CS151
Complexity Theory

Lecture 5

April 13, 2004

Introduction

Power from an unexpected source?

- we know **P ≠ EXP**, which implies no poly-time *algorithm* for Succinct CVAL
- poly-size Boolean *circuits* for Succinct CVAL ??

Introduction

…and the depths of our ignorance:

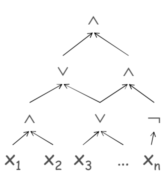
Does **NP** have linear-size, log-depth Boolean circuits ??

Outline

- Boolean circuits and formulae
- uniformity and advice
- the **NC** hierarchy and parallel computation
- the quest for circuit lower bounds
- a lower bound for formulae

Boolean circuits

- **circuit** C
  – directed acyclic graph
  – nodes: AND ($\wedge$); OR ($\vee$); NOT ($\neg$); variables x_i



- C computes function $f:\{0,1\}^n \rightarrow \{0,1\}$ in natural way
  – identify C with function f it computes

Boolean circuits

- **size** = # gates
- **depth** = longest path from input to output
- **formula (or expression)**: graph is a tree

- every function $f:\{0,1\}^n \rightarrow \{0,1\}$ computable by a circuit of size at most $O(n2^n)$
  – AND of n literals for each x such that f(x) = 1
  – OR of up to 2^n such terms

Circuit families

- circuit works for specific input length
- we're used to $f:\Sigma^* \rightarrow \{0,1\}$
- circuit **family** : a circuit for each input length $C_1, C_2, C_3, \ldots$ = "$\{C_n\}$"
- "$\{C_n\}$ computes f" iff for all x

$$C_{|x|}(x) = f(x)$$

- "$\{C_n\}$ decides L", where L is the language associated with f

Connection to TMs

- TM M running in time t(n) decides language L
- can build circuit family $\{C_n\}$ that decides L
  - size of $C_n = O(t(n)^2)$
  - Proof: CVAL construction
- Conclude: $L \in$ **P** implies family of polynomial-size circuits that decides L

Connection to TMs

- other direction?
- A poly-size circuit family:
  - $C_n = (x_1 \vee \neg x_1)$ if M_n halts
  - $C_n = (x_1 \wedge \neg x_1)$ if M_n loops
- decides (unary version of) HALT!
- oops…

Uniformity

- Strange aspect of circuit family:
  - can "encode" (potentially uncomputable) information in family specification
- solution: uniformity – require specification is simple to compute
  - Definition: circuit family $\{C_n\}$ is **logspace uniform** iff TM M outputs C_n on input 1^n and runs in O(log n) space

Uniformity

Theorem: **P** = languages decidable by logspace uniform, polynomial-size circuit families $\{C_n\}$.

- Proof:
  - already saw ($\Rightarrow$)
  - ($\Leftarrow$) on input x, generate $C_{|x|}$, evaluate it and accept iff output = 1

TMs that take advice

- family $\{C_n\}$ without uniformity constraint is called "non-uniform"
- regard "non-uniformity" as a limited resource just like time, space, as follows:
  - add read-only "advice" tape to TM M
  - M "decides L with advice A(n)" iff

    $M(x, A(|x|))$ accepts $\Leftrightarrow x \in L$
  - note: A(n) depends only on |x|

TMs that take advice

- Definition: **TIME(t(n))/f(n)** = the set of those languages L for which:
  – there exists A(n) s.t. $|A(n)| \leq f(n)$
  – TM M decides L with advice A(n)
- most important such class:

$$\mathbf{P/poly} = \cup_k \mathbf{TIME}(n^k)/n^k$$

TMs that take advice

Theorem: L $\in$ **P/poly** iff L decided by family of (non-uniform) polynomial size circuits.

