CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 25

April 28, 2010 **Scribe: Kevin Lee**

1 Agenda

1. Lower Bounds for Algebraic Circuits
2. Survey of Algebraic Circuit Lower Bounds
3. Survey of Boolean Circuit Lower Bounds

2 An Algebraic Circuit Lower Bound

Theorem 1 *The smallest algebraic circuit over any field $\mathbb{F}$ computing $p(x_1, \ldots, x_n) = x_1^d + \cdots x_n^d$ has size $\Theta(n \log d)$ provided $char(\mathbb{F}) \nmid d$ where $char(\mathbb{F})$ is the* characteristic *of the field $\mathbb{F}$ (i.e. the smallest positive integer k such that $k \stackrel{\text{def}}{=} 1 + 1 + \cdots + 1 = 0$ in $\mathbb{F}$, or zero if no such k exists).*

Proof of Upper Bound: We can always compute $p(x_1, \ldots, x_n)$ with size $O(n \log d)$ circuits by performing fast exponentiation via repeated squaring for each term, and then adding the terms up. (Note that if $\text{char}(\mathbb{F}) = d$, then

$$x_1^d + \cdots x_n^d = (x_1 + \cdots + x_n)^d$$

gives a faster way of computing the polynomial, motivating the additional condition in the theorem.) ■

Proof of Lower Bound: Proving the lower bound uses the "Method of Partial Derivatives," which is the idea that the complexity of the partial derivatives of a function explains something about the complexity of the function itself. For a polynomial $p(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n \geq 0} c_{i_1, \ldots, i_n} x_1^{i_1} \cdots x_n^{i_n}$, the partial derivative $\partial p / \partial x_j$ is defined to be the polynomial

$$\frac{\partial p}{\partial x_j}(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n \geq 0, i_j \geq 1} i_j \cdot c_{i_1, \ldots, i_n} \cdot x_1^{i_1} \cdots x_j^{i_j - 1} \cdots x_n^{i_n}.$$

This formal definition of derivative for polynomials doesn't require taking limits, and thus even makes sense over finite fields. It can be verified to satisfy the familiar properties of derivatives, such as the product rule and chain rule.

Before we prove the lower bound, we need to develop some techniques involving partial derivatives.

Lemma 2 *If $p(x_1, \ldots, x_n)$ is computable by an algebraic circuit with s binary gates, then there is an algebraic circuit with $5s$ binary gates computing*

$$p(x_1, \ldots, x_n), \frac{\partial p}{\partial x_1}(x_1, \ldots, x_n), \ldots, \frac{\partial p}{\partial x_n}(x_1, \ldots, x_n)$$

simultaneously.

Note that here we are converting a circuit computing a single polynomial to one computing $n+1$ related polynomials with only a constant-factor blow-up in size. The intuition for why this is possible is that each gate of the circuit has constant arity, so only has a constant number of nontrivial partial derivatives (when viewed as a function of the earlier gates in the circuit).

Proof: We induct on s.

Base case: There are no binary gates. Which means that $p(x_1, \ldots, x_n) = x_i$ or $p(x_1, \ldots, x_n) = c$. (Note: We are only counting the binary gates, not the inputs) In the former case, we have that

$$\frac{\partial p}{\partial x_j}(x_1, \ldots, x_n) = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$$

All of these can be computed with no binary gates. In the latter case (where p is a constant), all of the partial derivatives are zero.

Induction step: We have C computing $p(x_1, \ldots, x_n)$. Pick a gate $g(x_i, x_j)$ just above the leaves. We have four cases:

$$g = x_i + x_j$$

$$g = x_i x_j$$

$$g = x_i + c$$

$$g = x_i c$$

We are only going to do the multiplication case because it is the most interesting. The proof for the other cases is similar.

Let $C'(x_1, \ldots, x_n, z)$ be circuit C with g replaced by a new variable z. It computes a polynomial $q(x_1, \ldots, x_n, z)$ that has one fewer binary gate, so by induction we have a circuit $C''(x_1, \ldots, x_n, z)$ that computes q and all of its $n+1$ partial derivatives. We now show how to use C'' plus 5 more binary gates to compute p and all its n partial derivatives.

Since

$$p(x_1, x_2, \ldots, x_n) = q(x_1, \ldots, x_n, g(x_i, x_j))$$

computing $p(x_1, x_2, \ldots, x_n)$ can be done first computing $g(x_i, x_j)$, evaluating C'' on $(x_1, \ldots, x_n, g(x_i, x_j))$, and taking the first output.

For the partial derivatives of p, we use the chain rule to obtain:

$$\frac{\partial p}{\partial x_l}(x_1, \ldots, x_n) = \left(\frac{\partial q}{\partial x_l} + \frac{\partial q}{\partial z}\frac{\partial g}{\partial x_l}\right)(x_1, \ldots, x_n, g(x_i, x_j)).$$

Note we have already computed $(\partial q/\partial x_l)(x_1, \ldots, x_n, g(x_i, x_j))$ and $(\partial q/\partial z)(x_1, \ldots, x_n, g(x_i, x_j))$ in evaluating $C''(x_1, \ldots, x_n, g(x_i, x_j))$. Moreover, $\partial g/\partial x_l$ is 0 unless $l \in \{i, j\}$, so no additional computation is needed for most of the partial derivatives of p. In case $l \in \{i, j\}$, we have:

$$\frac{\partial g}{\partial x_l} = \begin{cases} x_j & l = i \\ x_i & l = j \end{cases}$$

In these cases, we can compute the partial $\partial p/\partial x_l$ by adding a multiplication gate and an addition gate.

