Lower Bounds for Algebraic Circuits

Notes from a series of lectures by

Ran Raz
The Weizmann Institute of Science

The 2005 Barbados Workshop on Computational Complexity
February 27th – March 6th, 2005

Organizer:
Denis Thérien

Scribes:
Arkadev Chattopadhyay
Faith Ellen Fich
Anna Gál
Ricard Gavaldà
Michal Koucký
Benoit Larose
Dana Randall
Pascal Tesson

Date:
August 23rd, 2005

1 Introduction

We consider circuits over a field F, with inputs $x_1, \ldots, x_n$ and $+, \times$ gates. The indegree of gates is always 2, except for bounded depth circuits, for which the indegree is unbounded. $F[x_1, \ldots, x_n]$ is the ring of polynomials with coefficients in F. Sometimes there will be one output, other times there will be many.

The size of a circuit is defined to be the number of edges it contains. If the indegree of gates is 2, then the number of edges is twice the number of gates.

F is a field, usually $GF[p], \mathbb{Q}, \mathbb{R}$, or $\mathbb{C}$. Sometimes it is any algebraically closed field.

Syntactic computation of a polynomial $f(x_1, .., x_n)$ by a circuit must produce exactly the polynomial $f(x_1, .., x_n)$ when the computation is done over $F[x_1, \ldots, x_n]$. Semantic computation of a polynomial $f(x_1, .., x_n)$ by a circuit produces a polynomial $f'(x_1, .., x_n)$ that produces the same result for all choices of inputs from F, i.e. $f'(a_1, \ldots, a_n) = f'(a_1, \ldots, a_n)$ for all $a_1, \ldots, a_n \in F$. For example, semantic computation of x^p over $GF[p][x]$ can be done by a circuit that computes x, since $x^p = x$ for all $x \in GF[p]$. Over $\mathbb{C}$ (or any field of characteristic 0), semantic and syntactic computation are the same.

Polynomials of large degree cannot be computed by circuits of small depth and small fan-in. For example, x^{2^s} cannot be computed by a fan-in 2 circuit of depth less than s. The proof is by induction, using the fact that the degree of the output of a gate can be at most double the degree of its inputs. Thus, the problem of interest is to prove lower bounds for polynomials of smaller degree.

2 Notation

- n usually denotes the number of input variables (although sometimes the number of input variables will be n^2 when the input is a matrix)
- s usually denotes the size of the circuit
- d usually denotes the depth of the circuit
- r usually denotes the degree of the polynomial computed by the circuit

In this series of lectures we will not care about uniformity of circuits. The complexity classes we will mention most often are:

- P denotes the set of polynomials that can be computed by circuits with $s, r \in poly(n)$
- NC_k denotes the set of polynomials that can be computed by circuits with $s, r \in poly(n)$ and $d \in O(\log^k n)$.
- NP denotes the set of polynomials $f(x_1, \ldots, x_n) = \sum\{g(x_1, \ldots, x_n, e_1, \ldots, e_m) \mid e_1, \ldots, e_m \in \{0,1\}\}$ where $g \in P$ and $m \in poly(n)$. The functions in arithmetic NP are also called P-expressible.

The permanent is complete for NP [18]: If $f(x_1, \ldots, x_n)$ is in NP, then there exists a matrix $Y = (y_{ij})$ such that $y_{ij} \in \{x_1, \ldots, x_n\} \cup F$ and $perm(Y) = f(x_1, \ldots, x_n)$.

A central result in this theory is:

Theorem 1 (Valiant, Skyum, Berkowitz, and Rackoff [19]). *In arithmetic circuits, $P = NC_2$. More generally, given an arithmetic circuit with size s and degree r, there is an arithmetic circuit that computes the same function with size $s' \in poly(s, r)$ and depth $d' \in O((\log s)(\log r) + (\log r)^2)$.*

A formula is a circuit in which there is at most one path between any two gates. The size of a formula is in $2^O(d)$, when the fan-in is 2.

We do not consider subtraction gates, since we can istead use multiplication by -1 and an addition gates.

We usually do not consider division gates, because the following theorem shows that they do not give us much benefit either.

Theorem 2 (Strassen [16]). *If a polynomial function can be computed by a circuit of size s using $+, \times, \div$, then there is a circuit of size $poly(s, n, r)$ using only $+, \times$ that computes the same polynomial.*

Note that Strassen's construction may increase the depth of the circuit by a factor of $\log n$. Also, this theorem might only hold for infinite fields. If we have a circuit over a finite field F, then there is a large enough extension of F over which the same approach works.

The idea is to replace $1/g$ by its power series expansion $1 + g + g^2 + g^3 + \cdots$ but to truncate the high powers of g (those bigger than r).

3 Important Functions

- Permanent of a matrix (x_{ij}):
  $Perm(x_{ij}) = \sum\{x_{1,\sigma(1)} \cdots x_{n,\sigma(n)} \mid \sigma$ is a permutation of $\{1, \ldots, n\}\}$. As mentioned before, the permanent is a complete problem for NP.

- Determinant of a matrix (x_{ij}):
  $Det(x_{ij}) = \sum\{sign(\sigma)\, x_{1,\sigma(1)} \cdots x_{n,\sigma(n)} \mid \sigma$ is a permutation of $\{1, \ldots, n\}\}$. Determinant is "universal" (i.e. hard) for formula size: If there is a polynomial $f(x_1, \ldots, x_n)$ with formula size s, then there is a matrix $Y = (y_{ij})$ of dimension at most $3s$ such that $y_{ij} \in \{x_1, \ldots, x_n\} \cup F$ and $Det(Y) = f(x_1, \ldots, x_n)$.

- Product of two $n \times n$ Matrices (x_{ij}) and (y_{ij}):
  There are $2n^2$ input variables and n^2 output variables. This can be computed by a circuit of size $O(n^{2.34\cdots})$.

- Product of n matrices, each of dimension n:
  There are n^3 input variables and n^2 output variables.

- Evaluating Symmetric polynomials:
  for example, majority, threshold k, parity.

- Evaluating Linear Functions:
  $f(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i x_i$ where $a_0, \ldots, a_n \in F$. If M is an $n \times n$ matrix over F and $x = (x_1, \ldots, x_n)$, then Mx is a linear function. In general, the circuit size of a matrix vector product is $O(n^2)$. For special matrices such as the Hadamard matrix H and the Discrete Fourier Transform matrix, the circuit size is $O(n \log n)$.

- Product of Two Polynomials:
  $(\sum_{i=1}^{n} x_i t^i) \times (\sum_{i=1}^{n} y_i t^i)$ Here the inputs x_i and y_i are coefficients of univariate polynomials with formal variable t.

4 Homogenization and Multilinearization

A polynomial is *homogeneous* if all its monomials have the same degree.

We will use often the following property, or variations of it: Suppose that $f(x_1, \ldots, x_n)$ is homogeneous of degree r and is computed by a circuit of size s and depth d. Then there is a circuit of size $O(sr^2)$ and depth $O(d \log r)$ in which all gates compute homogeneous functions.

The idea is to split each gate v that computes a function f_v of degree r_v into at most $r_v + 1$ nodes, each of which computes the terms of f_v of one degree. For example, a gate that computes the product of $(2 + x_1)$ and $(3 + x_2)$ would be replaced by 3 gates, one that computes the product of 2 and 3, one that computes $2x_2 + 3x_1$, and one that computes the product of x_1 and x_2. To compute $2x_2 + 3x_1$, two new intermediate gates are used, one that computes the product of 2 and x_2 and the other that computes the product of x_1 and 3.

Note that we can assume that $r_v \leq r$, since terms of degree greater than r can be thrown away.

A polynomial on two sets of variables $\{x_1, \ldots, x_n\}$ and $\{y_1, \ldots, y_n\}$ is *bilinear* if all of its monomials contain at most one variable from each of these sets. For example, $x_1y_3 + x_2y_3$ is bilinear.

A bilinear circuit is a homogeneous circuit such that the function computed at every node has degree at most 1 in $\{x_1, \ldots, x_n\}$ and degree at most 1 in $\{y_1, \ldots, y_n\}$. All monomials that contain more than one x variable or more than y variable can be thrown away.

In a bilinear circuit, we may assume that each product gate has one input that is a linear functions of only x variables and one input that is a linear functions of only y variables. To bilinearize a circuit that computes a bilinear function, linear functions of x variables, linear function functions of y variables, and functions all of whose monomials contain one x variable and one y variable are computed separately. The cost is only a constant factor overhead in size and depth.

More in general, suppose that $S_1, \ldots, S_m$ are disjoint sets of variables. A polynomial on $S_1 \cup \cdots \cup S_m$ is multilinear with respect to this partition if each of its monomials contains at most one variable from each set. In a multilinear circuit, the function computed at every node is multilinear. The overhead for making a circuit multilinear is $O(2^m)$: For each subset I of $\{1, \ldots, m\}$, separately compute the monomials that contain exactly one variable in S_i for each $i \in I$.

5 A Survey of Known Lower Bounds on Circuit Size for Arithmetic Circuits

- For syntactic computation (but no other restrictions), Strassen proved an $\Omega(n \log r)$ lower bound, when $r \in n^{\Theta}(1)$, for computing n polynomials simultaneously [15]. Baur and Strassen improved this result so that it holds for computing only 1 polynomial [2].

  If the degree $r \in O(1)$, then the lower bound is $\Omega(n)$, which is not exciting.

- A counting argument shows that there are n input n output linear functions that require circuit size about $\Omega(n^2)$.

- For constant depth circuits over $GF[2]$, Razborov and Smolensky proved exponential lower bounds, in the semantic model. This follows from their well-known lower bounds in the Boolean circuit model with $\vee$ and $\wedge$ replaced by $+$ and $\times$.

- For depth 2 circuits over any field, exponential lower bounds are easy, because the circuits produce only sums of monomials or products of linear functions.

- For depth 3 circuits, Shpilka and Wigderson [14] have proved $\Omega(n^2)$ lower bounds.

- Pudlak [10] proved slightly superlinear lower bounds for constant degree polynomials computed by constant depth circuits. For depth 3, his bounds are $\Omega(n \log n)$, for depth 4, they are $\Omega(n \log \log n)$, and for depth 5, they are $\Omega(n \log^* n)$.

- Monotone circuits over R allow $+$, $\times$, but do not allow negative constants. Lots is known about these circuits, because results about monotone Boolean circuits translate over.

- The bounded coefficient model allows $+$, $\times$, but does not allow multiplication by field elements of absolute value greater than 1, nor addition of two field elements. In this model, Morgenstern [6] proved an $\Omega(n \log n)$ lower bound for the product of a Hadamard matrix times a vector over $\mathbb{R}$ or $\mathbb{C}$. This problem is an example of computing n constant degree polynomials.

- In a noncommutative ring of polynomials, variables do not commute. For example, $x_1x_2 \neq x_2x_1$. The proof that $P = NC_2$ doesn't work in this model.

  Noam Nisan [7] proved superpolynomial lower bounds on formula size for multilinear functions computed by multilinear formulas.

- There are some other very restricted results.

6 Strassen's $\Omega(n \log n)$ Lower Bound

Strassen [15] proved the following lower bound for computing n polynomials, each of degree at most n. It works over any field F, and can be extended to allow division gates. We will assume that F is algebraically closed. But this is good enough, since a lower bound over F implies the same lower bound over any field that is contained in F.

The lower bound is given for explicit polynomials:

$$h_i(x_1, \ldots, x_n) = \sum_{j_1 < \cdots < j_i} x_{j_1} \cdots x_{j_i}.$$

The polynomial h_i is called the symmetric polynomial of degree i. For example $h_1(x_1, \ldots, x_n) = \sum_{j=1}^{n} x_j$ and $h_n(x_1, \ldots, x_n) = x_1 \cdots x_n$.

The proof relies on a slightly restricted version of Bézout's Theorem:

Bézout's Theorem: Suppose we have k polynomials over an algebraically closed field F. Say $f_i(x_1, \ldots, x_n)$ has degree r_i, for $i = 1, \ldots, k$. Let $S \subseteq F^n$ denote the set of common zeros of these polynomials. Then S is either infinite or $|S| \leq \prod_{i=1}^{k} r_i$. If the polynomials are algebraically independent and if multiplicities are counted correctly, then the inequality becomes an equality.

