Notes on Complexity Theory Last updated: December, 2011

Lecture 28

Jonathan Katz

1 Circuit Lower Bounds

Recall that one motivation for studying non-uniform computation is the hope that it might be *easier* to prove lower bounds in that setting. (This is somewhat paradoxical, as non-uniform algorithms are more powerful than uniform algorithms; nevertheless, since circuits are more "combinatorial" in nature than uniform algorithms, there may still be justification for such hope.) The ultimate goal here would be to prove that $\mathcal{NP} \not\subset \mathcal{P}_{/\mathsf{poly}}$, which would imply $\mathcal{P} \neq \mathcal{NP}$. Unfortunately, after over two decades of attempts we are unable to prove anything close to this. Here, we show one example of a lower bound that we *have* been able to prove; we then discuss one "barrier" that partly explains why we have been unable to prove stronger bounds.

2 Parity Cannot be Solved by AC^0 Circuits

Recall that AC^0 is the set of languages/problems decided by *constant-depth*, polynomial-size circuits (with gates of unbounded fan-in). We consider the basis consisting of AND, OR, and NOT gates, though we do not count NOT gates when measuring the depth or size of the circuit. The parity function is given by $f(x_1 \cdots x_n) = x_1 \oplus \cdots \oplus x_n$. In this lecture we give the "polynomial proof" that parity cannot be computed in AC^0. We will actually prove something stronger:

Theorem 1 *For sufficiently large n, any depth-d circuit that computes parity on n-bit inputs must have at least $\frac{1}{50} \cdot 2^{0.5 \cdot n^{1/2d}}$ gates.*

Thus, for any fixed depth-bound d, any circuit family computing parity grows as 2^{n^ε} for some $\varepsilon > 0$.

Proof Fix a circuit C of depth d that computes parity on inputs of length n. Let $x_1, \ldots, x_n$ denote the inputs to the circuit. We will assume that C has only OR gates and NOT gates; this assumption is without loss of generality since we may convert any AND gate to a combination of OR and NOT gates using De Morgan's laws (by setting $\bigwedge_i a_i = \neg \bigvee_i (\neg a_i)$) without affecting the size or depth of the circuit.

Let $\mathbb{F}_3 = \{-1, 0, 1\}$ be the field of size 3. Say a polynomial $p \in \mathbb{F}_3[x_1, \ldots, x_n]$ is *proper* if $p(x_1, \ldots, x_n) \in \{0, 1\}$ whenever $x_1, \ldots, x_n \in \{0, 1\}$. Note that any proper polynomial can be viewed as a boolean function in the natural way.

The proof hinges on two lemmas: we first show that any circuit in AC^0 can be approximated fairly well by a (proper) low-degree polynomial, and then show that parity cannot be approximated well by any low-degree polynomial.

Lemma 2 *For every integer $t > 0$, there exists a (proper) polynomial of total degree $(2t)^d$ that differs with C on at most $\mathsf{size}(C) \cdot 2^{n-t}$ inputs.*

Proof We will associate a proper polynomial with each wire of the circuit, and then bound the error introduced. Begin at the input wires and associate the monomial x_i to the input x_i. Now consider the output wire of some gate g, all of whose input wires have already been associated with polynomials. Then:

- If g is a NOT gate, and its input wire is associated with the polynomial p, then associate its output wire with $1 - p$. Note that this does not increase the degree of the polynomial.
- If g is an OR gate with k input wires associated with the polynomials $p_1, \ldots, p_k$, then do the following:

  Choose sets $S_1, \ldots, S_t \subseteq [k]$ (see below for how these sets are chosen), and define $q_i = \left(\sum_{j \in S_i} p_j\right)^2$ for $i = 1, \ldots, t$. Then set $p = 1 - \prod_{i=1}^{t}(1 - q_i)$.

  (Note that p is just the OR of the q_i.) If the maximum (total) degree of the $\{p_i\}$ is b, then the (total) degree of polynomial p is at most $2tb$. Note further that p is proper.

For a given wire with associated polynomial p, an *error* is an input $x_1, \ldots, x_n$ on which the value of the wire and the value of p differ. We now bound the fraction of errors in the polynomial p^* associated with the output wire of the circuit. No errors are introduced at input wires or at NOT gates. Looking at any OR gate with k input wires associated with the polynomials $p_1, \ldots, p_k$, we claim that there is *some* choice of subsets $S_1, \ldots, S_t \subseteq [k]$ that will not introduce too many errors. On any input where all the p_i's evaluate to 0, the resulting polynomial p will also evaluate to 0. Consider any input where at least one of the p_i's evaluates to 1, and let $S_1, \ldots, S_t$ be random subsets of $[k]$. With probability at least half over choice of subset S_j, polynomial q_j will evaluate to 1. If *any* of the q_j evaluate to 1 then so does p. So the probability that p does not evaluate to 1 is at most 2^{-t}. By an averaging argument, this implies the *existence* of some collection of subsets that introduces errors on at most a 2^{-t} fraction of the inputs at this gate.