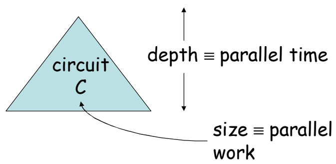
- Proof:
  – ($\Rightarrow$) C_n from CVAL construction; hardwire advice A(n)
  – ($\Leftarrow$) define A(n) = description of C_n; on input x, TM simulates $C_n(x)$

Approach to P/NP

- Believe **NP** $\not\subset$ **P**
  – equivalent: "**NP** does not have uniform, polynomial-size circuits"
- *Even believe* **NP** $\not\subset$ **P/poly**
  – equivalent: "**NP** (or, e.g. SAT) does not have polynomial-size circuits"
  – implies **P** $\neq$ **NP**
  – many believe: best hope for **P** $\neq$ **NP**

Parallelism

- uniform circuits allow refinement of polynomial time:



Parallelism

- the **NC** ("**N**ick's **C**lass") Hierarchy (of logspace uniform circuits):

$$\mathbf{NC}_k = O(\log^k n) \text{ depth, poly(n) size}$$

$$\mathbf{NC} = \cup_k \mathbf{NC}_k$$

- captures "efficiently parallelizable problems"
- not realistic? overly generous
- OK for proving non-parallelizable

Matrix Multiplication

n x n matrix A	n x n matrix B	=	n x n matrix AB

- what is the parallel complexity of this problem?
  – work = poly(n)
  – time = $\log^k(n)$? (which k?)

Matrix Multiplication

- two details
  - arithmetic matrix multiplication…

    $A = (a_{i,\,k})$ $B = (b_{k,\,j})$ $(AB)_{i,j} = \Sigma_k\,(a_{i,k} \times b_{k,\,j})$

  … vs. Boolean matrix multiplication:

    $A = (a_{i,\,k})$ $B = (b_{k,\,j})$ $(AB)_{i,j} = \vee_k\,(a_{i,k} \wedge b_{k,\,j})$

  - single output bit: to make matrix multiplication a language: on input A, B, (i, j) output $(AB)_{i,j}$

April 8, 2004 CS151 Lecture 4 19

Matrix Multiplication

- Boolean Matrix Multiplication is in $\mathbf{NC_1}$
  - level 1: compute n ANDS: $a_{i,k} \wedge b_{k,\,j}$
  - next log n levels: tree of ORS
  - n^2 subtrees for all pairs (i, j)
  - select correct one and output

April 8, 2004 CS151 Lecture 4 20

Boolean formulas and $\mathbf{NC_1}$

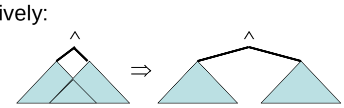
- Previous circuit is actually a formula. This is no accident:

<u>Theorem</u>: $L \in \mathbf{NC_1}$ iff decidable by polynomial-size uniform family of Boolean formulas.

April 8, 2004 CS151 Lecture 4 21

Boolean formulas and $\mathbf{NC_1}$

- Proof:
  - ($\Rightarrow$) convert $\mathbf{NC_1}$ circuit into formula
    - recursively:

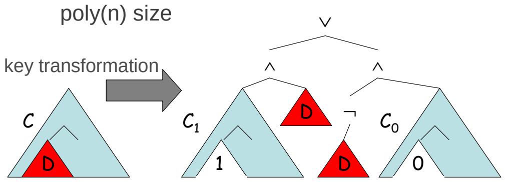


    - note: logspace transformation (stack depth log n, stack record 1 bit – "left" or "right")

April 8, 2004 CS151 Lecture 4 22

Boolean formulas and $\mathbf{NC_1}$

- ($\Leftarrow$) convert formula of size n into formula of depth O(log n)
  - note: size ≤ 2^{depth}, so new formula has poly(n) size