In algebraic geometry, this theorem can be stated as

$$degree(A \cap B) \leq degree(A) \cdot degree(B),$$

where A and B are algebraic varieties and

$$degree(A) = \max\{|A \cap V| \ : \ V \text{ is an affine subspace and } |A \cap V| \text{ is finite}\}.$$

The common zeros of a set of polynomials is an algebraic variety.

Let us prove a lower bound on the size of any circuit computing all symmetric polynomials $h_1, \ldots, h_n$. Consider the family of equations E

$$\begin{aligned} h_1(x_1, \ldots, x_n) &= b_1 \\ &\vdots \\ h_n(x_1, \ldots, x_n) &= b_n. \end{aligned}$$

Then, by Bezout's Theorem, the number of solutions is at most $\prod_{i=1}^{n} i = n!$ or infinite. We'll show that there exist $b_1, \ldots, b_n$ such that the number of solutions is exactly $n!$. Fix some such choice for $b_1, \ldots, b_n$.

Suppose we have a circuit C with only $+, \times$ gates that computes $h_1, \ldots, h_n$. For every node v in C, including leaves and outputs, define a new variable y_v.

Consider the following set E' of polynomial equations over $x_1, \ldots, x_n, \{y_v\}_{v \in C}$:

- If v is a leaf labelled by a constant a, then write the equation $y_v = a$.
- If v is a leaf labelled by a variable x_j, then write the equation $y_v = x_j$.
- If v is a $+$ gate with inputs from nodes v_1 and v_2, then write the equation $y_v = y_{v_1} + y_{v_2}$.
- If v is a $\times$ gate with inputs from nodes v_1 and v_2, then write the equation $y_v = y_{v_1} \times y_{v_2}$.
- If v is the i'th output, then write the equation $y_v = b_i$.

Note that all of these equations are of degree 1, except the equations for the $\times$ gates, which have degree 2.

For every solution $a_1, \ldots, a_n$ of E, substituting $x_1 = a_1, \ldots, x_n = a_n$ into the circuit satisfies all the equations in E'. Similarly, every solution of E' gives a solution of E. Thus the number of solutions of E' and the number of solutions of E are the same.

Let q be the number of product gates in the circuit. By Bézout's Theorem, the number of solutions of E' is at most 2^q. Since the number of solutions of E is $n!$, it follows that $q \geq \log n! \in \Omega(n \log n)$.

Next, we prove that there are choices for $b_1, \ldots, b_n$ for which E has $n!$ solutions. Consider the polynomial

$$\prod_{i=1}^{n}(t - x_i) = 1 \cdot t^n - h_1(x_1, \ldots, x_n) \cdot t^{n-1} + \cdots + (-1)^n h_n(x_1, \ldots, x_n) \cdot t^0$$

in $(F[x_1, \ldots, x_n])[t]$. Let $a_1, \ldots, a_n$ be distinct elements of F. Let $b_i = h_i(a_1, \ldots, a_n)$ for $i = 1, \ldots, n$.

We claim that the number of solutions of E is $n!$. Because all of the polynomials are symmetric, every permutation of $a_1, \ldots, a_n$ is a solution of E, so there are at least $n!$ solutions.

We must show that there are not an infinite number of solutions. It suffices to show that every solution $a'_1, \ldots, a'_n$ of E is a permutation of $a_1, \ldots, a_n$.

Consider $g(t) = t^n - b_1 t^{n-1} + \cdots + (-1)^n b_n \cdot t^0$. Substituting $a'_1, \ldots, a'_n$ for $x_1, \ldots, x_n$ in $\prod_{i=1}^{n}(t - x_i)$ gives $g(t)$. So $\prod_{i=1}^{n}(t - a'_i) = g(t) = \prod_{i=1}^{n}(t - a_i)$. Polynomials over a field factor uniquely. Thus $a'_1, \ldots, a'_n$ is a permutation of $a_1, \ldots, a_n$.

This concludes the proof that computing all symmetric polynomials on n variables requires size $\Omega(n \log n)$.

A lower bound is easier to prove in the following case (although it is still interesting to be able to prove it for the symmetric polynomials).

The set of polynomial equations

$$\begin{aligned} x_1^n &= 1 \\ &\vdots \\ x_n^n &= 1 \end{aligned}$$

has exactly n^n solutions, namely the n'th roots of unity. Thus, any arithmetic circuit that computes $x_1^n, \ldots, x_n^n$ $\Omega(n \log n)$ gates. This is a simple example in which the Direct Sum Conjecture is true.

To extend this proof to allow division, we proceed as follows: If v is a $\div$ gate with inputs from nodes v_1 and v_2, then write the equation $y_{v_2} = y_v \times y_{v_1}$. This equation also has degree 2, so our lower bound counts the number of $\times, \div$ gates. There is a minor technical point to ensure that no solution of the resulting set of polynomials E' is a zero of y_{v_2}.

7 The Baur-Strassen Lower Bound

In this section we present the lower bound by Baur and Strassen [2] that we mentioned in the previous lecture. Interestingly, the lower bound follows from this surprising upper bound:

Theorem 3. *For every function $f(x_1, \ldots, x_n)$ computable by a circuit of size s and depth d there is a circuit of size $5s$ and depth $3d$ computing* all *of*

$$f,\ \frac{\partial f}{\partial x_1}, \ldots,\ \frac{\partial f}{\partial x_n}.$$

Proof. We will use the Chain Rule for partial derivatives: If

$$f(x_1, \ldots, x_n) = h(x_1, \ldots, x_n, g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$$

then

$$\frac{\partial f}{\partial x_i} = \frac{\partial h}{\partial x_i} + \sum_{i=1}^{m} \frac{\partial h}{\partial g_i} \cdot \frac{\partial g_i}{\partial x_i},$$

where $\partial h/\partial g_i$ stands for the result of substituting each occurrence of y in $\partial h/\partial y$ with $g_i(x_1, \ldots, x_n)$.

Take a circuit C computing f, of size s and depth d. Take a deepest gate v in C. It is connected to two leaves, so it computes a function f_v which is one of $x_i + x_j$, $x_i * x_j$, $x_i + c$, or $x_i * c$, and there exists $h(x_1, \ldots, x_n, y_v)$ such that

$$f(x_1, \ldots, x_n) = h(x_1, \ldots, x_n, f_v(x_i, x_j)).$$

Then by the chain rule, for every k

$$\frac{\partial f}{\partial x_k} = \frac{\partial h}{\partial x_k} + \frac{\partial h}{\partial f_v} * \frac{\partial f_v}{\partial x_k},$$

and the second term is 0 if $k \neq i, j$.

Now: since f is computed by C (of size s), $h(x_1, \ldots, x_n, y_v)$ is computed by the circuit (of size $s-1$) obtained by replacing gate v with variable y_v in C.

Then, by induction, there is a circuit C_v of size $5(s-1) = 5s-5$ computing $h(x_1, \ldots, x_n, y_v)$ and all its partial derivatives. We show how to use C_v and 5 additional gates to compute f and all its partial derivatives.

Function $f(x_1, \ldots, x_n)$ is computed by $C_v(x_1, \ldots, x_n, f_v(x_i, x_j))$, which requires only the additional gate to compute f_v.

For $k \neq i, j$ we have $\partial f/\partial x_k = \partial h/\partial x_k$, which is already one of the outputs of C_v, so no additional gates are required.

To compute $\partial f/\partial x_i$, observe that

$$\frac{\partial f}{\partial x_i} = \frac{\partial h}{\partial x_i} + \frac{\partial h}{\partial f_v} \cdot \frac{\partial f_v}{\partial x_k}$$

But

- $\partial h/\partial x_i$ is one of the outputs of C_v; no gate needed.
- $\partial h/\partial f_v$ is obtained by plugging $f_v(x_i, x_j)$ (which we just computed) into the y_v input of $\partial h/\partial y_v$ (which is one of the outputs of C_v); no gate needed.
- $\partial f_v/\partial x_i$ is either a constant or the variable x_j, depending on f_v; no gates needed either.

So we only need one $\times$ and one $+$ gate to compute it. Two more gates may be necessary to compute $\partial f/\partial x_j$, hence a total of 5 gates as promised.

It is also easy to argue inductively that if C has depth d, then the resulting circuit has depth $3d$. $\square$

Now to prove the lower bound: Let $f(x_1, \ldots, x_n, y_1, \ldots, y_n)$ be $\sum_{i=1}^{n} y_i \cdot x_i^n$. We claim that any circuit for f has size $\Omega(n \log n)$.

Indeed, take any circuit C for f. By the previous theorem, there is a circuit C' with 5 times as many gates as C computing f and all its partial derivatives, among which there are $x_1^n, \ldots, x_n^n$. At the end of Session 1 we proved that this requires circuits of size $\Omega(n \log n)$.

8 Towards Stronger Lower Bounds

Consider now functions of constant degree r. The Strassen and Baur-Strassen lower bounds let us show lower bounds $\Omega(n \log r)$. Can we do any better?

Next we will review three classical approaches to this problem:

- Tensor rank
- Direct sum conjecture
- Valiant's size/depth tradeoff

In Session 6 we will cover in more detail the second and third approaches.

9 Approach 1: Tensor Rank and Lower Bounds

We first recall the definition of tensor rank by analogy with matrix rank.

Let $T = \{T_{ij}\}$ be an $n \times n$ matrix, i.e., a function $[n]^2 \to F$. We say that T has rank at most 1 if there are $a_1, \ldots, a_n, b_1, \ldots, b_n$ such that $T_{ij} = a_i \cdot b_j$. Then the rank of T is the minimum r such that there are matrices $T_1, \ldots, T_r$, all of rank 1, with $T = \sum_{i=1}^{r} T_i$. By a probabilistic argument, one can show that most matrices T have rank n.

Similarly, let $T = \{T_{ijk}\}$ be a tensor, i.e., a function $[n]^3 \to F$. We say that T has rank at most 1 if there are $a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n$ such that $T_{ijk} = a_i \cdot b_j \cdot c_k$. Then the rank of T is the minimum r such that there are matrices $T_1, \ldots, T_r$, all of rank 1, with $T = \sum_{i=1}^{r} T_i$

Again by a probabilistic argument, one can show that a random tensor will have rank $c\, n^3/3n \approx c\, n^2$. But we do not know of any *explicit* tensor of rank greater than n. (To construct a tensor or rank n, simply take a rank n matrix and expand it to a tensor).

The problem of building explicit tensors of high rank is interesting because of the following: if we have an explicit tensor or rank R, then we have a circuit size lower bound of R, for an explicit function. Here is why:

Let T be the explicit tensor of rank R. Define functions $f_1, \ldots, f_n$ on $x_1, \ldots, x_n, y_1, \ldots, y_n$, by

$$f_k(x_1, \ldots, x_n, y_1, \ldots, y_n) = \sum_{k=1}^{n} T_{ijk} x_i y_j.$$

Note that these functions are bilinear in the x_i and y_i. Hence, optimal circuits for the f_k are bilinear by the tricks similar to homogenization described in Session 1.

Any bilinear circuit to compute the f_k necessarily will have this structure:

1) Compute a number of products, say s, of the form

$$\left(\sum_{i=1}^{n} a_i^1 x_i\right) \times \left(\sum_{i=1}^{n} b_i^1 y_i\right), \ldots, \left(\sum_{i=1}^{n} a_i^s x_i\right) \times \left(\sum_{i=1}^{n} b_i^s y_i\right).$$

Call them functions $h_1, \ldots, h_s$.

2) Compute

$$f_k = c_k^1 \, h_1 + \cdots + c_k^s \, h_s. \tag{1}$$

The coefficient of $x_i y_j$ is, by definition, T_{ijk} and, by (1), $\sum_{v=1}^n c_k^v a_i^v b_j^v$. So if we can do 1)+2) with some s, then T has rank at most s. In other words, if T has rank R, the size of any circuit computing $f_1, \ldots, f_n$ is at least R.

[At this point, Mario and Ran discuss whether theorems on the eigenvalues of tensors might help in proving high rank. Pavel points out that it will be hard to prove such theorems for tensors, since Tensor Rank is NP-complete [5], while Matrix Rank is in P.]

