Algorithms as Lower Bounds (and vice-versa)
Lecture 1: Introduction

Ryan Williams
Stanford University

A view of algorithms and complexity – from 50,000 ft

- 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**

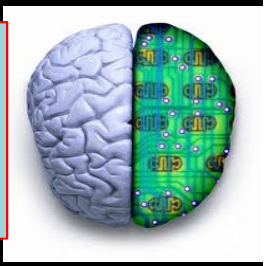
This belief is strongly reflected in the literature

What are lower bounds good for?

Algorithm Engineering

Machine Learning

Lower bound techniques have been



???

We have *no idea* what the proofs of *strong* lower bounds look like!

Pseudorandom Generators

Lower bounds can “remove randomness” from algorithms

Lower bounds are one of the great scientific mysteries of our time!

Why are lower bounds hard to prove?

There are *many* known "no-go" theorems

- **Relativization [70's]**
- **Natural Proofs [90's]**
- **Algebrization [00's]**

The common proof techniques are simply not good enough to prove even **weak** lower bounds!

Great pessimism in complexity theory

"Duality" Between Algorithms and Lower Bounds

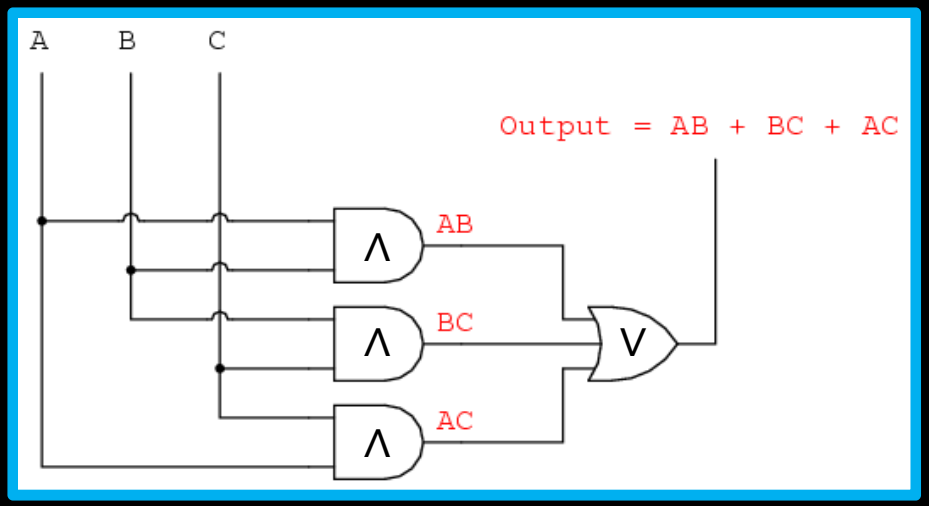
Thesis: Algorithm design is *at least as hard as* proving lower bounds.

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"

Logical Circuits



A logical circuit takes 0-1 inputs and outputs a single bit

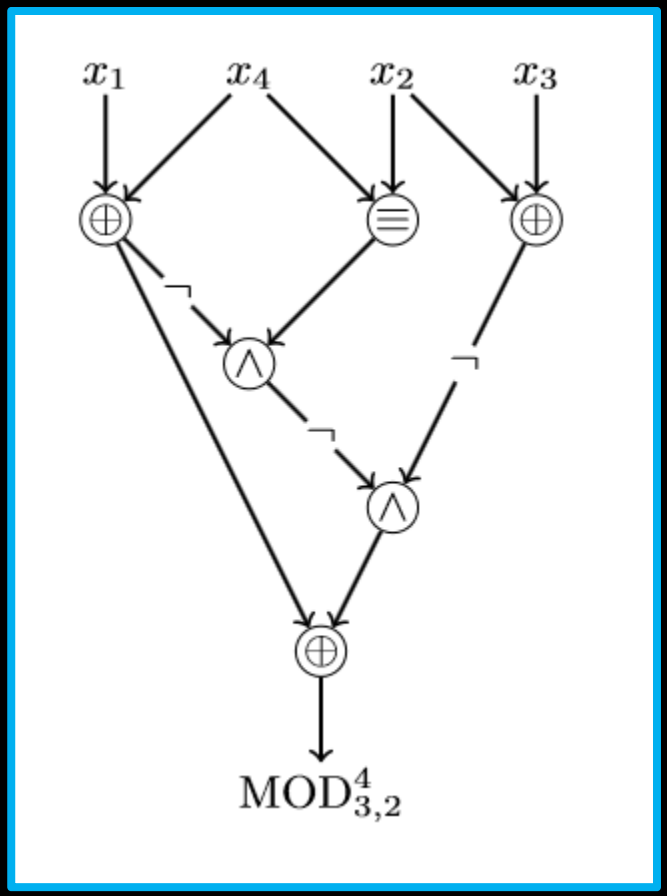
Along the way, it repeatedly takes some previously-computed bits and computes a new function of these bits.

