COMS 6998: Unconditional Lower Bounds and Derandomisation **Spring 2024**

Lecture 3: January 30, 2024

Lecturer: Rocco Servedio *Scribe: Ekene Ezeunala*

Last time, we:

- Finished the last bit of overview; showed that if we have a PRG for a class $\mathcal{C}$, we not only get worst-case lower bounds, but also average-case lower bounds. (And so if we have a class of functions in a restricted computational model, we should try to first prove worst-case lower bounds; if we succeed, we should then try to prove average-case lower bounds, and if we succeed, we can try to give PRGs for that class.)
- Mentioned deterministic approximate counting—which gets us to the same endpoint of figuring out how many satisfying assignments there are for these different types of functions, but via an algorithmic route.
- Gave various worst-case lower bounds for Boolean formulas:
  - Shannon: non-explicit $\Omega(2^n/\log n)$ lower bound;
  - Subbotovskaya: lower bound of $\Omega(n^{1.5})$ for the parity function PAR;
  - Andre'ev: lower bound of $\Omega(n^{2.5})$ for $A(x,y) = f_y(x)$, where f_y is the function
    $$f_y(x_1,\ldots,x_m) = y\left(\bigoplus_{i=1}^{m} x_i, \bigoplus_{i=m+1}^{2m} x_i, \ldots, \bigoplus_{i=n-m+1}^{n} x_i\right),$$
    where $m = n/\log n$.

Today, we will:

- See a little more on formulas: the KRW conjecture (briefly!), size-depth tradeoffs, $\Theta(\log \text{size}(f))$ lower bound for the depth of full-basis Boolean formulas F.
- Start the unit on constant-depth circuits. (A $2^{\Omega\left(n^{1/(d-1)}\right)}$ size lower bound for depth-d circuits.)

1 Introduction, KRW Conjecture

The readings for today are Sections 6.1, 12.1, and 12.2 of [J+12] and Sections 3.3 and 3.4 of [BS90].

Last time we proved a lower bound of $\tilde{\Omega}(n^{2.5})$ for the Andre'ev function $A(x, y)$ formula size, if we believe that the shrinkage exponent $\Gamma \geq 3/2$ (which we proved, so we should believe this!). Furthermore, we showed a $\tilde{\Omega}(n^{\Gamma+1})$ lower bound in general for $A(x, y)$. The following question is natural:

Question: Can we do better than this, perhaps by recursively applying the reasoning we used in the construction of a hard function for an augumented version of the Andre'ev function?
▷ Suppose we take an input of size $2n \log n + n$ which we view as $f(x_1, \ldots, x_{\log n}, \psi)$, where all the x_i have length $2n$ and we interpret the last n bits as encoding a function on $\log n$ variables. However, the proof doesn't work via Andre'ev's methods.

The KRW conjecture is a famous conjecture related to this question of whether the formula size behaves as we expect it to for these simple combinations.

Conjecture 1 (Karchmer-Raz-Wigderson (KRW), [KRW95])**.** *Given two Boolean functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$, write $g \circ f : \{0,1\}^{nm} \to \{0,1\}$ to denote the* disjoint composition

$$(g \circ f)(x_1, \ldots, x_{nm}) = g\left(f(x_1, \ldots, x_n), f(x_{n+1}, \ldots, x_{2n}) \ldots, f(x_{nm-n+1}, \ldots, x_{nm})\right).$$

It is clear that we can get a formula upper bound for the depth of this combined function given formula upper bounds for the depth of f and g:

$$\text{depth}(g \circ f) \leq \text{depth}(f) + \text{depth}(g).$$

The KRW conjecture claims that this is the best we can do: that is, for any two nontrivial functions f and g, the formula depth of $g \circ f$ is more or less the same as the sum of the formula depths of f and g: $\text{depth}(g \circ f) \cong \text{depth}(f) + \text{depth}(g)$.

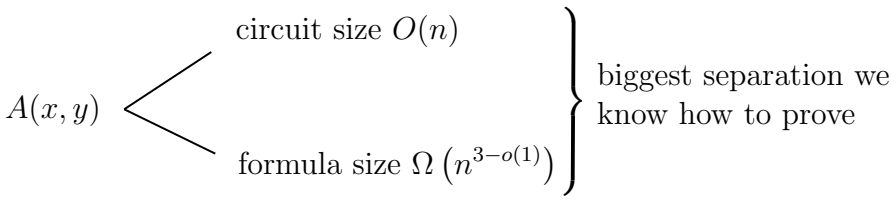
There's been a lot of progress on proving certain special cases of this conjecture, but it remains open in general. If it were true, we would get very strong formula size lower bounds (and super-polynomial circuit lower bounds), because there is a close connection between formula depth and size in general as we will see next.