10 Approach 2: Direct Sum Conjectures and Lower Bounds

Suppose we know that $f(x_1, \ldots, x_m)$ has circuits of size m^2. Now we want to compute f on two distinct instances, $f(x_1, \ldots, x_m), \ldots, f(x_{m+1}, \ldots, x_{2m})$. It is not clear that one can share any computation at all among both tasks, but can we actually prove that we need size $2m^2$?

If not, can we prove that we need size $1.1m^2$?

... at least for particularly hard fs? (we then say that Direct Sum holds for f)

... say, for a random f?

Nonexplicitly, one can show that a random quadratic polynomial needs circuits of size n^2. Let m be $\sqrt{n}$ and define

$$f(x_1, \ldots, x_m, y_1, \ldots, y_{m^2}) + \sum_{i,j=1}^{m} y_{ij} \; x_i x_j.$$

Setting the y_{ij} variables randomly gives a random degree 2 polynomial. Fix that random polynomial p. Clearly p (on $2m$ variables) can be computed in size $O(m^2) = O(n)$.

Consider the function that applies p to $\sqrt{n}$ sets of distinct variables. If Direct Sum holds for p, then essentially no computation can be shared among these $\sqrt{n}$ copies, so we have a function of complexity about $n^{1.5}$.

11 Approach 3: Valiant's method

Valiant's goal is to design *linear* functions $f_1, \ldots, f_n$ such that any circuit computing all of the f_i requires *either* superlinear size *or* superlogarithmic depth.

For this purpose, he introduced the notion of *em matrix rigidity.* Say that matrix E is ϵ-sparse if in each row there are at most n^ϵ nonzero entries. Matrix A is rigid if for every ϵ-sparse matrix E,

$$rank(A+E) \geq c\, n / \log\log n$$

(where $c = c(\epsilon)$).

Valiant shows [17] that an explicit construction of rigid matrices yields lower bounds of the type above (either size or depth). Coming up with such a construction looks at first sight like a simple problem, but it is not; Ran admits having thought on it for many many hours.

Valiant's argument will be presented in Session 6.

12 Introduction

In this lecture we prove lower bounds for so-called *non-commutative* arithmetic circuits, due to N. Nisan [7].

13 Preliminaries

Let $\mathbb{F}$ be a field, let $\mathbb{F}\{x_1, \ldots, x_n\}$ denote the non-commutative algebra consisting of all linear combinations over $\mathbb{F}$ of 1 and monomials $x_{i_1}x_{i_2}\cdots x_{i_k}$, $1 \leq i_j \leq n$. In other words, this is the usual ring of polynomials in $x_1, \ldots, x_n$ over $\mathbb{F}$ except that the identities $x_i x_j = x_j x_i$ do not hold; for instance, $x_1 x_2$ and $x_2 x_1$ are distinct monomials, so that

$$(x_1 + x_2)(x_1 - x_2) = x_1^2 - x_1x_2 + x_2x_1 - x_2^2 \neq x_1^2 - x_2^2.$$

The *degree* of a monomial $x_{i_1}x_{i_2}\cdots x_{i_k}$ is k, and we say that $f(x_1, \ldots, x_n)$ is *homogeneous* if all its monomials have the same degree.

The determinant and permanent need to be (re)defined in this non-commutative setting:

Definition 1. *The* determinant *of the $n \times n$ matrix (x_{ij}) is defined by*

$$Det\{x_{ij}\} = \sum_{\sigma \in S_n} (sgn\,\sigma)\, x_{1\sigma(1)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}.$$

The permanent *is defined by*

$$Per\{x_{ij}\} = \sum_{\sigma \in S_n} x_{1\sigma(1)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}.$$

We also redefine the notion of partial derivative; for convenience let $\overline{x} = (x_1, \ldots, x_n)$. Given any $f \in \mathbb{F}\{x_1, \ldots, x_n\}$, we may express it in the form

$$f(\overline{x}) = \sum_{i=1}^{n} x_i f_i(\overline{x})$$

where the f_i are unique. We then define, for each $1 \leq i \leq n$,

$$\frac{\partial f}{\partial x_i} = f_i(\overline{x}).$$

For instance, if

$$f(x_1, x_2, x_3, x_4) = x_1^2x_3 - 5x_2x_3^5x_4^3x_1 + x_1^3x_2x_1 + 2x_3x_2$$

then we may rewrite it as

$$f(x_1, x_2, x_3, x_4) = x_1(x_1x_3 + x_1^2x_2x_1) + x_2(-5x_3^5x_4^3x_1) + x_3(2x_2) + x_4(0).$$

and hence

$$\tfrac{\partial f}{\partial x_1} = x_1x_3 + x_1^2x_2x_1, \quad \tfrac{\partial f}{\partial x_2} = -5x_3^5x_4^3x_1, \quad \tfrac{\partial f}{\partial x_3} = 2x_2, \quad \tfrac{\partial f}{\partial x_4} = 0.$$

We may then define higher derivatives in the obvious way, i.e.

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = \frac{\partial}{\partial x_i}\left(\frac{\partial f}{\partial x_j}\right)$$

and so on. In the previous example, we obtain

$$\frac{\partial^2 f}{\partial x_1^2} = x_3 + x_1 x_2 x_1, \quad \frac{\partial^2 f}{\partial x_3 \partial x_2} = -5x_3^4 x_4^3 x_1, \quad \frac{\partial^2 f}{\partial x_2 \partial x_3} = 2$$

and

$$\frac{\partial^2 f}{\partial x_i \partial x_j} = 0$$

for all other $1 \leq i, j \leq 4$.

14 Arithmetic Branching Programs (APB's)

Definition 2. *An* Arithmetic Branching Program *(ABP) is a directed acyclic graph with a source s and a sink t, with the vertices of the graph partitioned in levels numbered $0, \ldots, d$ such that every edge goes from level i to level $i+1$ for some $0 \leq i \leq d-1$. The source is the only vertex at level 0 and the sink is the only vertex at level d. Each edge is labelled by a linear function, i.e. a function l of the form $l(x_1, \ldots, x_n) = \sum_{i=1}^n a_i x_i + a_0$ where $a_0, \ldots, a_n \in \mathbb{F}$.*

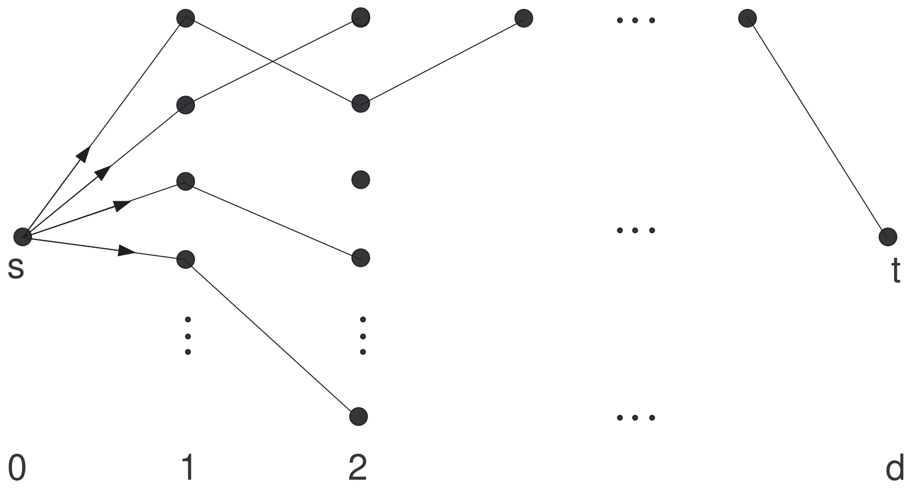


Figure 1: An arithmetic branching program (APB)

We use the following notation to denote the element of $\mathbb{F}(x_1, \ldots, x_n)$ computed by the APB: let

$$A_{st}(\overline{x}) = \sum l_1(\overline{x}) l_2(\overline{x}) \cdots l_k(\overline{x})$$

where $l_1, l_2, \ldots, l_k$ are the labels of the edges of a path $e_1, \ldots, e_k$ from s to t and the sum is taken over all paths from s to t. More generally, for any nodes u and v in the APB with $level(u) \leq level(v)$ let

$$A_{uv}(\overline{x}) = \sum l_1(\overline{x}) l_2(\overline{x}) \cdots l_k(\overline{x})$$

where the sum is taken over all paths from u to v. Notice that if $v_1, \ldots, v_m$ denote all the distinct vertices of a certain level then

$$A_{st}(\overline{x}) = \sum_{i=1}^{m} A_{sv_i}(\overline{x}) A_{v_i t}(\overline{x}).$$

We say that node v *computes* $A_{sv}(\overline{x})$.

In order to obtain a lower bound on the formula size for $f(\overline{x}) \in \mathbb{F}\{x_1, \ldots, x_n\}$ it will suffice to obtain it for APB's: indeed, if f has formula size S then we can obtain an APB of size polynomial in S for $f(\overline{x})$. The straightforward simulation goes as follows: if C consists of a leaf x_i the corresponding branching program has one edge from s to t labelled by the linear function $l(\overline{x}) = x_i$; similarly if the leaf is c with $c \in \mathbb{F}$ we label the edge of the APB by $l(\overline{x}) = c$. Now suppose we have a sum gate $f + g$ where f and g are the inputs. Given APB's for f and g, construct the APB for $f + g$ as in Figure 2; if f and g do not have the same depth, then some padding will be necessary to ensure that the resulting APB has the right level structure. The edges shown are all labelled by 1.

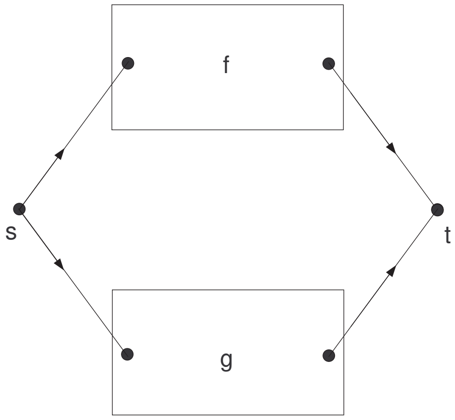


Figure 2: An APB for a sum gate.

If we have a product gate with inputs f and g; given APB's for f and g we construct an APB for fg as in Figure 3.

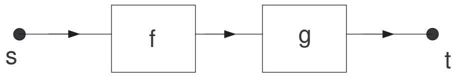


Figure 3: An APB for a product gate.

Given an APB, we make the program homogeneous as follows: if node v computes $A_{sv}(\overline{x})$ of degree r, split the node into nodes $v_0, \ldots, v_{r+1}$ where v_i computes the homogeneous part of $A_{sv}(\overline{x})$ of degree i. We then obtain an APB such that, if v is a node in level i then A_{sv} computes a homogeneous polynomial of degree i. [1]

[1] Alternatively: if one defines an APB to be labelled only by linear functions $l(\overline{x}) = a_1x_1 + \ldots a_nx_n$, then APB's can only compute homogeneous polynomials. To simulate more general polynomials one simply allows each homogeneous component to be computed by a different APB.

Proposition 1. *Let $f(\overline{x})$ be computed by a (homogeneous) APB of size S. Then the dimension of the subspace of $\mathbb{F}\{x_1,\ldots,x_n\}$ spanned by the set of all partial derivatives*

$$\frac{\partial^k f}{\partial x_{i_1}\cdots\partial x_{i_k}}$$

(for all k and all $i_1,\ldots,i_k$) is at most S.

Proof. Let $v_1,\ldots,v_m$ denote the nodes at level 1, and let $l_1,\ldots,l_m$ denote the linear functions labelling their respective edges. Then we have that

$$A_{st}(\overline{x})=\sum_{i=1}^{m} l_i(\overline{x})A_{v_i t}(\overline{x}).$$

We claim that for any $1\le j\le m$,

$$\frac{\partial f}{\partial x_j}(\overline{x})=\sum_{i=1}^{m} c_i A_{v_i t}(\overline{x})$$

for some $c_i\in\mathbb{F}$. Indeed, this follows immediately from the definition of partial derivative and the fact that each l_i is homogeneous.