Taking a union bound, we conclude that p^* is a polynomial of degree at most $(2t)^d$ having at most $\mathsf{size}(C) \cdot 2^{n-t}$ errors with respect to C. ■

Setting $t = n^{1/2d}/2$ we get a polynomial of degree at most $\sqrt{n}$ that differs from C on at most $\mathsf{size}(C) \cdot 2^{n-t}$ inputs.

Lemma 3 *Let $p \in \mathbb{F}_3[x_1, \ldots, x_n]$ be a proper polynomial of degree at most $\sqrt{n}$. Then for sufficiently large n the polynomial p differs from the parity function on at least $2^n/50$ inputs.*

Proof Consider the "translated" parity function $\mathsf{parity}' : \{-1, 1\}^n \to \{-1, 1\}$ defined as

$$\mathsf{parity}'(x_1, \ldots, x_n) = \prod_i x_i.$$

Since $\mathsf{parity}'(x_1, \ldots, x_n) = \mathsf{parity}(x_1 - 1, \ldots, x_n - 1) + 1$, there exists a polynomial p' of degree at most $\sqrt{n}$ that agrees with parity' on the same number of inputs for which p agrees with parity.

Let $S \subseteq \{-1, 1\}^n$ be the set of inputs on which p' and parity' agree, and let $\mathcal{F}$ denote the set of *all* functions from S to $\mathbb{F}_3$. Note that $|\mathcal{F}| = 3^{|S|}$. Now, for every function $f \in \mathcal{F}$ we can associate a polynomial $p_f \in \mathbb{F}_3[x_1, \ldots, x_n]$ that agrees with f for all $x \in S$: just set

$$p_f(x_1, \ldots, x_n) = -\sum_{y \in S} f(y) \cdot \prod_{i=1}^{n} (y_i \, x_i + 1) \, .$$

Although p_f, as constructed, has degree 1 in each input variable, the total degree of p_f may be as large as n. We claim that, in fact, we can associate with each f a polynomial $\hat{p}_f$ whose degree is at most $n/2 + \sqrt{n}$. To see this, fix f and p_f and look at some monomial $\pm \prod_{i \in T} x_i$ appearing in p_f where $|T| > n/2 + \sqrt{n}$. For any $x \in S \subseteq \{-1, 1\}^n$ we have

$$\begin{aligned} \pm \prod_{i \in T} x_i &= \pm \prod_{i=1}^{n} x_i \cdot \prod_{i \notin T} x_i \\ &= \pm p'(x) \cdot \prod_{i \notin T} x_i \,. \end{aligned}$$

Since p' has degree at most $\sqrt{n}$, we see that we can re-write p_f as a polynomial $\hat{p}_f$ that agrees with p_f on S and has degree at most $n/2 + \sqrt{n}$.

The number of monomials whose total degree is at most $n/2 + \sqrt{n}$ is $\sum_{i=0}^{n/2+\sqrt{n}} \binom{n}{i}$, which is less than $49 \cdot 2^n/50$ for large enough n. So the total number of polynomials whose degree is at most $n/2 + \sqrt{n}$ is upper bounded by $3^{49 \cdot 2^n/50}$. Given that $|\mathcal{F}| = 3^{|S|}$, this means we must have $|S| \leq 49 \cdot 2^n/50$ as claimed. ■

To complete the proof, we just combine the two lemmas. The first lemma gives a polynomial p of degree at most $\sqrt{n}$ that differs from parity on at most $\mathsf{size}(C) \cdot 2^{n - n^{1/2d}/2}$ inputs. The second lemma tells us that, for large enough n, we must have $\mathsf{size}(C) \cdot 2^{n - n^{1/2d}/2} \geq 2^n/50$. We conclude that $\mathsf{size}(C) \geq \frac{1}{50} \cdot 2^{0.5 \cdot n^{1/2d}}$, completing the proof. ■

3 Limits of Proving Lower Bounds: Natural Proofs

The lower bound proved in the previous section is, to a rough approximation, about the best we are able to show. Why is that? We explore one reason here.