Question: What does $\cong$ mean in the statement of the conjecture?
▷ The statement of the KRW conjecture is a bit vague and can be formulated in different ways depending on how much precision is desired. For instance, we could

say that "$\text{depth}(g \circ f) \cong \text{depth}(f) + \text{depth}(g)$" means $\text{depth}(g \circ f) = \Theta(\text{depth}(f) + \text{depth}(g))$. The point is that the formula depth of $g \circ f$ is (claimed to be) not much smaller or much larger than the sum of the formula depths of f and g.

2 Depth Versus Size for Formulas

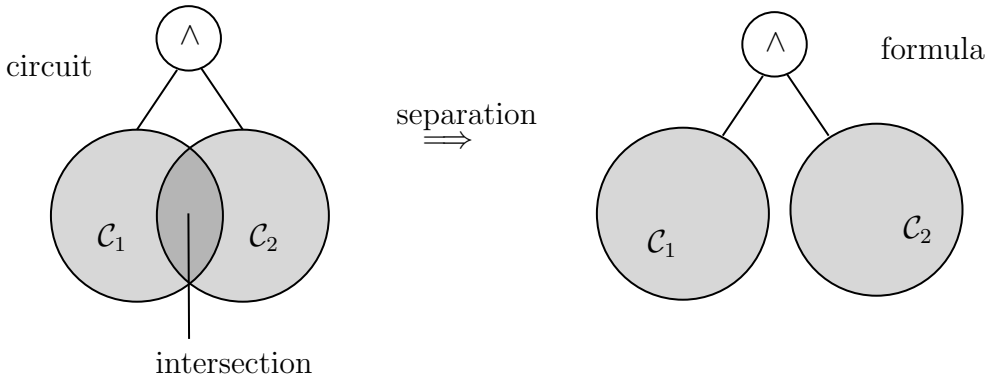
We already saw that Andre'ev's function A has



since the shrinkage exponent is $2 - o(1)$[1]. But if we're interested in depth, there is no gap for any function!

Claim 2. *If a Boolean function has a fanin-2, depth-d* AND/OR/NOT *circuit, then f has a fanin-2, depth-d* AND/OR/NOT *formula of size 2^d (and the converse also holds for the depth).*

Proof. Trivially, if a function f has a fanin-2 formula of depth d, then f has a fanin-2 circuit of depth d, since every formula is a circuit. For the forward direction, we can prove the claim by induction on the depth of the circuit.

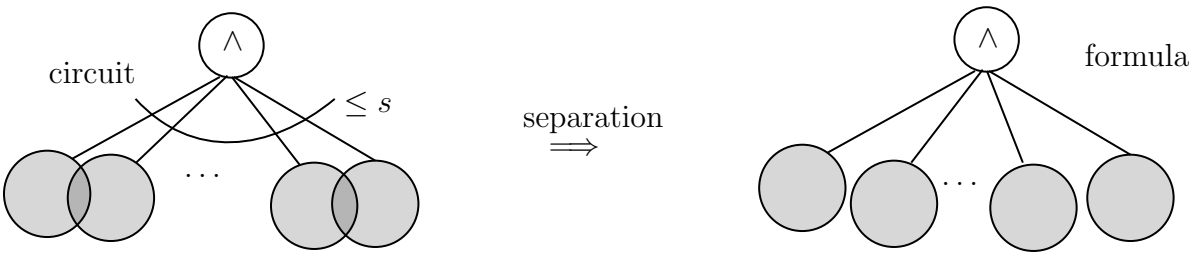


[1]We believe there are bigger separations; we believe that $\mathbf{P} \neq \mathbf{NC}^1$, and so we believe that there are functions which have polynomial-sized circuits but do not have superpolynomial-sized formulas, but we don't know how to prove this.