Example: Above circuit outputs 1 ⇔ at least two inputs are 1

Size or *complexity* of the circuit is 4,
fan-in of ANDs is 2, fan-in of the OR is 3

Circuits are a natural model for computing *finite functions*

Logical Circuits



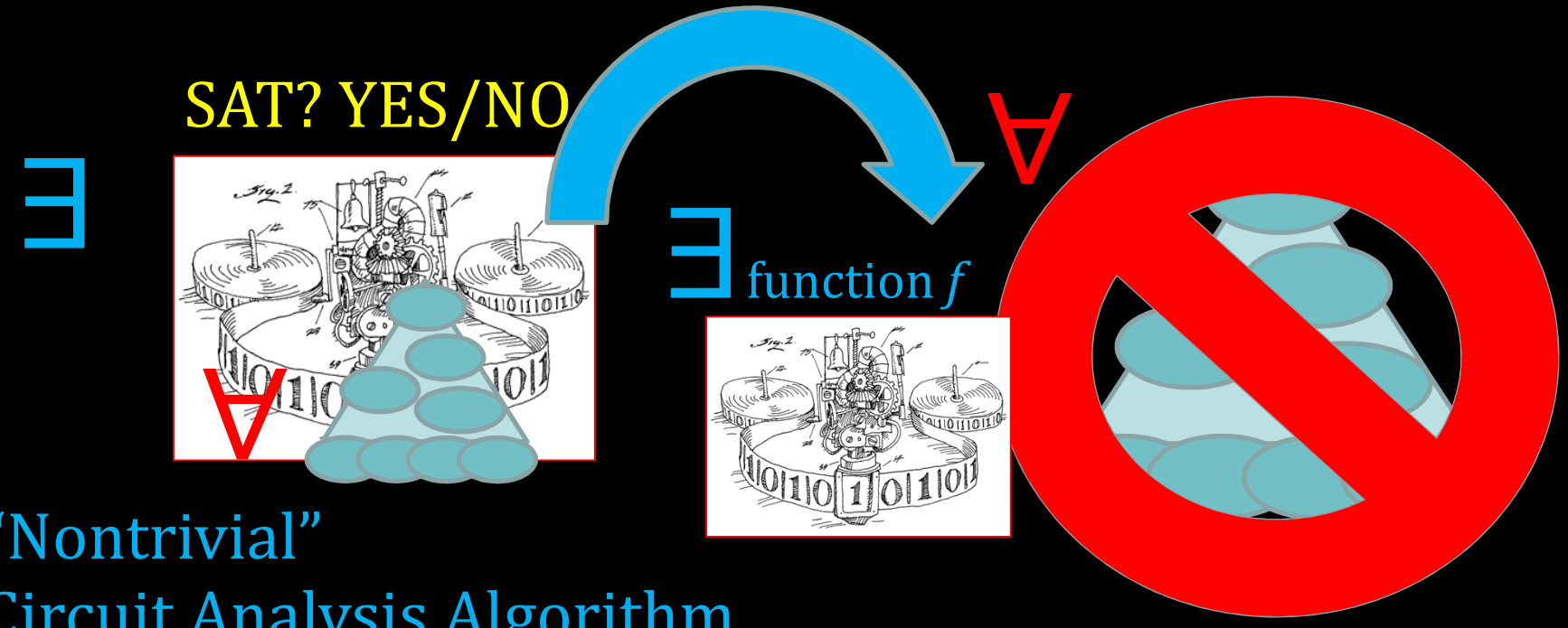
Outputs 1 if and only if $x_1 + x_2 + x_3 + x_4 = 2 \; mod \; 3$

[Kojevnikov, Kulikov, Yaroslavtsev '09]

Above is smallest circuit over all Boolean functions of fan-in 2.