More generally, suppose that $v_1,\ldots,v_m$ are the nodes at level d. Then we know that

$$A_{st}(\overline{x})=\sum_{i=1}^{m} A_{sv_i}(\overline{x})A_{v_i t}(\overline{x}).$$

The corresponding claim is that, for any $1\le j\le m$,

$$\frac{\partial^d f}{\partial x_{i_1}\cdots\partial x_{i_d}}(\overline{x})=\sum_{i=1}^{m} c_i A_{v_i t}(\overline{x})$$

for some $c_i\in\mathbb{F}$. Again, this is clear because the $A_{sv_i}(\overline{x})$ are homogeneous of degree d. Obviously, this result also holds for any d greater than the level of the sink t since the partial derivatives of order d are identically 0. We conclude that the partial derivatives of all orders are linear combinations of the $A_{v_i t}(\overline{x})$; since there are S of these, this completes the proof of the proposition. □

15 The Permanent

We now show that for the permanent, the dimension of the span of all its partial derivatives is large.

Proposition 2. *Let $f(\overline{x})=Per\{x_{ij}\}$, where $1\le i,j\le n$. Then the dimension of the subspace spanned by the partial derivatives of $f(\overline{x})$ is at least 2^n.*

Proof. Let $X=(x_{ij})$. Then for all $1\le d\le n$,

$$\frac{\partial^d Per}{\partial x_{d,i_d}\cdots\partial x_{2,i_2}\partial x_{1,i_1}}(\overline{x})=Per\{Y\}$$

where Y is the minor of X obtained by removing columns $1,\ldots,d$ and rows $i_1,\ldots,i_d$. Indeed, if D denotes the set of all $\sigma\in S_n$ such that $\sigma(1)=i_1,\ldots,\sigma(d)=i_d$ then

$$\begin{aligned}
Per(X) &= \sum_{\sigma\in S_n} x_{1\sigma(1)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}\\
&= \sum_{\sigma\in D} x_{1\sigma(1)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}+\sum_{\sigma\notin D} x_{1\sigma(1)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}\\
&= x_{1,i_1}x_{2,i_2}\cdots x_{d,i_d}\left(\sum_{\tau\in S_{n-d}} x_{d+1\tau(d+1)}\cdots x_{n\tau(n)}\right)+\sum_{\sigma\notin D} x_{1\sigma(a)}x_{2\sigma(2)}\cdots x_{n\sigma(n)}
\end{aligned}$$

and thus

$$\frac{\partial^d Per}{\partial x_{d,i_d} \cdots \partial x_{2,i_2} \partial x_{1,i_1}}(\overline{x}) = \sum_{\tau \in S_{n-d}} x_{d+1\tau(d+1)} \cdots x_{n\tau(n)}$$

which is easily seen to be $Per(Y)$ as stated.

For a fixed $0 \leq d \leq n$, there are $\binom{n}{d}$ minors of X with columns $1, \ldots, d$; and it is immediate that for two distinct minors Y and Y', $Per(Y)$ and $Per(Y')$ are linearly independent because the set of monomials that appear in each are actually disjoint. If Y and Y' have a different number of columns, this clearly holds also, hence all the $Per(Y)$ are linearly independent; since there are

$$\sum_{d=0}^{n} \binom{n}{d} = 2^n$$

different $Per(Y)$, we are done.

□

16 Polynomial size arithmetic circuits have polylogarithmic depth

In this lecture we will show that any arithmetic circuit C of size s and degree r can be transformed into a circuit C' of size poly(s, r) and depth $O(\log s \log r)$ computing the same function. As a corollary we get:

Theorem 4 ([19]). $P = NC^2$.

Proof. Let C be an arithmetic circuit of size s and degree r. We assume that C is homogeneous and minimum. (If not we can transform it into a sum of homogeneous circuits as we have seen before and proceed on each of them separately.) Each gate w of C computes some function $f_w(x_1, \ldots, x_n)$ of degree $\deg w$. We will often denote the function $f_w(x_1, \ldots, x_n)$ by f_w.

Consider two gates v, w of C such that $\deg w < 2 \deg v$. Let $h_{v,w}(x_1, \ldots, x_n, y_v)$ be the function computed at w when the gate v is replaced by a new variable y_v. Since $\deg w < 2 \deg v$, $h_{v,w}(x_1, \ldots, x_n, y_v) = y_v h'(x_1, \ldots, x_n) + h''(x_1, \ldots, x_n)$, i.e., $h_{v,w}$ is linear in y_v. Define $f_{v,w} = \frac{\partial h_{v,w}}{\partial y_v} = h'(x_1, \ldots, x_n)$. Since C is homogeneous, if $\deg v \leq \deg w$ then $\deg f_{v,w} = \deg w - \deg v$.

Observation $f_{w,w} = 1$ for any gate w. If w does not depend on v then $f_{v,w} = 0$. For $w = w' + w''$, $f_{v,w} = f_{v,w'} + f_{v,w''}$, and for $w = w' \times w''$ where $\deg w' \geq \deg w''$, $f_{v,w} = f_{w''} f_{v,w'}$,

Let $a > 0$ be an integer. Define $G_a = \{t;\ t \text{ is a gate } t = t' \times t'' \ \&\ \deg t', \deg t'' \leq a < \deg t\}$.

The following claims will be useful.

Claim 1. *Let w be a gate of C and let $a > 0$ be an integer. If $a < \deg w \leq 2a$ then*

$$f_w = \sum_{t \in G_a} f_t \cdot f_{t,w}.$$

Proof of claim. By induction of the depth of w in the circuit C. The integer a is fixed.

Base case. $w \in G_a$. Since w cannot depend on any $t \in G_a - \{w\}$, $f_{t,w} = 0$ for such t's, so we have

$$f_w = f_w f_{w,w} = \sum_{t \in G_a} f_t \cdot f_{t,w}.$$

We consider two cases in the induction step:

Case 1. $w = w' + w''$. Notice, the induction hypothesis applies to both w' and w'' so

$$f_{w'} = \sum_{t \in G_a} f_t \cdot f_{t,w'} \qquad \& \qquad f_{w''} = \sum_{t \in G_a} f_t \cdot f_{t,w''}.$$

Hence,

$$f_w = f_{w'} + f_{w''} = \sum_{t \in G_a} f_t \cdot (f_{t,w'} + f_{t,w''}) = \sum_{t \in G_a} f_t \cdot f_{t,w}.$$

Case 2. $w = w' \times w''$, where $\deg w' \geq \deg w''$. Clearly, $\deg w' \geq a$ so by induction hypothesis

$$f_{w'} = \sum_{t \in G_a} f_t \cdot f_{t,w'}.$$

Hence,

$$f_w = f_{w'} f_{w''} = \sum_{t \in G_a} f_t \cdot (f_{t,w'} f_{w''}).$$

Clearly, $\deg w < 2 \deg t$ for any $t \in G_a$. Thus, $f_{t,w} = f_{t,w'} f_{w''}$. The claim follows immediately. $\square$

Claim 2. *Let v, w be gates of C and let $a > 0$ be an integer. If $\deg v \leq a < \deg w < 2 \deg v$ then*

$$f_{v,w} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w}.$$

Proof of claim. The proof is similar to the previous one but we provide it for the reader's convenience. It proceeds by induction of the depth of w in the circuit C. The integer a and gate v are fixed.

Base case. $w \in G_a$. Since w cannot depend on any $t \in G_a - \{w\}$ we have

$$f_{v,w} = f_{v,w} f_{w,w} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w}.$$

We consider two cases in the induction step:

Case 1. $w = w' + w''$. Notice, the induction hypothesis applies to both w' and w'' so

$$f_{v,w'} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w'} \qquad \& \qquad f_{v,w''} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w''}.$$

Hence,

$$f_{v,w} = f_{v,w'} + f_{v,w''} = \sum_{t \in G_a} f_{v,t} \cdot (f_{t,w'} + f_{t,w''}) = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w}.$$

Case 2. $w = w' \times w''$, where $\deg w' \geq a > \deg w''$. By induction hypothesis

$$f_{w'} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w'}.$$

Since $f_{t,w} = f_{t,w'} f_{w''}$, we get,

$$f_{v,w} = f_{v,w'} f_{w''} = \sum_{t \in G_a} f_{v,t} \cdot (f_{t,w'} f_{w''}) = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w}.$$

The claim follows. $\square$

Using the established properties of f_w and $f_{v,w}$ we will construct a shallow circuit C' computing the same function as C. The circuit C' will have size $O(s^3)$ and depth $O(\log r \log s)$. The construction of C' proceeds in $\log r$ stages, each stage adds depth $\log s$. At Stage 0 we will compute all f_v and $f_{v,w}$ of degree at most one and at Stage $i+1$ we will compute all f_w and $f_{v,w}$ of degree between $2^i + 1$ and 2^{i+1}, where v and w satisfy $\deg w < 2 \deg v$.

At Stage 0 we need to compute at most $O(s^2)$ linear and constant functions which can be done in depth $O(\log s)$, provided $n \leq s$.

At Stage $i+1$, we first compute all f_w such that $2^i < \deg w \leq 2^{i+1}$. Let $a = 2^i$. By Claim 1,

$$f_w = \sum_{t \in G_a} f_t \cdot f_{t,w} = \sum_{\substack{t = t' \times t'' \\ t \in G_a}} f_{t'} \cdot f_{t''} \cdot f_{t,w}.$$

Each $f_{t'}, f_{t''}, f_{t,w}$ must have already been computed in the previous stage, so we can compute all f_w of given degrees using extra $O(\log s)$ layers of gates.

Next we compute $f_{v,w}$ for every v, w such that $\deg w < 2 \deg v$ and $2^i < \deg w - \deg v = \deg f_{v,w} \leq 2^{i+1}$. Set $a = \deg v + 2^i$. By Claim 2,

$$f_{v,w} = \sum_{t \in G_a} f_{v,t} \cdot f_{t,w}.$$

If $\deg t > \deg w$, then $f_{t,w} = 0$. Hence,

$$f_{v,w} = \sum_{\substack{t=t'\times t'' \\ \deg t'' \leq \deg t' \\ \deg t \leq \deg w \\ t \in G_a}} f_{t''} \cdot f_{v,t'} \cdot f_{t,w}.$$

In this sum, if $f_{v,t'} \neq 0$, then $\deg v \leq \deg t'$. Furthermore, if $\deg t'' > 2^{i+1}$ then $\deg t > \deg v + 2^{i+1} \geq \deg w$, so if $f_{t,w} \neq 0$ then $\deg t'' \leq 2^{i+1}$ and $f_{t''}$ is already computed. Also, $\deg t' < \deg t \leq \deg w < 2 \deg v$, so $\deg t' < 2 \deg v$, $\deg f_{v,t'} = \deg t' - \deg v \leq 2^i$ and $f_{v,t'}$ is also computed. Finally, $\deg v + 2^i < \deg t \leq \deg w \leq \deg v + 2^{i+1}$ implies that $\deg f_{t,w} \leq 2^i$ and $\deg w < 2 \deg t$. Hence, all non-zero values in the sum are already computed and the sum can be evaluated in extra depth $O(\log s)$.

Since all gates w in the minimum circuit C are of degree at most r, the depth of C' is $O(\log r \log s)$ and the size is $O(s^3)$. $\square$

2005 Barbados Workshop on Computational Complexity	
LECTURE NOTES ON LOWER BOUNDS FOR ALGEBRAIC CIRCUITS	
Title: Lower Bounds for Multilinear Formulas	Lecture #: 5
Lecturer: Ran Raz	Scribe: Pascal Tesson

17 Introduction

The best upper bounds known for the size of formulas for the determinant and permanent are $n^{O(\log n)}$ [3] and $O(n^2 2^n)$ respectively but the best known lower bound is still Bauer-Strassen's $\Omega(n \log n)$ (See Lecture 2). The objective of this lecture is to prove a super-polynomial lower bound for the size of multilinear formulas computing Perm and Det, a result of Raz [11]. A formula is said to be *multilinear* if every node of the formula computes a multilinear polynomial of its inputs (e.g. $(x_1 \cdot x_2) + (x_1 \cdot x_3)$). This restriction was first investigated by Nisan and Wigderson [9] who obtained size-depth tradeoffs for multilinear circuits computing matrix product. The motivations for studying this restriction are numerous. First, most of the known formulas computing multilinear functions (such as Det, Perm and matrix product) are indeed multilinear (Berkowitz's construction being an exception). In fact, non-multilinear formulas are fairly counter-intuitive in this context as they imply cancellations of non-multilinear terms. Of course, multilinear polynomials and formulas are of interest on their own and multilinear formulas form a rather strong subclass of formulas. For instance, monotone formulas for a multilinear function are always multilinear since, by monotonicity, any higher power occurring in the computation cannot be cancelled at a later stage. Recently, Aaronson used the techniques outlined below to prove that certain quantum states used in quantum error correction cannot be expressed by a polynomial-size formula (using tensor and addition gates) [1].