We have added 5 gates total as desired, so we are done. ■

Lemma 3 *Any algebraic circuit computing* $\left(x_1^d, \ldots, x_n^d\right)$ *must have at least* $\Omega(n \log d)$ *multiplication gates.*

Proof: Say we have circuit of size s computing $\left(x_1^d, \ldots, x_n^d\right)$. Introduce a new variable y_i for each ith gate of circuit. We can write equations describing the computation. e.g

$$y_i = y_j + y_k,\ y_i = y_j + y_k,\ y_i = x_j, \text{ or } y_i = c$$

For the outputs $y_{s-n+1}, \ldots, y_s$, let's also add equations $y_{s-n+1} = 1, \ldots, y_s = 1$. The number of solutions to this is the number of solutions to $x_1^d = x_2^d = \cdots = x_n^d = 1$ since our system of equations effectively computes those values, and our constraints set them equal to 1. The number of solutions is d^n, if our field is algebraically closed (like $\mathbb{C}$), which we can assume without loss of generality, since every field is contained in an algebraically closed one and an algebraic circuits can always be evaluated over extension fields.

Bezout's Theorem from algebraic geometry states that if $f_1(y_1, \ldots, y_s), \ldots, f_t(y_1, \ldots, y_s)$ are polynomials, the number of solutions to $f_1(y) = \cdots = f_t(y) = 0$ over an algebraically closed field is either infinite or at most $\Pi_i \deg(f_i)$).

In our case, all equations have degree one except for those corresponding to multiplication gates, which have degree two. Therefore, $d^n \leq 2^{\#\text{mult gates}}$. Taking the logarithm of both sides gives us

$$n \log d \leq \#\text{mult gates}$$

as desired. ■

These two lemmas imply the theorem because $\frac{\partial}{\partial x_i}\left(x_1^d + x_2^d + \cdots + x_n^d\right) = dx_i^{d-1}$. We can divide out by d if $\text{char}(\mathbb{F}) \nmid d$. So we can compute $(x_1^{d-1}, \ldots, x_n^{d-1})$ with $5s$ gates. Thus, $5s \geq n \log d$. ■

3 Overview of Algebraic Circuit Lower Bounds

Not much is known for the general class of algebraic circuits, but more lower bounds are known if we restrict the model.

- $\Theta(n \log d)$ is the best lower bound known for general algebraic circuits computing a function in **AlgNP/poly**.
- Most polynomials of deg d in n variables require algebraic circuits of size $\Omega\left(\binom{n+d+1}{d+1}\right)$. This is approximately how many coefficients are needed, and is analogous to the idea that most binary functions require as many gates as the size of the truth table.

- In the Bounded Coefficients Model (all constants from $\mathbb{C}$ have magnitude at most 1): the Discrete Fourier Transform requires size $\Omega(n \log n)$, which is tight.
- In the Monotone Arithmetic Circuits Model (all constants are nonnegative): Only monotone polynomials (where all coefficients are nonnegative) are computable. Perm requires size $2^{\Omega(n)}$.
- In Multilinear Circuits: Perm and Det require size $n^{\Omega(\log n)}$.
- Constant Depth Algebraic Circuits (with unbounded fan-in on $+, \times$)
  - Depth 2: Exponential lower bounds are easy.
  - Depth d: We have lower bounds $2^{\Omega\left(n^{1/d}\right)}$ over $\mathbb{F} = \mathbb{Z}_2$.
  - Depth 3: Lower bounds of $\Omega\left(n^2\right)$ are known over larger fields.
  - Depth $O(1)$: Slightly superlinear lower bounds are known for polynomials of constant degree.
  - $n^{\Omega(d)}$ lower bound for depth 4 $\implies$ $n^{\Omega(d)}$ lower bound for general arithmetic circuits.

There are also algebraic computation models for decision problems: allow branching on equality and/or inequality. More on this in the Arora–Barak text.

4 Overview of Boolean Circuit Lower Bounds

We can also give an overview of boolean circuits with respect to various restricted models.

- For general circuits, the best known lower bound is $5n - o(n)$ over $(\wedge, \vee, \neg)$.
- For boolean formulas, we have lower bounds of the form $\Omega\left(n^2\right)$. This is the lower bound for Majority.
- For monotone circuits (no negation): $2^{n^{\Omega(1)}}$. This is proved for Clique and Perfect Matching.
- Constant depth circuits (with unbounded fan-in on $\wedge$, $\vee$).
  - Depth d: $2^{\Omega\left(n^{1/d}\right)}$. This is once again the lower bound for Majority, and also Parity. Holds for majority even if we allow unbounded fan-in $\oplus$. Note that proving lower bounds with $\oplus$ also gives lower bounds for algebraic circuits over the field $\mathbb{F}_2$, which is how the lower bounds mentioned earlier are obtained.

There are also Time-Space tradeoffs

- We have explicit problems that cannot be solved simultaneously in time $O(n)$ and space polylog (n) even nonuniformly (this amounts to length vs log (width) lower bounds for branching programs).