“Duality” Between Circuit Analysis Algorithms and Circuit Lower Bounds

Thesis: Algorithm design is *at least as hard as* proving lower bounds.

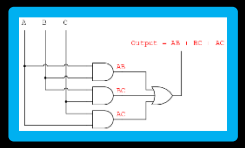


Outline

- **Circuit Analysis (Algorithms)**
- **Circuit Complexity (Lower Bounds)**
- **Connections**

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 $K(x_1,\ldots,x_n) \in \mathcal{C}$, is there an assignment $(a_1, \ldots, a_n) \in \{0,1\}^n$ such that $K(a_1,\ldots,a_n) = 1$?

$\mathcal{C}$-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**

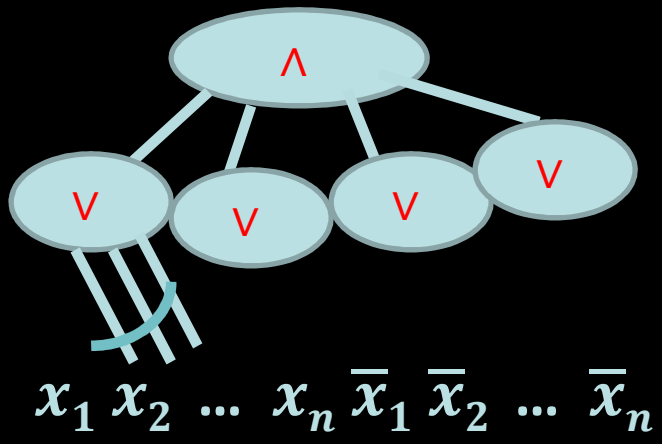
Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

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

  $2^{n - n/(ck)}$ 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

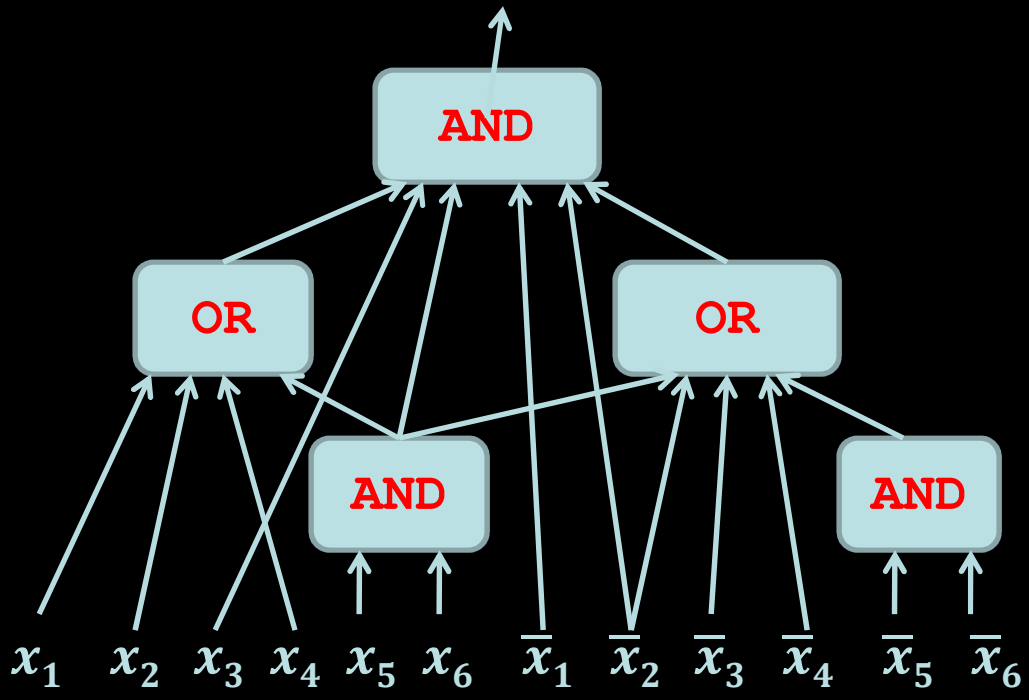
ETH: $\exists \delta > 0$ s.t. 3-SAT requires $2^{\delta n}$ time

