterministic efficient algorithms for Polynomial Identity Testing

⇒ **Arithmetic Formula Size Lower Bounds!**

(NEXP not in P/poly, or the Permanent does not have arithmetic formulas of polynomial size)

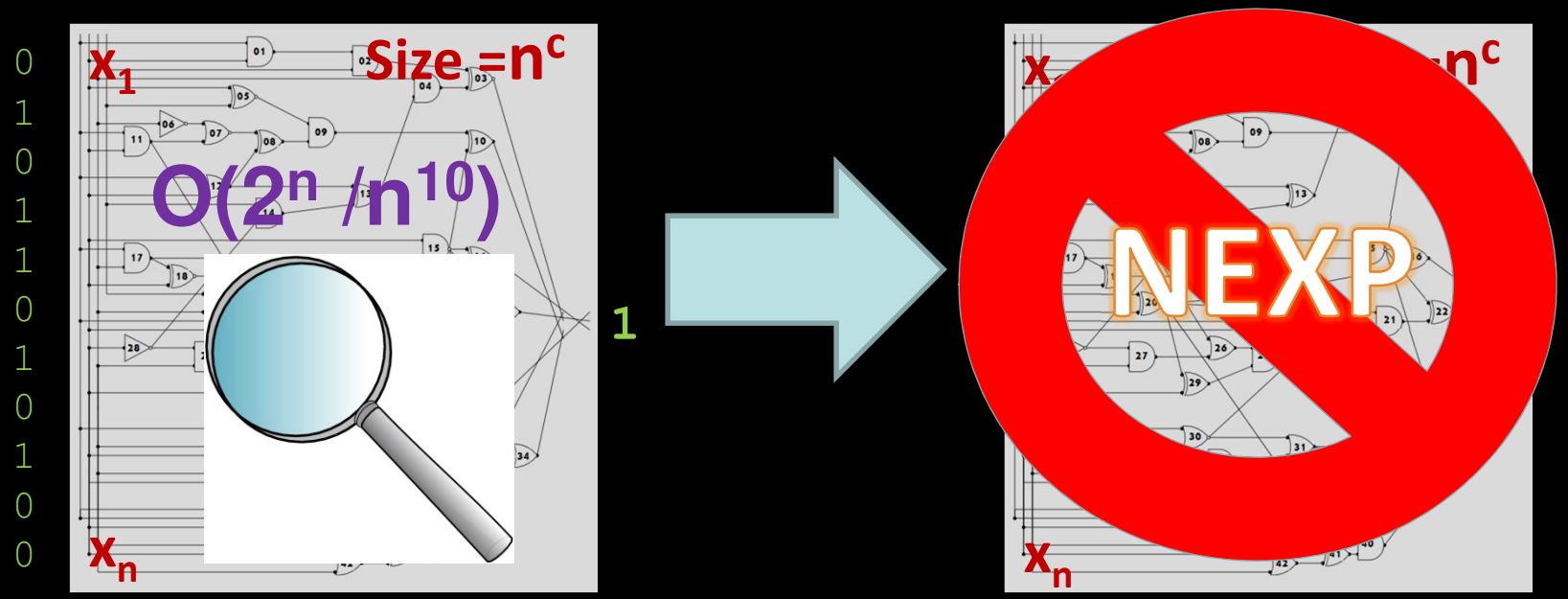
Efficient algorithms for analyzing arithmetic formulas

imply

limits on representing explicit polynomials with small formulas!

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 $\Rightarrow$ 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*

Can "mine circuit lower bound proofs" for other algorithms!

[W'14] [AWY'15] applied the polynomial method of R-S to:

- 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 assignments of one's choice
- Find a longest common substring with don't cares
- Solve 0-1 Integer LP faster than 2^n time

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 a Boolean function *f* of 2^n bits
- For **infinitely many n,**
  - There is an *f* on n bits such that A(*f*) outputs 1
  - For all sufficiently large *f* computable with an n^k size circuit, A(*f*) outputs 0

Questions

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

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

- Make progress on both algorithms and lower bounds by studying both algorithms and complexity as a ***unit***
- ***Next, an explicit example: NEXP vs ACC***

Outline

- **Circuit Analysis Algorithms (Last Time)**
- **Circuit Complexity (Today)**
- **Connections**
- **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)

$MOD_m(x_1,\ldots,x_t) = 1$ 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***

Where does ACC come from?

Prove **P** ≠ **NP** by proving **NP** ⊄ **P/poly**.
The simple combinatorial nature of circuits should make it easier to prove impossibility results.