We formalize a notion of proving lower bounds using "natural" proof techniques. We then show that natural proofs *cannot* be used to prove strong lower bounds assuming some (widely believed) cryptographic assumptions hold.[1]

3.1 Defining Natural Proofs

Let $\mathcal{C}_0, \mathcal{C}_1$ denote classes of predicates. ($\mathcal{C}_0, \mathcal{C}_1$ need to satisfy certain technical restrictions, but these conditions are not very restrictive.) For example, $\mathcal{C}_0$ or $\mathcal{C}_1$ could be AC^0 or $\mathcal{P}_{/\mathsf{poly}}$, or even something more specific like "depth-5 circuits having at most n^2 gates". Say we want to show that some function $f : \{0,1\}^* \to \{0,1\}$ is not in $\mathcal{C}_0$. One way to prove this would be to define a "hardness predicate" P on functions such that, for n large enough, $P(f_n) = 1$ (where f_n denotes the restriction of f to n-bit inputs) but for any $g \in \mathcal{C}_0$ and n large enough, $P(g_n) = 0$. (Here we view P as taking the *truth table* of f_n as input. Note that the truth table of f_n has size 2^n, and we measure complexity of P in terms of that length.) A proof of this form is called a $\mathcal{C}_1$-*natural against* $\mathcal{C}_0$ if P satisfies two conditions: (1) P is in $\mathcal{C}_1$ (this is called the *constructiveness* condition),

[1] This is an interesting situation, as it means that one way to make progress on proving lower bounds (i.e., to prove that some function is "hard") would be to show better upper bounds (namely, to show that certain cryptographic problems are actually "easy".)

and (2) for n sufficiently large, a random predicate $g_n : \{0,1\}^n \to \{0,1\}$ satisfies $P(g_n) = 1$ with probability at least $1/n$ (this is called the *largeness* condition).

Why are these properties "natural"? There are two types of answers here. The first is just to observe that (almost?) all known lower bounds do, in fact, satisfy these properties. For example, in the lower bound for parity that we showed in the previous section, the predicate P corresponds informally to

$$P(f_n) = 0 \Leftrightarrow f_n \text{ can be "well approximated" by a "low-degree" polynomial.}$$

A random function cannot be approximated by a low-degree polynomial, with high probability. Constructiveness is more difficult to show, and actually requires us to consider a slightly different predicate P' (for which the largeness condition still holds); we refer to [4] for details. All-in-all, it is possible to show that the parity lower bound is NC^2-natural against AC^0. (Interestingly, previous lower bounds for parity were AC^0-natural against AC^0.)

The second argument in favor of our definition of "natural" is to appeal to intuition. This argument is somewhat harder to make (which gives hope that we can circumvent the barrier imposed by natural proofs, and find new proof techniques for showing lower bounds). Constructiveness is motivated by the fact that, as part of any lower-bound proof of the above form, we must show that $P(f_n) = 1$; it seems that any "constructive" proof of this fact should involve some "efficient" computation involving the truth table of f_n. The largeness condition is perhaps more natural. For starters, random functions are typically hardest. Moreover, every proof that a given function $f_n : \{0,1\}^n \to \{0,1\}$ cannot be computed by a circuit of S also proves that at least half the functions on n bits cannot be computed by a circuit of size $\approx S/2$. If not, then write

$$f_n(x) = (f_n \oplus g_n) \oplus g_n$$

for a random function $g_n : \{0,1\}^n \to \{0,1\}$; with probability strictly greater than 0, both g_n and $f_n \oplus g_n$ can be computed by a circuit of size $\approx S/2$ (note that both g_n and $f_n \oplus g_n$ are random functions), but then f_n can be computed by a circuit of size S.

3.2 Ruling Out Natural Proofs of Lower Bounds

For simplicity, let us fix some n instead of working asymptotically. Then we can view $\mathcal{C}_0, \mathcal{C}_1$ as classes of functions on n-bit and 2^n-bit inputs, respectively. Say we have a function $f : \{0,1\}^n \to \{0,1\}$ and want to prove that f is not in $\mathcal{C}_0$ using a $\mathcal{C}_1$-natural proof. This means that we want to define a predicate $P \in \mathcal{C}_1$ such that $P(f) = 1$ but $P(g) = 0$ for all $g \in \mathcal{C}_0$.