18 Basic Properties of Multilinear Formulas

18.1 Semantic *vs* Syntactic Multilinearity

When talking about multilinear formulas, we need to distinguish two cases: we will say that formulas in which every gate computes a multilinear function of the inputs are *semantically* multilinear. If furthermore the formula is such that for any product gate each variable occurs in at most one of the two subtrees of this gate, we say that the formula is *syntactically* multilinear. For the purpose of proofs, it is easier to handle syntactically multilinear formulas and, conveniently, this restriction can be imposed at no cost.

Lemma 1. *Any semantically multilinear formula F has a syntactically multilinear equivalent of equal size.*

Proof. Suppose that $v = v_1 \cdot v_2$ is a product gate which is *not* syntactically multilinear and let x_i be a variable appearing in both subtrees of v. Since v computes a multilinear function, one of v_1, v_2 (say v_1) computes a function that does not depend on x_i. Accordingly, we can replace the subtree rooted at v_1 by the subtree obtained by setting x_i to the constant 0. □

It is not clear whether or not this lemma holds (perhaps in some weaker form) for circuits rather than formulas.

18.2 Partial Derivatives

As in lectures 2 (for Bauer-Strassen's Theorem) and 3 (for non-commutative circuits), partial derivatives will be at the heart of the arguments. Let f be a multilinear polynomial over the set of variables $Y \cup Z$, with $Y = \{Y_1, \ldots, Y_m\}$ and $Z = \{Z_1, \ldots, Z_m\}$. We will denote by P the set of multilinear monomials in the variables of Y and by Q the set of multilinear monomials over Z. The *partial derivatives matrix* $M = M_f$ is the $2^m \times 2^m$ matrix whose rows and columns are indexed by P and Q respectively and where the entry $M_f(p, q)$ is the coefficient of the monomial pq in f. In other words, the row corresponding to $p = Y_{i_1} \ldots Y_{i_t}$ gives the coefficients of the monomials of Q in the partial derivative $\frac{\delta f}{\delta Y_{i_1} \ldots \delta Y_{i_t}}$.

The cornerstone of Nisan's method for non-commutative formulas was the relationship between the rank of M_f and the formula-size s of the function f. (In fact we had $Rank(M_f) = poly(s_f)$ for non-commutative formulas.) Several other papers showed that an analogous theorem could be proved for other classes of formulas [7, 9, 13, 14] but is the same intuition valid for the multilinear case? Let us consider the simple example $f = \prod_{i=1}^{m}(Y_i + Z_i)$. Clearly, f has very small formula-size yet M_f is simply a permutation matrix and so $Rank(M_f) = 2^m$. At the very least, the naïve approach will not suffice.

We can prove a few simple facts about the rank of M_f.

1. If f depends on only k variables in Y, then $Rank(M_f) \leq 2^k$. This is clear since M_f contains at most 2^k non-zero rows.

2. If $f = g + h$ then
$$Rank(M_f) \leq Rank(M_g) + Rank(M_h).$$
This follows simply from the sub-additivity of the rank since $M_f = M_g + M_h$.

3. If $f = gh$ then
$$Rank(M_f) = Rank(M_g) \cdot Rank(M_h).$$
Here we have to use the fact that f, g, h are multilinear: since f is multilinear, g and h depend on disjoint sets of variables. So the rank of M_f is the rank of the tensor product of M_g and M_h.

19 Balanced and Unbalanced Multilinear Functions

We will denote as $Y_f \subseteq Y$ and $Z_f \subseteq Z$ the subsets of variables on which f depends. If $||Y_f| - |Z_f|| \geq k$ then we say that f is *k-unbalanced*. The crucial observation is that the rank of M_f cannot be too large if f is a product of multilinear function, one of which is k-unbalanced. More precisely

Lemma 2. *If $f = g \cdot h$ is computed by a product gate and one of g, h is k-unbalanced then $Rank(M_f) \leq 2^{m-k}$.*

Proof. It must be that either $|Y_g| + |Z_h| \leq m - k$ or $|Y_h| + |Z_g| \leq m - k$. Indeed, suppose w.l.o.g. that g is k-unbalanced and that $|Y_g| \geq |Z_g| + k$ but that $|Y_g| + |Y_h| \leq m$. Then $|Y_h| + |Z_g| + k \leq m$. Now if $|Y_g| + |Z_h| \leq m - k$ then by our previous remarks we have
$$Rank(M_f) = Rank(M_g) \cdot Rank(M_h) \leq 2^{|Y_g|} \cdot 2^{|Z_h|} \leq 2^{m-k}.$$
□

Corollary 1. *If $f = \sum_{i=1}^{s} g_i$ where each g_i is a product gate having a k-unbalanced son, then $Rank(M_f) \leq s \cdot 2^{m-k}$.*

Proof. Simply apply rank sub-additivity: by the previous lemma, $Rank(M_{g_i}) \leq 2^{m-k}$. □

Of course, these observations are useless if we fail to find unbalanced nodes in our multilinear formulas. Note however that we can pick Y and Z as we choose. We will do this in a random fashion as follows. We first randomly partition our set of inputs $X = \{X_1, \ldots, X_{2m}\}$ into Y and Z. If v depends on about m variables then with high probability v becomes m^ϵ unbalanced. We would hope that this suffices to simultaneously unbalance all of the top product gates (i.e. the first product gates on paths from the output node to the inputs). The caveat is that with probability about $m^{-1/2}$ the node v is completely balanced and if the formula contains more than $\sqrt{m}$ top product gates then some of them have balanced sons. Thus the probabilistic method is not quite good enough.

To work around this problem, we proceed inductively: if there are balanced top product gates, we can try to find unbalanced sons for each of these exceptions and the rank of the corresponding M_{g_i} will still be relatively small. So it is enough to prove that along every path from an input leave to the root there *exists* an unbalanced gate.

For a multilinear formula Ψ, we denote as $|\Psi|$ the size of the formula. A path from the leaf to the root is *central* if the total number of inputs feeding into successive gates along the path increases by a factor of at most 2 from node to node and we will say that Ψ is *k-weak* if every *central* path contains a k-unbalanced gate. Note that any Ψ contains at least one central path.

Lemma 3. *If Ψ is k-weak then*

$$Rank(M_\Psi) \le |\Psi| \cdot 2^{m-k/2}.$$

Proof. Say that a node v is k-weak if every central path to v contains a k-unbalanced node. We will show that if v is k-weak then $Rank(M_v) \le |\Psi_v| \cdot 2^{b(v)-k/2}$ where $b(v) = \frac{|Y_v|+|Z_v|}{2}$. We proceed by induction on the depth of the formula: If v is k-weak then it must be that $b(v) \ge k$ for otherwise, none of v or its children can be k-unbalanced.

We distinguish a number of cases:

- If v itself is unbalanced, then w.l.o.g. $|Y_v| \ge b(v) + k/2$ and $|Z_v| \le b(v) - k/2$. Thus, $Rank(M_v) \le 2^{b(v)-k/2}$ by our earlier observations.

- If v is a product $v = v_1 v_2$ but v itself is not k-unbalanced, we have $b(v) = b(v_1) + b(v_2)$ since we assume that the formula is syntactically multilinear. W.l.o.g. we assume that $b(v_1) \ge 1/2 \cdot b(v)$. Thus every central path to v_1 extends to a central path to v. So v_1 is also k-weak. By induction, we have $Rank(M_{v_1}) \le |\Psi_{v_1}| \cdot 2^{b(v_1)-k/2}$ and by our earlier remarks $Rank(M_{v_2}) \le 2^{b(v_2)}$. Thus

$$Rank(M_v) = Rank(M_{v_1}) Rank(M_{v_2}) \le |\Psi_{v_1}| \cdot 2^{b(v_1)-k/2} 2^{b(v_2)} \le |\Psi_v| \cdot 2^{b(v)-k/2}.$$

- Suppose $v = v_1 + v_2$. We distinguish two subcases. If $b(v_1)$ and $b(v_2)$ are at least $1/2 \cdot b(v)$ then, as in the previous case, both v_1 and v_2 are both k-weak since central paths to either node can be extended to central paths to v. By induction we have $Rank(M_{v_i}) \le |\Psi_{v_i}| \cdot 2^{b(v)-k/2}$. By rank sub-additivity we obtain

$$Rank(M_v) \le (|\Psi_{v_1}| + |\Psi_{v_2}|) \cdot 2^{b(v)-k/2} \le |\Psi_v| \cdot 2^{b(v)-k/2}.$$

  If on the other hand we have $b(v_1) \ge 1/2 \cdot b(v)$ but $b(v_2) < 1/2 \cdot b(v)$ then we must have $b(v_2) < b(v) - k$ since $b(v_2) < b(v)/2 < k/2$. Thus $Rank(M_{v_2}) \le 2^{b(v)-k/2}$ and by induction $Rank(M_{v_1}) \le |\Psi_{v_1}| \cdot 2^{b(v_1)-k/2}$. Hence,

$$Rank(M_v) \le (|\Psi_{v_1}| + 1) \cdot 2^{b(v)-k/2}.$$

□

For the purposes of the proof, we will later on choose $k \approx m^\epsilon$.

Lemma 4. *Let Ψ be a multilinear formula computing a degree m polynomial. If $|\Psi| \le m^{\log m/100}$, then a random partition makes Ψ k-weak with high probability for $k \approx m^\epsilon$.*

Proof sketch. Consider any given partition. The number of central paths is bounded above by the size $s \le m^{\log m/100}$ of the formula (since of course, there is at most one central path from any given node to the root). If we can show that for any central path, the probability that this path contains a k-unbalanced node is at least $1 - \epsilon/s$ then the union bound will allow us to conclude that all central paths are unbalanced w.p. at least ϵ.

Consider a central path γ: the length of the path is $\Omega(\log m)$ because the degree along this path increases by a factor of at most 2 at each step. Fix $l = \Omega(\log m)$ and consider the first node v_1 on the path where $b(v_1) \ge k \approx m^\epsilon$. We can find a sequence $v_1, \ldots, v_l$ of nodes on the path such that $b(v_{i+1}) \ge 100 b(v_i)$. The probability that γ is balanced is bounded above by the probability that each v_i is balanced and we can break this down using conditional probabilities:

$$\Pr[\gamma \text{ is balanced}] \le \prod_{i=1}^{l=\Omega(\log m)} \Pr[v_i \text{ is balanced} | v_j \text{ is balanced for each } j < i].$$

Initially, the probability that v_1 is balanced is at most $m^{-\delta}$ (for some properly chosen universal constant δ). If it is balanced, we consider v_2: since we have $b(v_2) \ge 100 \cdot b(v_1)$, the probability that v_2 is balanced is again smaller than $m^{-\delta}$, even if we condition on the event that v_1 is balanced. Similarly, the probability that v_i is balanced conditioned on all previous v_j's also being balanced is at most $m^{-\delta}$. So the probability that this central path is not unbalanced is at most

$$\prod_{i=1}^{l=\Omega(\log m)} \Pr[v_i \text{ is balanced} | v_j \text{ is balanced for each } j < i] \le m^{-\Omega(\log m)}.$$

□

Corollary 2. *If f is a multilinear function such that for every partition of the input variables we have $Rank(M_f) \approx 2^m$ then any multilinear formula Ψ for f has size $m^{\Omega(\log m)}$.*

20 Lower Bounds for Determinant and Permanent

Lower bounds for determinant and permanent would follow if these functions satisfied the conditions of Corollary 2. However, this is quite far from being the case: each of these functions have n^2 inputs but we have $Rank(M_f) \leq 2^n$ under any partition so we must further refine our techniques.

We will look at random restrictions of Det and Perm. Formally, we map the input variables $\{X_{ij}\} \to \{Y_1, \ldots, Y_m\} \cup \{Z_1, \ldots, Z_m\} \cup \{0, 1\}$ with $m = n^\epsilon$. The distribution is obtained as follows.

Step 1: Choose m submatrices of size 2×2, uniformly at random in the matrix.

