18.405J/6.841J: Advanced Complexity Theory Spring 2016

Lecture 7: Razborov-Smolensky

Mohammad Bavarian

Scribe: Brian Chen
Scribe Date: Spring 2016

1 Intro

In the last lecture, we showed that PARITY $\notin$ AC^0: in other words, there is no bounded-depth, polynomial-size circuit that computes PARITY.

Today, we will improve the theorem. Broadly speaking, there are two ways we might try to do so: qualitatively (proving lower bounds on stronger circuit models) and quantitatively (proving tighter bounds with the same circuit models). We will pursue the former direction today, in accordance with the Sipser program, the direction of complexity theory research starting from the 80s and 90s. The goal is to work our way up to understanding P/poly and from there P.

2 New Circuit Models

Here are some complexity classes that are targets of the Sipser program after AC^0, listed as a series of inclusions:

$$AC^0 \subseteq AC^0[m] \subseteq ACC^0 \subseteq TC^0.$$

The complexity classes $AC^0[m]$ and ACC^0 are based on circuits with bounded depth, just like circuits in AC^0, but they are more powerful because their circuits may include *mod-m gates*:

Definition 1. *Let $m \geq 2$ be an integer; then a* mod-m *gate is a gate that accepts unbounded fan-in and outputs 1 iff the number of inputs that are 1 is not 0 mod m. Formally, on inputs $y_1, \ldots, y_k$:*

$$\mathrm{Mod}_m(y_1, \ldots, y_k) = \begin{cases} 0 & \text{if } \sum y_i \equiv 0 \bmod m \\ 1 & \text{if } \sum y_i \not\equiv 0 \bmod m \end{cases}.$$

Definition 2. *For an integer $m \geq 2$, the class $AC^0[m]$ consists of languages decidable by bounded-depth polynomial-size circuits with AND, OR, NOT, and mod-m gates.*

The complexity class ACC^0 is like AC^0, except that it allows mod-m gates for arbitrary m instead of just a fixed one. The class TC^0 instead uses *threshold gates*, which are more powerful than AND, OR, and mod-m gates. Neither of these classes are involved in today's main proof, though, so we defer discussion and formal definitions of these stronger classes to the end.

3 Main Theorem

Today, we will show the following theorem:

Theorem 3.

$$\text{PARITY} \notin \text{AC}^0[3].$$

In fact, any depth-d circuit using AND, OR, NOT, and mod-3 gates that computes PARITY *must have*

$$\text{SIZE} \geq 2^{\Omega(n^{1/2d})}.$$

Our strategy will be to approximate any circuits in $\text{AC}^0[3]$ with a low-degree polynomial over $\mathbb{F}_3$, and then to prove that low-degree polynomials cannot approximate PARITY that well, leading to a contradiction.

3.1 Arithmetization

In the proof, we will view $\{0,1\}^n$ as a subset of $\mathbb{F}_3^n$ (where $\mathbb{F}_3$ is the field with 3 elements).

Given a circuit $C : \{0,1\}^n \to \{0,1\}$, we will develop a polynomial $\tilde{C} : \mathbb{F}_3^n \to \mathbb{F}_3$ that behaves like C on inputs in the set $\{0,1\}^n$.

Now, there is a naïve way to do this:

- Given a boolean b, computing NOT is just $1 - b$;
- Given a list of booleans $b_1, b_2, \ldots, b_k \in \{0,1\}$, computing AND is just taking the product $b_1 b_2 \cdots b_k$;
- Given a list of booleans $b_1, b_2, \ldots, b_k \in \{0,1\}$, computing OR is, by de Morgan's law, just $1 - (1 - b_1) \cdots (1 - b_k)$;
- Given a list of booleans $b_1, b_2, \ldots, b_k \in \{0,1\}$, computing mod-3 is just the square of the sum $(b_1 + b_2 + \cdots + b_k)^2$ (since $0^2 \equiv 0$ and $1^2 \equiv 2^2 \equiv 1 \bmod 3$; note that this generalizes to other moduli m by taking the $m-1$th power in $\mathbb{F}_m$).

The problem with this plan arises in the simulation of AND and OR gates when those gates are not narrow. Unbounded fan-in can cause our polynomials to have very high degree, but we want low-degree polynomials because we understand them better. So we will settle for an *approximation*, a $\tilde{C}$ that behaves like C on a large fraction of, but not all, inputs.

