# Algorithms and Lower Bounds: Some Basic Connections

## Lecture 2: Circuit Complexity and Connections

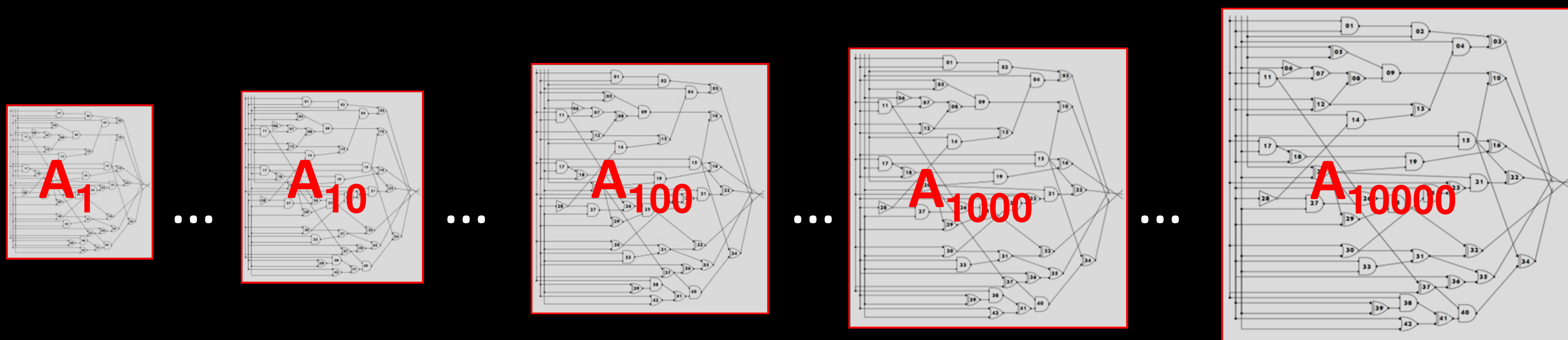
Ryan Williams
Stanford University

# Outline

- Circuit Analysis Algorithms (Last Time)
  "Algorithms for Circuits"
- Circuit Complexity (Today)
- Connections
- NEXP not in ACC

# Circuit Complexity of Infinite Languages

Allow a distinct logical circuit A$_n$ to run on inputs of length $n$




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

In this model, programs have ***infinite-length descriptions***
**$\{1^n$ | the $n$th Turing machine halts on blank tape$\}$ ∈ 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 this would be a 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 $\neq$ 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 $\mathbf{2^{O(n)}}$ 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 $\mathbf{2^{O(n)}}$ 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)**
  **"Algorithms for Circuits"**
- **Circuit Complexity (Today)**
- **Connections**
- **NEXP not in 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**
**Input:** 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. $\mathrm{EXP}^{\mathrm{NP}}$ 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 $\neq$ 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)***

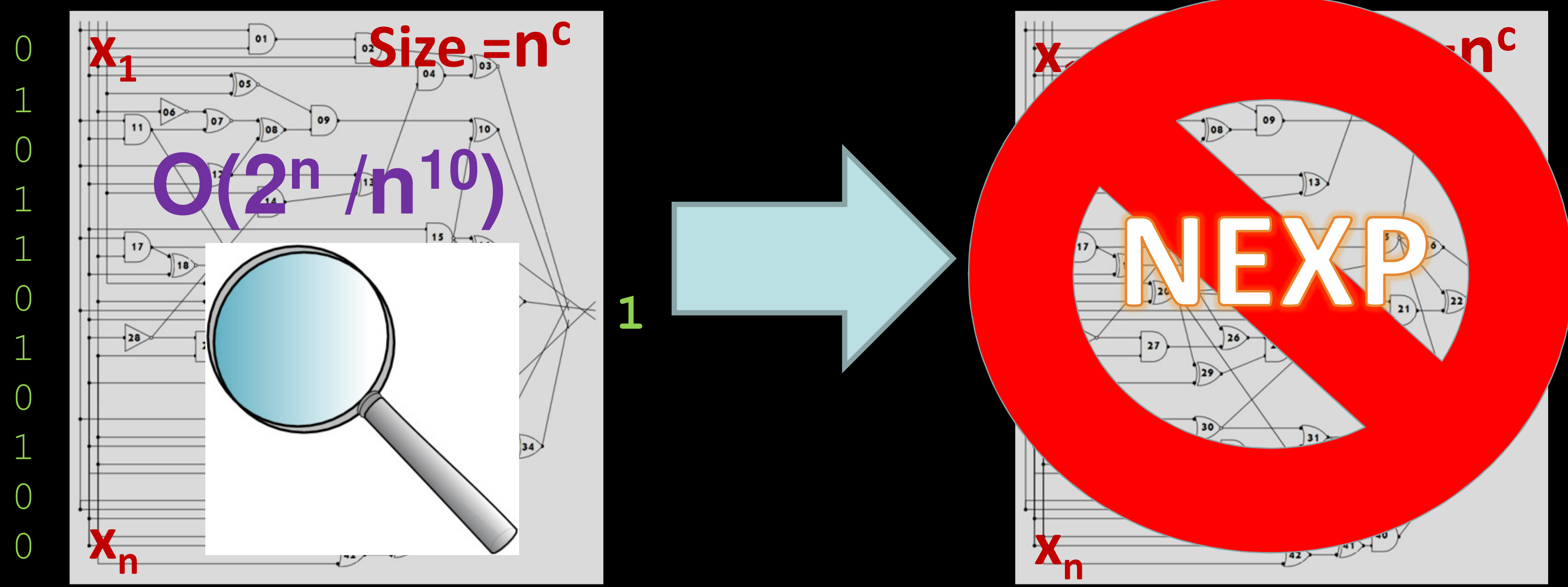
**Efficient algorithms for analyzing arithmetic formulas**

***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




# $\text{NEXP} \not\subset \text{P/poly}$

# Faster Algorithms $\Longrightarrow$ 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**