Step 2: Map each submatrix i to one of $\begin{pmatrix} Y_i & Z_i \\ 1 & 1 \end{pmatrix}$ or $\begin{pmatrix} Y_i & 1 \\ Z_i & 1 \end{pmatrix}$.

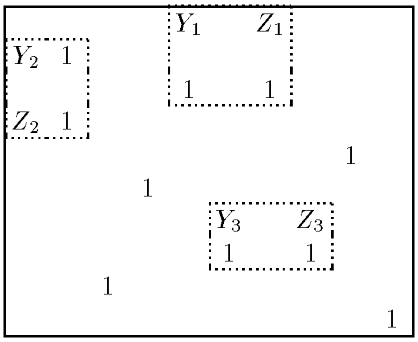
Step 3: Choose a perfect matching of the remaining rows and columns.

Step 4: Map the perfect matching to 1 and the remaining entries to 0.

For any such restriction r, let Ψ_r denote the formula from Ψ obtained by applying r.

Lemma 5. *After any such random restriction r, we have $Rank(M_{\Psi_r}) = 2^m$.*

Proof. After the restriction, Ψ is computing the permanent of a matrix of the form:



Thus, the function computed by Ψ after the restriction is simply $\prod_{i=1}^m (Y_i + Z_i)$. As we argued earlier, it is easy to see that $Rank(M_\Psi) = 2^m$. □

Note that we can argue similarly for the determinant, in which case we end up with the polynomial $\prod_{i=1}^m (Y_i - Z_i)$ which also yields full rank.

It is now possible to adapt Lemma 4 to show

Lemma 6. *If $|\Psi| \leq n^{\log n/100}$ and we map $\{X_{ij}\}$ to $\{Y_1, \ldots, Y_m\} \cup \{Z_1, \ldots, Z_m\} \cup \{0, 1\}$ as just described, then Ψ is k-weak for $k = n^\epsilon$ with high probability.*

Corollary 3. *If $|\Psi| \leq n^{\log n/100}$, then after applying the random restriction, $Rank(M_\Psi) < 2^m$ with high probability.*

This contradicts Lemma 5 so we can conclude:

Corollary 4. *Any multilinear formula for computing the Permanent or Determinant of an $n \times n$ matrix requires size at least $n^{\Omega(\log n)}$.*

21 Multilinear $\text{NC}_1 \neq$ Multilinear NC_2

In the first part of this lecture, we will construct an explicit multilinear polynomial $f(x_1, \ldots, x_n)$ with coefficients in $\{0, 1\}$ such that over any field:

- there exists a polynomial size, $O(\log^2 n)$ depth multilinear circuit computing f.
- any multilinear formula computing f has superpolynomial size.

This will directly imply that multilinear $\text{NC}_1 \neq$ multilinear NC_2 (see [12]). The idea in bounding the formula size of f will be to analyze the rank of the *partial derivative matrix* M_f of f for any partition of the input variables into two sets of equal size. For a given partition of the variables, the matrix M_f is defined as follows. For a multilinear polynomial f in variables $\{y_1, \ldots, y_m\}$ and $\{z_1, \ldots, z_m\}$, the entry $M_f(p, q)$ corresponding to monomials p over $\{y_1, \ldots, y_m\}$ and q over $\{z_1, \ldots, z_m\}$ is the coefficient of the monomial pq in f. The following statement can be proved using results of the previous lecture.

Proposition 3. *A random partition of the variables of any polynomial size multilinear formula results in a low rank partial derivative matrix with high probability*[2].

The function that we construct will have the property that for *every* partition of its input variables into two equal size sets, M_f will have full rank (i.e. 2^m assuming we have $2m$ input variables). Note that this is a much stronger property than having full rank with just some probability over random partitions. Thus, one can conclude that f requires superpolynomial size multilinear formulas.

Let F be an arbitrary field. Let f be a multilinear polynomial over F and over a set of variables $X = \{x_1, \ldots, x_n\}$ and a set of auxiliary variables $X' = \{x'_1, \ldots, x'_r\}$. Let $A' : X' \to F$ be an assignment to the variables in X'. Denote by $f_{A'}$ the polynomial obtained from f using the assignment A'. Note that $f_{A'}$ is a multilinear polynomial over F in variables from X.

We will use the following two simple corollaries of the above proposition:

Proposition 4. *Let f be a multilinear polynomial over a field F and over the set of variables $X = \{x_1, \ldots, x_n\}$ and $X' = \{x'_1, \ldots, x'_r\}$, with r being polynomially bounded in n. If there exists some assignment $A' : X' \to F$ such that $M_{f_{A'}}$ has full rank over F for every partition of X into sets of size m (assuming $n = 2m$), then for any multilinear formula Φ computing f, we have $|\Phi| \geq n^{\Omega(log\ n)}$.*

Proof. Consider $\Phi_{A'}$, the formula obtained from Φ after fixing the variable in X' according to the assignment A'. Then clearly $\Phi_{A'}$ is a formula for $f_{A'}$ and $|\Phi_A| = |\Phi|$. The statement follows by the previous proposition. □

To obtain assignments that give full rank for every partition, we sometimes will need to consider extensions of the field F.

Proposition 5. *Let f be a multilinear polynomial over a field F just as in the previous proposition. Consider a field extension G of F, and now treat f as a polynomial over G. If there exists an assignment $A' : X' \to G$, such that $M_{f_{A'}}$ has full rank over G for every partition of X into two equal size sets, then any multilinear formula representing f over F must have size at least $n^{\Omega(log\ n)}$.*

The proof follows by observing that any multilinear formula for f over F is also a multilinear formula for f over the extended field G.

[2] this is shown in two steps. In the first step we show that the formula with high probability becomes k-weak for $k = O(n^{1/8})$. In the last step we show that the partial derivative matrix of every polynomial size k-weak multilinear formula has low rank

21.1 Construction of the function

Before we can give the construction of the multilinear polynomial f, we need to introduce some more notation. Let $[n] = \{1, \ldots, n\}$. For each $i, j \in [n]$, denote by $[i, j]$ the interval of $[n]$ starting at i and ending at j. Let S denote the set of all intervals in $[n]$ (including the empty interval denoted by $\emptyset$) and Γ be the set of all pairs of disjoint intervals in S. (Two intervals are called disjoint if the two corresponding sets are disjoint.) For any $\tau \in \Gamma$, the size of τ (denoted by $|\tau|$), is the sum of the cardinalities of the two intervals in τ. We shall call a pair of intervals τ to be even if it has an even size.

For every $\tau_1, \tau_2, \tau \in \Gamma$, where $\tau_1 = (s_{1,1}, s_{1,2})$, $\tau_2 = (s_{2,1}, s_{2,2})$ and $\tau = (s_1, s_2)$, we say that $\{\tau_1, \tau_2\}$ is a *partition* of τ iff $s_1 \cup s_2 = s_{1,1} \cup s_{1,2} \cup s_{2,1} \cup s_{2,2}$, and $s_{1,1} \cup s_{1,2}$ is disjoint from $s_{2,1} \cup s_{2,2}$. For every even τ, let $\mathcal{P}(\tau) = \{(\tau_1, \tau_2) : \{\tau_1, \tau_2\}$ is a partition of τ and both τ_1 and τ_2 are even$\}$.

We will define recursively for any even $\tau = (s_1, s_2) \in \Gamma$, the function f_τ in the following way: $f_{\emptyset,\emptyset} = 1$, and if $|\tau| = 2$, then $f_\tau = x_i x_j + 1$ where $i, j \in [n]$ are the two indices appearing in τ. Otherwise,

$$f_\tau = \sum_{\{\tau_1,\tau_2\} \in \mathcal{P}(\tau)} X'_{\tau,\tau_1,\tau_2} \cdot f_{\tau_1} \cdot f_{\tau_2}$$

The polynomial f that we construct is simply $f_{[n],\emptyset}$ for even n. Each X'_{τ,τ_1,τ_2} is an auxiliary variable of f. Note that since the polynomials f_{τ_1} and f_{τ_2} depend on disjoint sets of variables, the resulting polynomial is multilinear.

Clearly, f can be computed by a polynomial size multilinear circuit. This simply follows from the recursive definition of f. Using the result of Valiant, Skyum, Berkowitz and Rackoff [19], this implies that f is in multilinear NC$_2$. Note that this holds over any field F. (Since f has only coefficients from $\{0, 1\}$ it can be interpreted over any field F.)

Now we want to show that over any field F, no polynomial size multilinear formula can compute f. To show this we will do the following: extend the field F to a field G such that the transcendental dimension of G over F is infinite, i.e. there are infinitely many elements in G that are algebraically independent over F. Consider f as a polynomial over the extended field G. Fix the auxiliary variables of f using an assignment $A' : X' \to G$, such that the values substituted are algebraically independent over F. Denote the value of the auxiliary variable X'_{τ,τ_1,τ_2} by C_{τ,τ_1,τ_2} after the substitution. We will show that the function $f_{A'}$ over G cannot have a polynomial size multilinear formula, by showing that its partial derivative matrix $M_{f_{A'}}$ has full rank over G for every partition. Then, proposition 5 will imply that f over the field F cannot be computed by a polynomial size multilinear formula.

We need some more notation. Let $X = \{x_1, \ldots, x_n\}$, $Y = \{y_1, \ldots, y_m\}$ and $Z = \{z_1, \ldots, z_m\}$ with $n = 2m$. Given any partition $P : X \to Y \cup Z$, we say that a pair of intervals $\tau \in \Gamma$ is balanced on P if the number of indices in τ that get mapped to an index in Y is equal to the number of indices in τ that get mapped to an index in Z. Given a partition P, we consider the partial derivative matrix with respect to that partition. (This will not be indicated in the notation, but should be clear from the context.) We shall prove the following lemma inductively:

Lemma 7. *For every partition P and every even $\tau \in \Gamma$ that is balanced on the partition P, Rank$(M_{f_\tau}) = 2^{|\tau|/2}$.*

Clearly, when the lemma above is applied to $f_{[n],\emptyset}$ for even n, in combination with Proposition 5 it gives us the result that polynomial size multilinear formulas computing f cannot exist. Thus, separating multilinear NC$_1$ from multilinear NC$_2$ reduces to proving lemma 7.

For a partition $P : X \to Y \cup Z$ define $B : [n] \to \{-1, 1\}$, where for any $i \in [n]$, $B(i) = -1$ iff there exists a $j \in [m]$ such that $P(x_i) = y_j$. For any interval $s \in S$ (or pair $\tau \in \Gamma$), denote by $B(s)$ (or $B(\tau)$) the sum of the values of B over the elements of s (or τ). Clearly, τ is balanced on partition P iff $B(\tau) = 0$. We shall prove the following easy claim:

Lemma 8. *Consider an arbitrary partition $P : X \to Y \cup Z$. For each $\tau \in \Gamma$ that is balanced on P with $|\tau| \geq 4$, there exists $\{\tau_1, \tau_2\} \in \mathcal{P}(\tau)$ such that both τ_1 and τ_2 are balanced on P.*

Proof. Let $\tau = \{s_1, s_2\}$, where s_1 and s_2 are disjoint intervals. If $B(s_1) = B(s_2) = 0$, then $\tau_1 = \{s_1, \emptyset\}$ and $\tau_2 = \{\emptyset, s_2\}$ gives us the claim. Otherwise, w.l.o.g. assume $B(s_1)$ is negative and $B(s_2)$ is positive.

Further, let $s_1 = [i, j]$ and $s_2 = [k, \ell]$. If $B(i) \neq B(k)$, then $\tau_1 = ([i, i], [k, k])$ and $\tau_2 = ([i + 1, j], [k + 1, \ell]$ does the job. Thus w.l.o.g, assume $B(i) = B(k) = 1$. In that case, as $B(s_1)$ is negative, there exists an $i' \in s_1$ such that $B([i, i']) = 0$. Then $\tau_1 = ([i, i'], \emptyset)$ and $\tau_2 = ([i' + 1, j], s_2)$ yields the claim. □

Now we are ready to prove lemma 7.

Proof of lemma 7. We shall prove this by induction on the size of τ. For $|\tau| = 2$, f_τ was defined as $x_i x_j + 1$. So if τ is balanced on a partition P, M_{f_t} is the 2×2 identity matrix and so has rank 2. This gives us the base case of the induction.