3.2 Proof

We will divide our proof into two lemmas, which we will prove later. First, a definition:

Definition 4. *A polynomial $p : \mathbb{F}_3^n \to \mathbb{F}_3$ is called* proper *if it maps $\{0,1\}^n$ to $\{0,1\}$.*

Lemma 5. *Let t be an integer, $t \geq 1$, and let C be an $\text{AC}^0[3]$ circuit of depth d. Then there exists a* ***proper*** *polynomial of degree at most $(2t)^d$ which agrees with C on at least the fraction $1 - \text{SIZE}(C)/2^t$ of all inputs in $\{0,1\}^n$.*

In this lemma, t is a parameter which we will adjust later.

Lemma 6. *Let $g : \mathbb{F}_3^n \to \mathbb{F}_3$ be a **proper** polynomial with degree $\leq \sqrt{n}$. Then g agrees with* PARITY *on at most* $49/50$ *of inputs in* $\{0,1\}^n$.

(The constant 49/50 is not tight, but rather unimportant. For better results, we'd be more interested in improving $\sqrt{n}$, say, to $n^{2/3}$.)

Assuming these two lemmas, we now give the proof of the main theorem:

Proof. Suppose for the sake of contradiction that PARITY $\in \mathrm{AC}^0[3]$, so that there is some fixed positive integer d such that, for all input sizes, there is a depth-d $\mathrm{AC}^0[3]$ circuit that computes PARITY. Let

$$t = \frac{n^{1/2d}}{2}$$

and apply Lemma 5; then there is a proper polynomial p with degree at most $\sqrt{n}$ that agrees with PARITY on $1 - \frac{\mathrm{SIZE}(C)}{2^{\frac{n^{1/2d}}{2}}}$ of inputs. Then, by lemma 6,

$$\frac{\mathrm{SIZE}(C)}{2^{\frac{n^{1/2d}}{2}}} \geq \frac{1}{50} 2^{n^{1/2d}},$$

which implies

$$\mathrm{SIZE}(C) \geq \frac{1}{50} 2^{\frac{n^{1/2d}}{2}},$$

contradicting the polynomial size of C and concluding the proof. More generally, the same proof shows that any bounded-depth circuit using AND, OR, NOT, and mod-3 gates that computes PARITY must have at least this size.

4 Proof of First Lemma

Without loss of generality, assume the circuit consists only of mod-3, NOT, and OR gates. (AND gates can be rewritten as a combination of one OR gate and many NOT gates using de Morgan's laws. Note that this increases the SIZE of the circuit by adding many NOT gates, but, as we shall see, NOT gates do not affect the size of our construction, so this is fine.)

Taking each layer in turn, we will approximate each gate with a low-degree polynomial. Specifically, we will approximate each gate in layer k (counting from the bottom) with a polynomial of degree at most $(2t)^k$.

Clearly, every input on the bottom layer is just a degree-1 polynomial, a monomial of the form x_i, so the base case works.

Now, suppose that we have approximated every polynomial in layer k with a polynomial of degree at most $(2t)^k$, and we wish to continue this to the next layer. Consider any gate on the $(k+1)$th layer:

- **If it is a NOT gate** and its input has been approximated with the polynomial $\tilde{f}$, we simply approximate the output as the polynomial $1 - \tilde{f}$. This is an exact simulation of the NOT gate! Also, it does not increase the degree of the input polynomial, which is why adding NOT gates is unimportant.

- **If it is a mod-3 gate** and its inputs have been approximated with the polynomials $\tilde{f}_i$, we simply approximate the output as the polynomial

$$\left(\sum_{k=1}^{s} \tilde{f}_i\right)^2.$$

The degree of this polynomial is at most $2(2t)^k \geq (2t)^{k+1}$. This is also an exact simulation of the mod-3 gate: if the inputs are all in $\{0, 1\}$, then the sum counts exactly how many of them are 1 and the final squaring maps 0 to 0 and non-zero to 1.