April 8, 2004 CS151 Lecture 4 23

Boolean formulas and $\mathbf{NC_1}$

- D any minimal subtree with size at least n/3
  - implies size(D) ≤ 2n/3
- define T(n) = maximum depth required for any size n formula
- C_1, C_0, D all size ≤ 2n/3

$$T(n) \le T(2n/3) + 3$$

$$\text{implies } T(n) \le O(\log n)$$

April 8, 2004 CS151 Lecture 4 24

Relation to other classes

- Clearly **NC** ⊂ **P**
  - recall **P** ≡ uniform poly-size circuits
- **NC$_1$** ⊂ **L**
  - on input x, compose logspace algorithms for:
    - generating $C_{|x|}$
    - converting to formula
    - FVAL

Relation to other classes

- **NL** ⊂ **NC$_2$**: S-T-CONN ∈ **NC$_2$**
  - given G = (V, E), vertices s, t
  - A = adjacency matrix (with self-loops)
  - $(A^2)_{i,j} = 1$ iff path of length ≤ 2 from node i to node j
  - $(A^n)_{i,j} = 1$ iff path of length ≤ n from node i to node j
  - compute with depth log n tree of Boolean matrix multiplications, output entry s, t
  - $\log^2 n$ depth total

NC vs. **P**

- can every efficient algorithm be efficiently parallelized?

$$\mathbf{NC} \stackrel{?}{=} \mathbf{P}$$

- **P**-complete problems least-likely to be parallelizable
  - if **P**-complete problem is in **NC**, then **P** = **NC**
  - Why?
  - we use logspace reductions to show problem **P**-complete; **L** in **NC**

NC vs. **P**

- can every uniform, poly-size Boolean circuit family be converted into a uniform, poly-size Boolean formula family?

$$\mathbf{NC_1} \stackrel{?}{=} \mathbf{P}$$

Lower bounds

- Recall: "**NP** does not have polynomial-size circuits" (**NP** ⊄ **P/poly**) implies **P** ≠ **NP**
- **major goal**: prove lower bounds on (non-uniform) circuit size for problems in **NP**
  - believe exponential
  - super-polynomial enough for **P** ≠ **NP**
  - best bound known: 4.5n
  - don't even have super-polynomial bounds for problems in **NEXP**

Lower bounds

- lots of work on lower bounds for restricted classes of circuits
  - we'll see two such lower bounds:
    - formulas
    - monotone circuits

Shannon's counting argument

- frustrating fact: *almost all* functions require huge circuits

Theorem (Shannon): With probability at least $1 - o(1)$, a random function

$$f:\{0,1\}^n \to \{0,1\}$$

requires a circuit of size $\Omega(2^n/n)$.

Shannon's counting argument

- Proof (counting):
  - $B(n) = 2^{2^n}$ = # functions $f:\{0,1\}^n \to \{0,1\}$
  - # circuits with n inputs + size s, is at most

$$C(n, s) \le ((n+3)s^2)^s$$

n+3 gate types

2 inputs per gate

s gates

Shannon's counting argument

- $C(n, c2^n/n) < ((2n)c^2 2^{2n}/n^2)^{(c2^n/n)}$
  $< o(1)2^{2c2^n}$
  $< o(1)2^{2^n}$ (if $c \le ½$)
- probability a random function has a circuit of size $s = (½)2^n/n$ is at most

$$C(n, s)/B(n) < o(1)$$

Shannon's counting argument

- frustrating fact: *almost all* functions require huge **formulas**

Theorem (Shannon): With probability at least $1 - o(1)$, a random function

$$f:\{0,1\}^n \to \{0,1\}$$

requires a **formula** of size $\Omega(2^n/\log n)$.