For $|\tau| > 2$, the definition of f_τ as given before implies

$$M_{f_\tau} = \sum_{\{\tau_1, \tau_2\} \in \mathcal{P}(\tau)} C_{\tau, \tau_1, \tau_2} \cdot M_{f_{\tau_1} \cdot f_{\tau_2}}$$

Since the coefficients C_{τ, τ_1, τ_2} and all other coefficients that appear inside the matrices used in the recursive definition are algebraically independent over F, we get

$$Rank(M_{f_\tau}) \geq max_{\{\tau_1, \tau_2\} \in \mathcal{P}(\tau)} Rank(M_{f_{\tau_1} \cdot f_{\tau_2}})$$

Using results from the previous lecture,

$$Rank(M_{f_{\tau_1} \cdot f_{\tau_2}}) = Rank(M_{f_{\tau_1}}) \cdot Rank(M_{f_{\tau_2}})$$

Lemma 8 says that there exists a pair $\{\tau_1', \tau_2'\} \in \mathcal{P}(\tau)$ such that both τ_1' and τ_2' are balanced on P. Since $|\tau_1'|, |\tau_2'| < |\tau|$, we can use our inductive hypothesis on them to conclude that

$$Rank(M_{f_{\tau_1}}) \geq Rank(M_{f_{\tau_1'}}) \cdot Rank(M_{f_{\tau_2'}}) \geq 2^{|\tau_1'|/2} \cdot 2^{|\tau_2'|/2} = 2^{|\tau|/2}$$

□

22 Valiant's Lemma

In the second part of this lecture, we prove a lemma by Valiant that we hinted at in Lecture 2, when presenting Valiant's approach based on matrix rigidity to prove superlinear size lower bounds for circuits with logarithmic depth [17].

We start with the statement of the main lemma. Note that this lemma only concerns the underlying graph of the circuit and completely ignores the type of gates used. Thus it can be applied to any kind of circuits. We will consider only fan-in 2 gates, but the statement can be easily modified to unbounded fan-in gates if we count the number of edges for size instead of the number of gates.

Lemma 9. *Let C be a fan-in 2 circuit over n input variables with size $s = O(n)$ and depth $d = O(\log n)$. Assume $d = 2^k$. Then we can remove $O(s/k) = O(n/\log\log n)$ edges, such that the depth of the remaining graph is at most $\epsilon \log n$, that is there is no path of length $> \epsilon \log n$ left in the graph.*

Proof. It will be enough to prove that by removing s/k edges, the depth can be reduced from d to $d/2$. Then, iterating this r times, we can reduce the depth to $d/2^r$. Choosing r to be a suitable constant, we get the statement of the Lemma.

First we group the vertices of the graph into *levels*, defined as follows. The level 0 consists of all the output gates. For every vertex, its level number will be the length of the *longest* path from it to an output. Thus, the largest level number used will be $d - 1 = 2^k - 1$. We think of the graph of the circuit as a directed acyclic graph, such that the edges are directed from the inputs towards the outputs. Thus, every edge from a given vertex is directed towards a vertex with a smaller level number. To simplify the presentation, we will refer to the level number of a vertex as its *label*. Since the largest label we use is $2^k - 1$, all labels can be represented by k-bit binary strings $i = i_1 \ldots i_k$. We denote by $EDGES_r$ the set of edges such that the first difference in the binary representation of the labels of its endpoints occurs at the r-th coordinate. That is, an edge with endpoints at level i and j belongs to $EDGES_r$ if $i_t = j_t$ for $t < r$ and $i_r \neq j_r$. Note that the sets $EDGES_r$ for $r = 1, \ldots, k$ form a partition of the set of all edges of the graph. This implies that for some r the set $EDGES_r$ contains at most s/k edges. We will remove all edges of this set (at most s/k edges) from the graph. It remains to prove, that after this there is no path of length $> d/2$ left in the graph.

Consider the longest path P in the remaining graph, and suppose it has length $\ell > d/2 = 2^{k-1}$. Note that the endpoints of any edge must have different labels, thus there must be ℓ different labels that appear along this path. Group the vertices of the path P according to the prefixes of their labels in the first $r - 1$ coordinates. This gives at most 2^{r-1} groups, since there are 2^{r-1} different possible prefixes. Note also that the labels along any path must appear in decreasing order, does we get a partition of the vertices of the path P into parts within each group. Since we removed all edges from $EDGES_r$, each vertex of P that belongs to the same group must have the same value in the r-th coordinate of its label, that is, either the labels of all vertices of a given group have their r-th coordinate equal 0 or all have their r-th coordinate equal 1. Otherwise, we would have an edge between two vertices within a group whose labels differ in their r-th coordinate, and this would be the first coordinate where they differ, since the labels of all vetices within a group have the same prefix in their first $r - 1$ coordinates. But this means that we can have at most 2^{k-r} different vertices in each group. On the other hand, if the path P consists of $> d/2 = 2^{k-1}$ vertices, then some group must contain $> 2^{k-1}/2^{r-1} = 2^{k-r}$ vertices and we get a contradiction. □

22.1 Valiant's approach based on rigidity

Let $f_1, \ldots, f_n$ be linear functions of the variables $x_1, \ldots, x_n$. Let A be the $n \times n$ matrix of the corresponding linear transformation, that is let the i-th row of the matrix consist of the coefficients of the function f_i.

We say that the matrix A is *rigid* if it is not possible to obtain from it a matrix of rank at most $m = O(n/\log\log n)$ by changing at most n^ϵ entries in each row.

Theorem 5. *If A is a rigid matrix, then the linear transformation Ax cannot be computed by a circuit of $+$ gates of size $O(n)$ and depth $O(\log n)$.*

Proof. Assume that there is a circuit of size $O(n)$ and depth $O(\log n)$ computing $f_1, \ldots, f_n$. Then by the Lemma, we can remove $m = O(n/\log\log n)$ edges from the circuit, such that the depth of the remaining circuit is at most $\epsilon \log n$. This means, that each output gate of the circuit depends on at most n^ϵ input variables in addition to at most m gates of the circuit that are adjacent to the m edges removed. Let $h_1, \ldots, h_m$ denote the linear functions computed by the subcircuits at those m gates. Then for each $i = 1, \ldots, n$, we have

$$f_i = \sum_{j=1}^{m} c_{ij} h_j + \sum_{\ell=1}^{n} e_\ell x_\ell$$

where at most n^ϵ of the e_ℓ-s are nonzero. Notice that we can change the entries of A corresponding to the nonzero e_{ell}-s in each row, such that the linear transformation A' after the changes computes the functions

$$f_i = \sum_{j=1}^{m} c_{ij} h_j$$

and notice that the rank of the new matrix A' is at most m. However, if A is rigid, this is not possible, and we get a contradiction. □

23 Introduction

Today we consider the bounded coefficient algebraic circuit model. This is defined over fields of characteristic 0, say $\mathbb{R}$ or $\mathbb{C}$. As before, we have inputs $x_1, \ldots, x_n$, $\times$ and $+$ gates, only now instead of all constants over the field, we restrict our operations to multiplying only by small constants c where $|c| \leq 1$. For example, to derive $3x$ we cannot multiply x by 3; instead we must add $x + x + x$.

24 Linear functions

We first consider the setting where we have n inputs $x_1, \ldots, x_n$ and we want to compute a family of linear functions on these inputs. We can view this as multiplying the input by H, the Hadamard (or Fourier transform) matrix. This is a linear computation, so it can be performed by a linear circuit. We do not need to use multiplication except to multiply by small constants in the field, so the computation only requires + and - gates.

Morgenstern [6] analyzed linear functions in this model. In particular, he asked how many gates are required in the bounded coefficient model to compute linear functions $L_1, \ldots L_k$ over $x_1, \ldots, x_n$. We will se that when $k = n$ and the linear functions are described by the Hadamard matrix H, then we require $\Omega(n \log n)$ operations. There is also an upper bound of $O(n \log n)$ based on a standard recursive construction, so these bounds are tight.

Note that since we just argued that linear circuits are sufficient, a lower bound for bounded coefficient linear circuits implies a lower bound on bounded coefficient general circuits. This is based on the homogenization trick to make the circuit linear.

Morgenstern's result is formalized in the following main lemma:

Lemma 10. *There are $n \times n$ matrices H, such that any circuit C with bounded coefficients that computes $H \cdot \vec{x}$ must have size*

$$size(C) \geq \log_2(\mathit{Det}(H)).$$

Proof. We order the gates in any way that respects the order gates are encountered in the circuit. Consider the set of s linear functions f_i that are computed at the various gates throughout the computation, where s is the size of C. For $1 \leq i \leq n$, let $f_i = x_i$. For $i > n$, let f_i be the linear function of $x_1, \ldots, x_n$ computed at the ith gate, where $f_{s-n+1}, \ldots f_s$ are the rows of the matrix H, and thus the output gates.

For $i \geq n$, define functions D_i as follows:

$$D_i = \max_{\substack{f_{i_1}, \ldots, f_{i_n} \\ i_1, \ldots, i_n \leq i}} \text{Det}[f_{i_1} \cdots f_{i_n}],$$

where each of the f_{i_j} are linear functions.

We have $D_n = 1$, since the first n functions are the inputs. Also, we shall see that for all $i > n$, $D_{i+1} \leq 2D_i$. There are two cases. If f_{i+1} does not appear in the set $f_{(i+1)_1}, \ldots, f_{(i+1)_n}$, then $\max \text{Det}[f_{i+1_1}, \ldots f_{i+1_n}]$ is the same, so $D_{i+1} = D_i$.

If f_{i+1} does appear, then it is the sum of two previous functions $f_j + f_k$ or the product of one previous function with a small constant. In the former case, we assume wlog that $f_{(i+1)_1} = f_{i+1}$ in the largest determinant, and note that

$$\text{Det}[f_{i+1}, f_{(i+1)_2}, \cdots, f_{(i+1)_n}] = \text{Det}[f_j, f_{(i+1)_2}, \cdots, f_{(i+1)_n}] + \text{Det}[f_k, f_{(i+1)_2}, \cdots, f_{(i+1)_n}],$$

by linearity of the determinant function. Each determinant in this sum is upper bounded by D_i, so $D_{i+1} \leq 2D_i$. In the latter case where f_{i+1} is computed by multiplying by a constant bounded by 1, the claim also follows.

Hence, we have inductively argued that $D_s \leq 2^{s-n+1}$. Moreover, since the last n functions f_i compute H, we have $\text{Det}(H) \leq D_s$, so $s \geq \log_2 \text{Det}(H)$. $\square$

Corollary 5. *If C is a bounded coefficient algebraic circuit for computing the Hadamard matrix, then*

$$size(C) = \Omega(n \log_2 n).$$

Proof. The determinant of the Hadamard matrix is n^n, so the size of the circuit $s \geq \log_2(n^n) = n \log_2 n$. $\square$

25 Quadratic Functions

What about lower bounds for computing quadratic functions in this model? Unfortunately, we cannot apply the Baur-Strassen method because we can no longer make one quadratic function out of n linear ones in the bounded coefficient model. We would like to show that matrix multiplication requires at least $\Omega(n^2 \log n)$ operations.

The method we shall see for proving this has been generalized by Bürgisser and Lotz [4] to show that this is a tight lower bound for the product of two polynomials. This is particularly nice since all known methods for this problem do not use large constants.

This result will be in the spirit of Morgenstern's, but will strengthen this method so that it will give meaningful lower bounds, even if two rows of H are the same and hence $\text{Det}(H) = 0$.

Let $L_1, \ldots, L_k$ be linear functions, where we now allow $k \neq n$. Let

$$\text{Vol}_r(L_1, \ldots, L_k) = \max\{\text{vol spanned by } L_{i_1}, \ldots, L_{i_r} \text{ for } \{L_{i_1}, \ldots, L_{i_r}\} \subseteq \{L_1, \ldots, L_k\}\}.$$

Note that an equivalent formulation is to maximize $(\text{Det}[AA^T])^{1/2}$, where A is an $n \times r$ minor of the $n \times k$ matrix described by $L_1, \ldots, L_k$.

The following lemma is a generalization of Morgenstern's.

Lemma 11. *For all linear functions $L_1, \ldots, L_k$, and for all $r \leq k$, and any circuit C for computing these functions must have*

$$size(C) \geq \log_2 \mathit{Vol}_r(L_1, \ldots, L_k).$$

The proof is almost identical to the proof of Lemma 10, and we omit it here.