- **If it is an OR gate**, we will finally need to use approximation. To exactly simulate an OR gate with s inputs, we'd have to use something like the polynomial

$$1 - \prod_{i=1}^{s}(1 - \tilde{f}_i),$$

which has degree $s(2t)^k \gg (2t)^{k+1}$, so this does not work (unless the gate is narrow enough, $s \geq 2t$; but we cannot rely on that.) Instead, we will approximate the gate as follows. We pick t random subsets $L_1, L_2, \ldots, L_t \subseteq \{1, 2, \ldots, s\}$ (where each element has an independent $1/2$ chance of being in each subset), and approximate the OR gate with the polynomial

$$\tilde{f} = 1 - \prod_{i=1}^{t}\left(1 - \left(\sum_{m \in T_i} \tilde{f_m}\right)^2\right).$$

We observe that, if all inputs are 0, then every sum is 0, every multiplicand in the product is 1, and $\tilde{f} = 0$, which is correct. If any input, say $\tilde{f}_j$, is 1, then each sum $\sum_{m \in T_i} \tilde{f_m}$ has probability $\geq 1/2$ of being nonzero (T_i has probability $1/2$ of including or excluding j, and those give different sums, of which at least one is nonzero); if any sum is nonzero, then its square is 1, the corresponding multiplicand is 0, and $\tilde{f} = 1$.

Thus, for all inputs, the OR simulation is correct with probability $\geq 1 - 1/2^t$.

Now, by applying a union-bound, we get a polynomial that disagrees with at most $\text{SIZE}(C)/2^t$ of all possible inputs, and we are done! (Note that, in bounding the error, we did not need the full size of the circuit, only the number of AND and OR gates.) □

The proof again suggests the important about distinguishing between narrow and wide gates: simulation of narrow OR gates can be done exactly in the degree we've allotted ourselves, whereas simulation of wide OR gates is where the random selection becomes important.

4.1 Proof of Second Lemma

We recall the lemma statement: Let g be a proper polynomial with degree $\leq \sqrt{n}$. The g agrees with PARITY on at most 49/50 of inputs.

A natural question is where the fraction 49/50 comes from. It turns out it arises from a weak bound on the binomial distribution, which we will not prove:

Fact:

$$\sum_{i=0}^{n/2+\sqrt{n}} \binom{n}{i} \leq \frac{49}{50} \cdot 2^n.$$

Now, let us change bases $\{0,1\} \to \{-1,1\}^n$. More precisely, consider the polynomial

$$q(x_1, \ldots, x_n) = 1 + g(x_1 + 1, \ldots, x_n + 1).$$

Then we observe that q maps $\{-1,1\}^n$ to $\{-1,1\}$, and after the change of base, PARITY becomes the simple product of every variable, $\prod x_i$. So for each input $(x_1, \ldots, x_n) \in \{0,1\}^n$, g agrees with PARITY iff $q(x_1 + 1, \ldots, x_n + 1)$ agrees with $\prod(x_i + 1)$. Thus, we want to understand for how many inputs the equation $q(x_1, \ldots, x_n) = \prod x_i$ can hold.

Let $G = \{u \in \{-1,1\}^n \mid q(u) = \prod_{x \in u} x\}$. Now, pick an arbitrary function $p : G \to \mathbb{F}_3$ and extend it to $p : \mathbb{F}_3^n \to \mathbb{F}_3$. Note that, over finite fields, every function can be expressed as a polynomial, so assume p is a polynomial. The idea below is that the properties of q and $\mathbb{F}_3$ will allow us to simplify p to a low-degree polynomial without affecting its behavior on G, which bounds the number of possible ways one could have picked a function $G \to \mathbb{F}_3$ at the start, which in turns bounds $|G|$.

First, express p as a sum of monomials:

$$p = \sum a_{i_1, i_2, \ldots, i_n} x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}.$$

Since we only care about p's behavior on $G \in \{-1,1\}^n$, and every $x_i^2 = 1$ for any x_i in this set, we can reduce every i_j mod 2. As a result, all $i_j \in \{0,1\}$, and p becomes multilinear without changing its behavior on G.