Circuit SAT Algorithms

For simple enough circuits, we know of faster algorithms

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

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

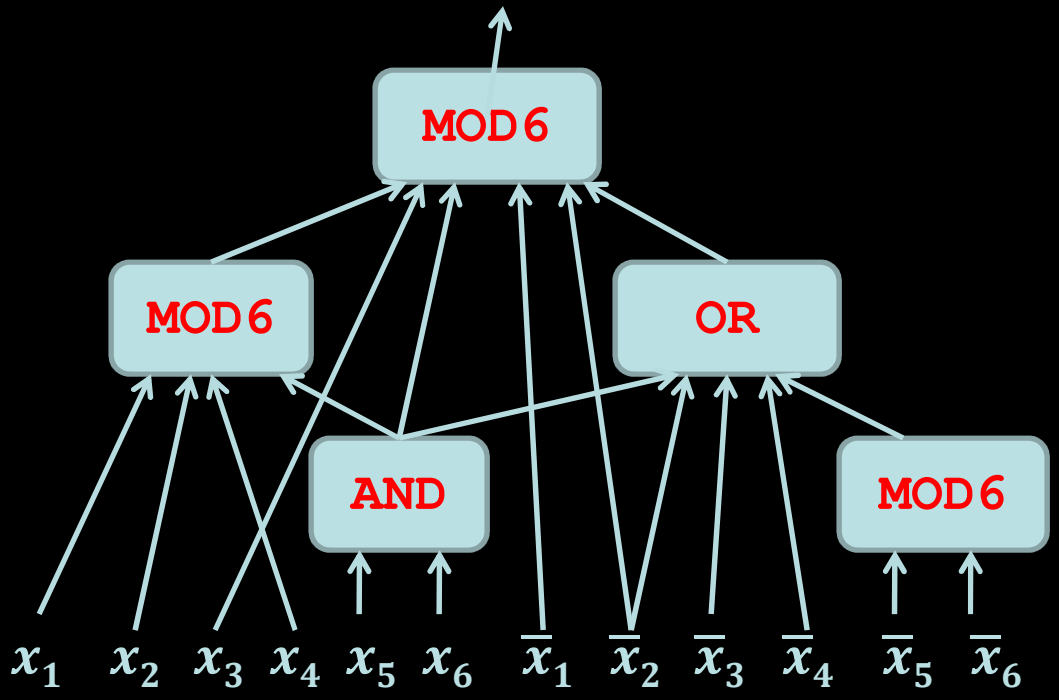


Circuit SAT Algorithms

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 e < 1 depending on m and d

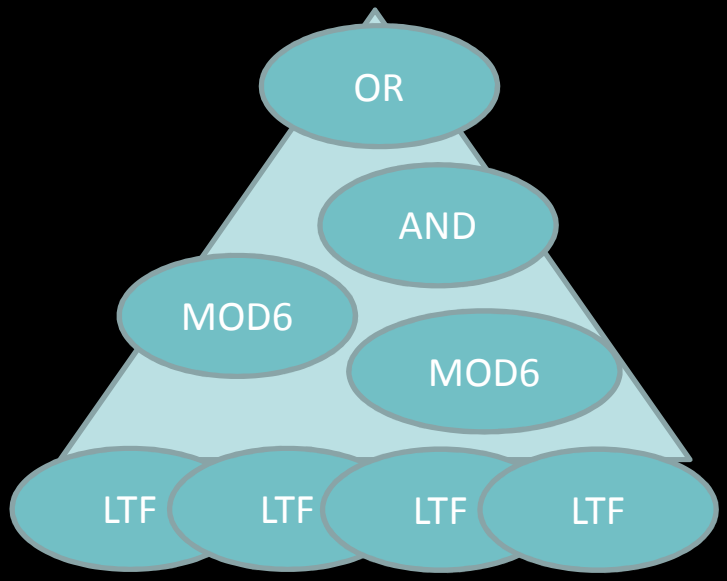


Circuit SAT Algorithms

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 $2^{n^{o(1)}}$



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

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 $\mathbf{K}(x_1,\ldots,x_n) \in \mathcal{C}$, output v such that

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

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?

(Observation: 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]

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 $\mathbf{RP}^{\text{Circuit Min}}$

Open problems:

Find interesting problems in $\mathbf{P}^{\text{Circuit Min}}$

In $\mathbf{P}^{\text{Circuit Min}}$ can we *produce* a min-size circuit, given a truth table?

Exponential Time Algorithms

The topic of “Algorithms for Circuits” constitutes one small facet of the growing area of

Exact algorithms for NP-hard problems

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