Shannon's counting argument

- Proof (counting):
  - $B(n) = 2^{2^n}$ = # functions $f:\{0,1\}^n \to \{0,1\}$
  - # formulas with n inputs + size s, is at most

$$F(n, s) \le 4^s 2^s (n+2)^s$$

4^s binary trees with s internal nodes

2 gate choices per internal node

n+2 choices per leaf

Shannon's counting argument

- $F(n, c2^n/\log n) < (16n)^{(c2^n/\log n)}$
  $< 16^{(c2^n/\log n)}2^{(c2^n)} = (1 + o(1))2^{(c2^n)}$
  $< o(1)2^{2^n}$ (if $c \le ½$)
- probability a random function has a **formula** of size $s = (½)2^n/\log n$ is at most

$$F(n, s)/B(n) < o(1)$$

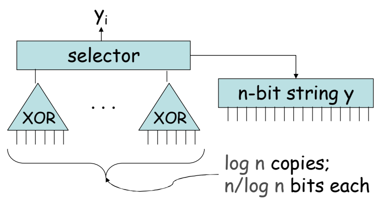
Andreev function

- best lower bound for formulas:

<u>Theorem</u> (Andreev, Hastad '93): the Andreev function requires $(\wedge,\vee,\neg)$-formulas of size at least

$$\Omega(n^{3-o(1)}).$$

Andreev function



the Andreev function A(x,y)

$A:\{0,1\}^{2n} \rightarrow \{0,1\}$

Random restrictions

- key idea: given function

  $$f:\{0,1\}^n \rightarrow \{0,1\}$$

  restrict by ρ to get f_ρ
  - ρ sets some variables to 0/1, others remain free
- $R(n, \epsilon n)$ = set of restrictions that leave ϵn variables free
- Definition: $L(f)$ = smallest $(\wedge,\vee,\neg)$ formula computing f (measured as leaf-size)

Random restrictions

- observation:

  $$E_{\rho\leftarrow R(n,\ \epsilon n)}[L(f_\rho)] \le \epsilon L(f)$$

  - each leaf survives with probability ϵ
- may shrink more…
  - propogate constants

<u>Lemma</u> (Hastad 93): for all f

$$E_{\rho\leftarrow R(n,\ \epsilon n)}[L(f_\rho)] \le O(\epsilon^{2-o(1)}L(f))$$

Hastad's shrinkage result

- Proof of theorem:
  - Recall: there exists a function

    $$h:\{0,1\}^{\log n} \rightarrow \{0,1\}$$

    for which $L(h) > n/2\log\log n$.
  - hardwire truth table of that function into y to get $A^*(x)$
  - apply random restriction from

    $$R(n,\ m = 2(\log n)(\ln \log n))$$

    to $A^*(x)$.

The lower bound

- Proof of theorem (continued):
  - probability given XOR is killed by restriction is probability that we "miss it" m times:

    $$(1 - (n/\log n)/n)^m \le (1 - 1/\log n)^m$$
    $$\le (1/e)^{2\ln \log n} \le 1/\log^2 n$$

  - probability even one of XORs is killed by restriction is at most:

    $$\log n(1/\log^2 n) = 1/\log n < ½.$$

The lower bound

- (1): probability even one of XORs is killed by restriction is at most:

  $\log n(1/\log^2 n) = 1/\log n < ½$.
- (2): by Markov:

  $\Pr[\ L(A^*_\rho) > 2\ E_{\rho\leftarrow R(n,\ m)}[L(A^*_\rho)]\ ] < ½$.

- Conclude: for *some* restriction ρ
  - all XORs survive, and
  - $L(A^*_\rho) \leq 2\ E_{\rho\leftarrow R(n,\ m)}[L(A^*_\rho)]$

The lower bound

- Proof of theorem (continued):
  - if all XORs survive, can restrict formula further to compute hard function h
    - may need to add ¬'s

$$L(h) = \mathbf{n}/2\log\log n \leq L(A^*_\rho)$$
$$\leq 2E_{\rho\leftarrow R(n,\ m)}[L(A^*_\rho)] \leq O((m/n)^{2-o(1)}L(A^*))$$
$$\leq O(\ ((\log n)(\ln\log n)/\mathbf{n})^{\mathbf{2\text{-}o(1)}}\ L(A^*)\ )$$

  - implies $\Omega(n^{3-o(1)}) \leq L(A^*) \leq L(A)$.