Now, we can express p as the sum

$$p = \sum_{S \subseteq [n]} a_S \prod_{i \in S} x_i$$

(where $[n] = \{1, 2, \ldots, n\}$). Consider some S such that $|S| \geq n/2$. Then

$$\prod_{i \in S} = \prod_{i \in \overline{S}} x_i \cdot \prod_{i \in [n]} x_i,$$

again because every x_i that is multiplied twice in the RHS simplifies to 1. By replacing the final product with q, we have:

$$\prod_{i \in S} = \prod_{i \in \overline{S}} x_i \cdot q(x_1, \ldots, x_n)$$

The degree of the RHS is now $\leq n/2 + \sqrt{n}$. Again, this equation is true for any $(x_i \ldots) \in G$. This means that, for every term with degree $> n/2 + \sqrt{n}$ in p, we can replace it with a term of degree $\leq n/2 + \sqrt{n}$ without affecting the behavior of p on G.

Now, the space of remaining polynomials p are those polynomials that are multilinear and have total degree $\leq n/2 + \sqrt{n}$. So its dimension (over $\mathbb{F}_3$) is less than $\sum_{i=1}^{n/2+\sqrt{n}} \binom{n}{i} \leq \frac{49}{50} 2^n$. But the dimension over $\mathbb{F}_3$ of the choices for the original function $p : G \to \mathbb{F}_3$ is just $|G|$. Therefore,

$$|G| \leq \frac{49}{50} 2^n,$$

as desired.

5 Other Comments

5.1 Generalization

The main proof today generalizes to proving that mod-p gates are not in $\mathrm{AC}^0[q]$ for any distinct primes p, q. However, it does not generalize to composites, and indeed, when m is composite, it was considerably difficult to prove things about $\mathrm{AC}^0[m]$. For example, no good bounds were known for $\mathrm{AC}^0[6]$ for 30 years; this only changed recently. (Note that $\mathrm{AC}^0[6]$ is more powerful than $\mathrm{AC}^0[2]$ and $\mathrm{AC}^0[3]$, since a mod-6 gate can simulate a mod-2 gate by taking three copies of every input and a mod-3 gate by taking two copies of every input.)

5.2 ACC^0 and TC^0

We define the stronger circuit models mentioned at the start of the notes:

Definition 7. ACC^0 *is the class of languages decidable by bounded-depth polynomial-size circuits with AND, OR, NOT, and mod-m gates for any m. Equivalently, since only finitely many types of mod-m gates can be used in any given circuit,*

$$\mathrm{ACC}^0 = \bigcup_{m_1,\ldots,m_k \in \mathbb{N}} \mathrm{AC}^0[m_1, m_2, \ldots, m_k].$$

(For more than one integer $m_1, m_2 \ldots \geq 2$, the class $\mathrm{AC}^0[m_1, m_2, \ldots]$ consists of languages decidable by bounded-depth polynomial-size circuits with AND, OR, NOT, and mod-m_i gates for any i.)

Definition 8. *Let m be an integer. A* threshold gate *is a gate that accepts unbounded fan-in and outputs 1 iff the number of inputs that are 1 is greater than or equal to m.*

Definition 9. TC^0 *is the class of languages decidable by bounded-depth polynomial-size circuits with threshold gates.*

Remark. In our models of bounded-depth polynomial-size circuits, threshold gates are more powerful than AND gates, OR gates, and mod-m gates.

- AND gates and OR gates are special cases of threshold gates, where m is set to either the number of inputs or 1, respectively.
- Mod-m gates can be built with a polynomial number of threshold gates and NOT gates arranged with bounded depth. First, note that we can determine using threshold gates whether the number of inputs that are 1 is *exactly* k for some k: we just copy all inputs twice and test whether the number of true inputs is $\geq k$ and $\not\geq k+1$ Suppose the number of inputs is k. Then we can test whether the number of inputs that are true is $0, m, \ldots$ for each possible value it could take on; k is polynomial in n, so the number of possible values is also only polynomial in n and the number of gates we need is also polynomial.

The consequence of all this is that TC^0 is more powerful than ACC^0.

5.3 Looking Forward

In the next lecture, we prove NEXP $\not\subseteq$ ACC^0, a result by Ryan Williams in 2010 that was the first result that went beyond today's results.

It is also conjectured, but unproven, that MAJ $\notin$ ACC^0, where MAJ is the majority problem: given a list of boolean inputs, is the majority of them true? Almost nothing is known about lower bounds for TC^0.

18.405J / 6.841J Advanced Complexity Theory

Spring 2016