Ajtai, Furst-Saxe-Sipser, Håstad (early 80's)
MOD2 ∉ **AC0** [i.e., $n^{O(1)}$ size **ACC** with ***only*** AND, OR, NOT]

Razborov, Smolensky (late 80's)
MOD3 ∉ (**AC0** with **MOD2** gates)
For p ≠ q prime, **MODp** ∉ (**AC0** with **MODq** gates)

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

Conjecture Majority ∉ **ACC**
Conjecture (early 90's) **NP** ⊄ **ACC**
Conjecture (late 90's) **NEXP** ⊄ **ACC**

ACC Lower Bounds

EXP^{NP} = Exponential Time with an NP oracle
NEXP = Nondeterministic Exponential Time

Theorem 1 There is a problem **Q** in **EXP^{NP}** such that for every *d*, m there is an $\varepsilon > 0$ such that **Q** does not have ACC circuits with MODm gates, depth *d,* and size **$2^{n^{\varepsilon}}$**

Theorem 2 There is a problem **Q** in **NEXP** such that **Q** does not have **$n^{poly(\log n)}$** size ACC circuits of any constant depth

Remark Compare with the following:

[MS 70's] **$EXP^{(NP^{NP})}$** doesn't have **$o(2^n/n)$** size circuits

[K82] **$NEXP^{NP} \not\subset SIZE(n^{poly(\log n)})$**

[BFT'98] **$MA\text{-}EXP \not\subset SIZE(n^{poly(\log n)})$**

How do we get started?

Find a **nice property** of **ACC** that **NEXP** doesn't possess.
Turn this into a proof.

NTIME[t(n)] = Class of problems solvable by nondeterministic algorithms running in t(n) time

Nondeterministic Time Hierarchy [SFM '78]
For functions **t**, **T** such that **t(n+1) ≤ o(T(n))**,

$$\text{NTIME}[t(n)] \subsetneq \text{NTIME}[T(n)]$$

Corollary There are **NTIME[2^n]** problems that aren't solved by nondeterminstic algorithms in **O(2^n/n)** time.

Idea: Try to show that, if NEXP were in ACC, then this corollary can be contradicted

Nice Property: "Fast" Satisfiability

Let C be a class of Boolean circuits

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

We will look at ACC-SAT.

Proof Strategy for ACC Lower Bounds

1. **Show that faster ACC-SAT algorithms imply lower bounds against ACC**

Theorem **(Example)**

If **ACC-SAT** with n inputs and $2^{n^{o(1)}}$ size is in **$O(2^n/n^{10})$ time** (for all constant depths and moduli), then **EXP^{NP}** doesn't have $2^{n^{o(1)}}$ size ACC circuits.

2. **Design faster ACC-SAT algorithms!**

Theorem For all **d, m** there's an **$\varepsilon > 0$** such that

ACC-SAT on circuits with n inputs, depth **d**, **MODm** gates, and $2^{n^{\varepsilon}}$ size can be solved in **$2^{n - \Omega(n^{\varepsilon})}$ time**

Proof Idea: Assume

- **NEXP has polynomial size circuits**
- **Circuit-SAT with n inputs and $n^{O(1)}$ size is in $O(2^n/n^{10})$ time**

Karp-Lipton, Meyer '80: **P = NP $\Rightarrow$ EXP $\not\subset$ P/poly**

Assume P = NP and EXP $\subset$P/poly

EXP $\subset$P/poly $\Rightarrow\exists$ polysize circuits C encoding accepting computation tableaus:

For every exptime machine M and every string x,
C(M,x,i,j) prints the content of the jth cell of M(x) in step i

The behavior of M(x) can be simulated as follows:

($\exists$ C)($\forall$ i, j) [C makes consistent claims of cells j, j+1, j+2 in steps i, i+1]

This part is computable in coNP

P = NP $\Rightarrow$ ($\exists$ C)R(x,C), where R(x,C) is a poly-time computable predicate

This is an NP problem

P = NP $\Rightarrow$ M(x) is in P. But then we contradict the time hierarchy!

Proof Idea: Assume

- **NEXP has polynomial size circuits**
- **Circuit-SAT with n inputs and $n^{O(1)}$ size is in $O(2^n/n^{10})$ time**

Impagliazzo-Kabanets-Wigderson '01:

NEXP $\subset$ P/poly $\Rightarrow\exists$ circuits C encoding accepting tableaus:

For every ***nondeterministic*** 2^n time machine M and every string x, C(M,x,i,j) prints the jth cell of M(x) in step i, *for some accepting path*

The behavior of M(x) can be simulated as follows:

($\exists$ C)($\forall$ i, j) [C makes consistent claims of cells j, j+1, j+2 in steps i, i+1]

Express this part as a Circuit-SAT instance with n variables??

$\Rightarrow$(

$\Rightarrow$M

The major difficulty:
The number of inputs to the Circuit-SAT instance would be $\approx 2n$

But then NTIME[2^n] $\subseteq$ NTIME[$2^n/n^{10}$],
contradicting the *nondeterministic* time hierarchy!

Detailed Proof

Theorem If ACC-SAT on circuits with n inputs and $2^{n^{o(1)}}$ size is in $O(2^n/n^{10})$ time, then
EXP^{NP} doesn't have $2^{n^{o(1)}}$ size ACC circuits.

Proof Idea Show that if both:

- **ACC-SAT with n inputs and $2^{n^{o(1)}}$ size is in $O(2^n/n^{10})$ time**
- **EXP^{NP} has $2^{n^{o(1)}}$ size ACC circuits**

then **$NTIME[2^n] \subseteq NTIME[o(2^n)]$** **(a contradiction)**

Work with a "compressed" version of the 3SAT problem:

Exponentially long formulas are encoded with polynomial-size circuits