It now will be useful to define the *rigidity* of our input $L_1, \ldots, L_k$. This will measure how close our linear functions are to lying in a subspace of small dimension. It is parameterized by r as follows:

$$\text{Rig}_r(L_1, \ldots, L_k) = \min_V \max_{1 \leq i \leq k} \text{Dist}(L_i, V),$$

where the minimum is taken over subspaces V of dimension r. See figure 4

Lemma 12. *For any linear functions $L_1, \ldots L_k$,*

$$\mathit{Vol}_r \geq (\mathit{Rig}_r)^r.$$

Proof. Order the L_i as follows. Let L_1 be the longest vector. T hen let L_2 be the vector that maximizes the volume spanned by L_1 and L_2. In general, let L_i be the vector that maximizes the volume of $L_1, \ldots, L_i$. Then, since Vol_r maximizes the volume over all choices of L_i, we have

$$\begin{aligned} \text{Vol}_r(L_1, \ldots, L_k) &\geq \text{Vol}_r(L_1, \ldots, L_r) \\ &= \text{Vol}_1(L_1)\frac{\text{Vol}_2(L_1, L_2)}{\text{Vol}_2(L_1)} \cdots \frac{\text{Vol}_r(L_1, \ldots, L_r)}{\text{Vol}_{r-1}(L_1, \ldots, L_{r-1}}. \end{aligned}$$

Now, observe that

$$\text{Vol}_1(L_1) \geq \frac{\text{Vol}_2(L_1, L_2)}{\text{Vol}_1(L_1)} \geq \cdots \geq \frac{\text{Vol}_r(L_1, \ldots, L_r)}{\text{Vol}_{r-1}(L_1, \ldots, L_{r-1})}$$

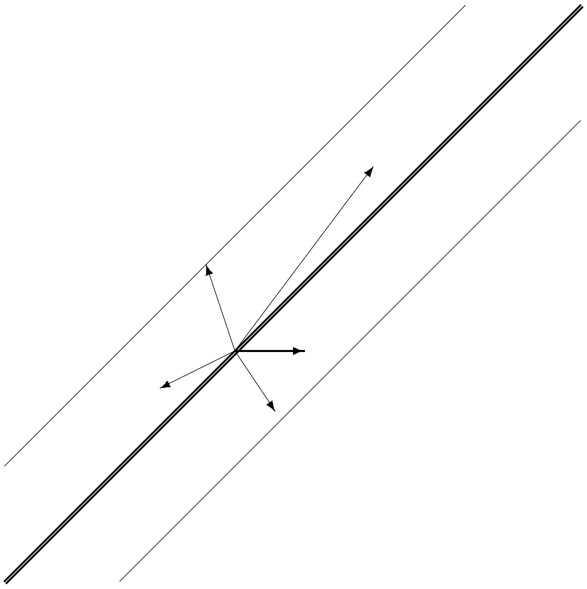


Figure 4: Example of rigidity $\mathrm{Rig}_r(L_1, \ldots, L_k)$.

from the way that we ordered the L_i. Putting these together, we find

$$\begin{aligned} \mathrm{Vol}_r(L_1, \ldots, L_k) &\geq \left(\frac{\mathrm{Vol}_{r+1}(L_1, \ldots, L_{r+1})}{\mathrm{Vol}_r(L_1, \ldots, L_r)} \right)^r \\ &\geq (\mathrm{Rig}_r)^r. \end{aligned}$$

The last inequality comes from the definition of Rig_r. The fraction is measuring the distance of L_{r+1} from the vector space defined by $(L_1, \ldots, L_r)$, since the ordering of the L_i says that L_{r+1} will maximize this distance, and Rig_r maximizes this order over all choices of r-dimensional vector spaces and hence can only be smaller. □

This gives the following corollary.

Corollary 6. *For any circuit C,*

$$size(C) \geq \log_2 \mathit{Vol}_r \geq r \log_2 \mathit{Rig}_r.$$

We return now to the problem of matrix multiplication. Given an $n \times n$ matrix H, let I be the $n \times n$ identity matrix and consider the tensor product $I \otimes H$ defined as

$$\begin{pmatrix} \boxed{H} & & & & \\ & \boxed{H} & & 0 & \\ & & \boxed{H} & & \\ & 0 & & \boxed{H} & \\ & & & & \boxed{H} \end{pmatrix}$$

This tensor product is potentially useful because, in order to compute the matrix $H \cdot A$, we can think of fixing H and viewing A as a column vector $\vec{x} = (x_{1,1}, \cdots, x_{1,n}, x_{2,1}, \cdots, x_{2,n}, \cdots, x_{n,n})$. Thus, $H \cdot A = (I \otimes H) \cdot \vec{x}$. This serves the function of treating matrix multiplication as n copies of H, each working on a different set of variables, like the direct sum problem.

Recall that

$$size(C) \geq \log_2 \mathrm{Vol}_r \geq r \log_2 \mathrm{Rig}_r$$

which implies
$$\log_2 \text{Vol}_{rm}(I \otimes H) \geq m \log_2 \text{Vol}_r(H),$$
or equivalently,
$$\text{Vol}_{rm}(I \otimes H) \leq [\text{Vol}_r(H)]^m.$$
This is from the strong direct sum theorem.

Take $r = m/10$. We can think of a bilinear circuit for matrix multiplication as taking inputs $\{X_i\}$ and $\{Y_i\}$ where we first computed products $L_1(Y) \times R_1(X), \ldots, L_s(Y) \times R_s(X)$. Afterwards we computed linear combinations of the outputs from these product gates.

Now, here's the main idea. We want to fix Y to be some fixed matrix H that is difficult to compute in the bounded coefficient model. Recall that when the coefficients are small, we can replace $\times$ by $+$. So we want to find H so that $L_1(Y), \ldots, L_s(Y)$ are all small, but they maintain high rigidity. Since the L_i are linear functions of the Y_i that are computed by a bounded coefficient circuit, these have size at least $\epsilon m^2 \log m$ altogether.

First, we observe the following.

Lemma 13. *The rigidity satisfies*
$$Rig_{rm}[L_1, \ldots, L_s] \leq m^{10\epsilon}.$$

Proof. Recall that the rigidity satisfies $size(C) \geq r \log_2 \text{Rig}_r$, which implies $size(C) \geq rm \log_2 \text{Rig}_{rm}$. Therefore
$$2^{\epsilon m^2 \log_2 m} \geq (\text{Rig}_{rm})^{rm}$$
which implies
$$2^{10\epsilon \log_2 m} \geq (\text{Rig}_{rm}),$$
as required. □

We can now state the main lemma.

Lemma 14. *There exists a matrix H such that*

1. *For all i, $L_i(H) \leq O(m^{10\epsilon} \log m)$*
2. *$Rig_r(H) \geq \Omega(m)^{1/2}$*

This lemma contains the key idea used to prove the lower bound. We have from Lemma 13 that $\text{Rig}_{rm}(L_1, \ldots, L_s) \leq m^{10\epsilon}$. This implies that there exists a vector space V of relatively small dimension rm such that all the L_i are close to V. We also know that any circuit must be large since $\text{Rig}_r(H) \geq \Omega(m^{1/2})$. These two facts together almost imply the lower bound. However, we need to convert this circuit to one with bounded coefficients.

The problem is that the field elements $L_i(Y)$ are of size $m^{10\epsilon}$ and are not bounded by 1. We can replace these products by $\log(|L_i(Y)|)$ additions, although there are k of them, where k can be almost as large as the circuit. Therefore this approach might increase the size of the circuit by more than a constant factor.

To avoid this problem, we first multiply all the $L_i(Y)$ by $m^{-20\epsilon}$ throughout, so these now have size at most 1. These are the inputs to the $\times$ gates. At the end when we compute the linear combinations of these products, we multiply the outputs by $m^{20\epsilon}$. Notice that each of these multiplications can be achieved with at most $\log(m^{20\epsilon})$ additions. The number of these outputs is only m^2, so this increases the size of the circuit by at most $20\epsilon m^2 \log m$. This will give us the $\Omega(m^2 \log m)$ lower bound, as desired. Moreover, since the circuit is bilinear, these modifications will not affect the overall output of the new circuit.

The upshot is that with Lemma 14 we can derive the desired lower bound.

Theorem 6. *If C is a bounded coefficient arithmetic circuit for computing the product of two $m \times m$ matrices, then*
$$size(C) = \Omega(m^2 \log m).$$

All that remains is proving Lemma 14. We sketch the main ideas below.

Proof of Lemma 14. We know $\mathrm{Rig}_{rm}(L_1, \ldots, L_s) \leq m^{10\epsilon}$. Therefore, there exists a vector space V of dimension rm that is close to all the L_i. We have

$$V \subseteq \mathbb{R}^{m^2},$$

since $L_1, \ldots, L_s$ is a vector space in m^2 variables.

We construct H in two stages.

1. Define W to be a random $m \times m$ matrix. For example, let

$$W_{ij} = N(0, 1),$$

   where $N(0, 1)$ is the normal distribution with expectation 0 and variance 1.

2. Now look at $V^\perp$ and let H be the projection of W onto $V^\perp$. H has a multinormal distribution which will be preserved under this projection.

We can now make some observations. First, for all L_i, we know L_i is close to V, and H is a projection onto $V^\perp$, so with high probability $L_i(H)$ is very small. This will give us condition (1).

Second, we can argue that $\mathrm{Rig}_r(H)$ is large. The projection of W onto V is about $\frac{m^2}{10}$, so the dimension of $V^\perp$ is about $\frac{9m^2}{10}$. The Hoffman-Weilandt Theorem (see e.g. [8]) states that singular values of H are close to the singular values of W. This allows us to derive an appropriate bound on $\mathrm{Rig}_r(H)$, ultimately giving us condition (2). $\square$

References

[1] S. Aaronson. Multilinear formulas and skepticism of quantum computing. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 118–127, 2004.

[2] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[3] S. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.

[4] P. Bürgisser and M. Lotz. Lower bounds on the bounded coefficient complexity of bilinear maps. *J. ACM*, 51(3):464–482, 2004.

[5] J. Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[6] J. Morgenstern. Note on a lower bound of the linear complexity of the fast Fourier transform. *Journal of the ACM*, 20(2):305–306, 1973.

[7] N. Nisan. Lower bounds for non-commutative computation. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 410–418, 1991.

[8] N. Nisan and A. Wigderson. On the complexity of bilinear forms. In *STOC*, pages 723–732, 1995.

[9] N. Nisan and A. Wigderson. Lower bounds on arithmetic circuits via partial derivatives. *Computational Complexity*, 6(3):217–234, 1997.

[10] P. Pudlák. Communication in bounded depth circuits. *Combinatorica*, 14(2):203–216, 1994.

[11] R. Raz. Multi-linear formulas for permanent and determinant are of super-polynomial size. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC'04)*, pages 633–641, 2004.

[12] R. Raz. Multilinear $\mathrm{NC}_1 \neq$ Multilinear NC_2. In *FOCS*, pages 344–351, 2004.

[13] R. Raz and A. Shpilka. Deterministic polynomial identity testing in non-commutative models. In *IEEE Conference on Computational Complexity*, pages 215–222, 2004.

[14] A. Shpilka and A. Wigderson. Depth-3 arithmetic circuits over fields of characteristic zero. *Computational Complexity*, 10(1):1–27, 2001.

[15] V. Strassen. Die Berechnungskomplexität von elementarsymmetrischen Funktionen und von Interpolationskoeffizienten. *Numerische Mathematik*, 20:238–251, 1973.

[16] V. Strassen. Vermeidung von divisionen. *J. Reine Angew. Math.*, 264:182–202, 1973.

[17] L. G. Valiant. Graph-theoretic properties in computational complexity. In *Proceedings of the Conf. on Mathematical Foundations of Computer Science (MFCS'77)*, volume 53 of *Lecture Notes in Computer Science*, pages 162–176. Springer-Verlag, 1977.

[18] L. G. Valiant. Completeness classes in algebra. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing (STOC'79)*, pages 249–261, 1979.

[19] L. G. Valiant, S. Skyum, S. Berkowitz, and C. Rackoff. Fast parallel computation of polynomials using few processors. *SIAM Journal on Computing*, 12(4):1983, 641–644.