Let $\mathcal{F}$ denote a keyed function mapping inputs of length n to boolean outputs, and having a key of length m, i.e., $\mathcal{F} = \{F_k : \{0,1\}^n \to \{0,1\}\}_{k \in \{0,1\}^m}$. We say that $\mathcal{F}$ *can be computed in* $\mathcal{C}_0$ if, for any key $k \in \{0,1\}^m$, the function $F_k : \{0,1\}^n \to \{0,1\}$ is in $\mathcal{C}_0$. (Note in particular that the function $F(k,x) \stackrel{\text{def}}{=} F_k(x)$ may have complexity "higher" than $\mathcal{C}_0$.) Informally, $\mathcal{F}$ is a *pseudorandom function* (PRF) if it is hard to distinguish whether a given function $h : \{0,1\}^n \to \{0,1\}$ is equal to F_k for a random key $k \in \{0,1\}^m$, or whether h is a random boolean function on n-bit inputs. Our notion of distinguishing will be very strong: rather than considering distinguishers that make oracle queries to h, we simply provide the distinguisher with the entire truth table of h.[2] Formally,

[2]For those used to thinking about polynomial-time distinguishers of cryptographic PRFs this may seem strange. If it helps, one can think of m as the security parameter and consider, for example, the case where $n = O(\log^2 m)$. Then we are just requiring security against slightly super-polynomial distinguishers running in time $\mathsf{poly}(2^n) = m^{O(\log m)}$.

then, we say that $\mathcal{F}$ is *pseudorandom against* $\mathcal{C}_1$ if for every distinguisher $D \in \mathcal{C}_1$ we have

$$\left| \Pr_{k \leftarrow \{0,1\}^m}[D(F_k) = 1] - \Pr_{h \leftarrow \mathsf{Func}_n}[D(h) = 1] \right| < 1/n,$$

where Func_n denote the space of all predicates on n-bit inputs.

We can now state the main result:

Theorem 4 *Assume there exists an $\mathcal{F}$ that can be computed in $\mathcal{C}_0$ and is pseudorandom against $\mathcal{C}_1$. Then there is no $\mathcal{C}_1$-natural proof against $\mathcal{C}_0$.*

Proof A $\mathcal{C}_1$ natural proof against $\mathcal{C}_0$ would imply the existence of a predicate $P \in \mathcal{C}_1$ such that $P(g) = 0$ for all $g \in \mathcal{C}_0$, while $\Pr_{h \leftarrow \mathsf{Func}_n}[P(h) = 1] \geq 1/n$. But then P acts as a distinguisher for $\mathcal{F}$, using the fact that $F_k \in \mathcal{C}_0$ for every key k. ■

Under suitable cryptographic hardness assumptions (e.g., the assumption that the discrete logarithm problem has hardness 2^{n^ε} for some $\varepsilon > 0$, or an analogous assumption for hardness of subset sum) there are pseudorandom functions that can be computed in NC^1 (or even[3] TC^0) and are pseudorandom against $\mathcal{P}$. Thus, if these cryptographic conjectures are true, there are no $\mathcal{P}$-natural proofs even against weak classes like NC^1 or TC^0. This helps explain why current circuit lower bounds are "stuck" at AC^0 and[4] ACC^0.

Bibliographic Notes

This proof given here that parity is not in AC^0 is due to Razborov [3] and Smolensky [5] (who proves a more general result); earlier proofs (using a different approach) were given by Furst, Saxe, and Sipser, by Yao, and by Håstad. Good surveys of circuit lower bounds include the (old) article by Boppana and Sipser [1] and the (new) book by Jukna [2]. Natural proofs were introduced by Razborov and Rudich [4].

References

[1] R. Boppana and M. Sipser. The Complexity of Finite Functions. In *Handbook of Theoretical Computer Science, vol. A: Algorithms and Complexity*, J. van Leeuwen, ed., MIT Press, 1990.

[2] S. Jukna. *Boolean Function Complexity: Advances and Frontiers*. Springer, 2012.

[3] A. Razborov. Lower Bounds on the Size of Bounded Depth Networks Over a Complete Basis with Logical Addition. *Matematicheskie Zametki* 41:598–607, 1987.

[4] A. Razborov and S. Rudich. Natural Proofs. *J. Computer and System Sciences* 55(1): 24–35, 1997.

[5] R. Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. STOC 1987.

[3] TC^0 is the class of predicates that can be computed by constant-depth circuits of polynomial size, over a basis of unbounded fan-in threshold gates. Note that $\mathsf{AC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1$.

[4] ACC^0 is the class of predicates that can be computed by constant-depth circuits of polynomial size, over a basis of unbounded fan-in AND, OR, and mod m gates (for any fixed constant m). Note that $\mathsf{AC}^0 \subseteq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0$.