If f is a single variable, then f has a fanin-2 formula of depth 1 and size 1. Suppose that f has a fanin-2, depth-$(d-1)$ circuit. Then we can write f as $f = \mathcal{C}_1 \wedge \mathcal{C}_2$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are fanin-2, depth-$(d-1)$ circuits. By the inductive hypothesis, $\mathcal{C}_1$ and $\mathcal{C}_2$ have fanin-2, depth-$(d-1)$ formulas of size 2^{d-1}. Then following the picture, f has a fanin-2, formula of depth $d-1+1 = d$ and size $2^{d-1} + 2^{d-1} = 2^d$. ■

Related claim. If f has an unbounded fan-in circuit of depth d and size s, then f has an unbounded formula of depth d and size s^d, and the converse also holds for the depth.

Proof. The proof is similar to that of the previous claim:



Just apply the same inductive argument as before. ■

Here's a natural question:

Question: Do we really need this exponential blowup in size, or is there a smarter way to avoid it by any chance?

▷ In some settings, yes. Rossman [Ros14] showed that for some functions and for some sufficiently small d, there exist size-s depth-d circuits with no size $s^{o(d)}$ depth-d formula.

So what *do* we know about the relationship between formula depth and formula size?

Theorem 3 ([Spi71])**.** *For any Boolean function f,* $\text{depth}(f) = \Theta(\log \text{size}(f))$.

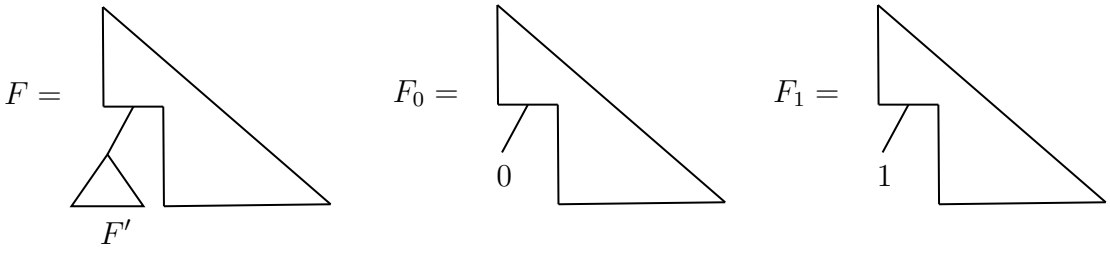
Proof. One direction is easy: given F a Boolean formula, we know that it is a binary tree by definition, and for every binary tree, the number of leaves is at most $2^{\text{depth}(F)}$. So $\text{size}(F) \leq 2^{\text{depth}(F)}$, and so $\log \text{size}(F) \leq \text{depth}(F)$. This holds for any formula, so it must hold for the optimal formula, and therefore $\log \text{size}(F) \leq \text{depth}(F)$ for any function F.

For the other direction, we want to show that $\text{depth}(F) \leq O(\log \text{size}(F))$. The following lemma will be key for us:

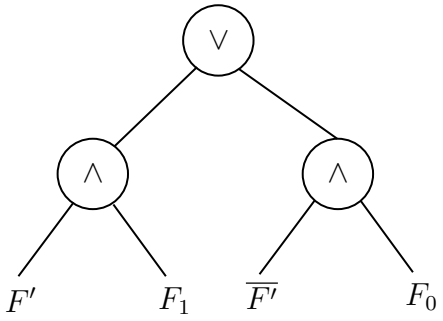
Lemma 4. *Let T be a rooted binary tree with s leaves. Then T has some subtree with s' leaves for some $s' \in [s/3, 2s/3]$.*

Proof. From a tree of size s, pick a particular node; this splits the tree into subtrees of size r and $s - r$. One of these is at least $s/2$. Pick that one and repeat this strategy of splitting it until we get a subtree in the range $[s/3, 2s/3]$. This has to work, because a tree of size at least $2s/3$ cannot be split into two subtrees both less than $s/3$. So we have a subtree of size s' in the range $[s/3, 2s/3]$. ■

Continuation of the proof of Theorem 3. Now fix an optimal formula F of size s for F. Let F' be its subformula of size $s' \in [s/3, 2s/3]$ (which exists via the lemma above). Let the subformula F_0 be F but with F' replaced by 0, and let the subformula F_1 be F but with F' replaced by 1.



Note that $\text{size}(F'), \text{size}(F_0), \text{size}(F_1) \in [s/3, 2s/3]$, because F' is a subformula of F, and F_0 and F_1 are merely modifications of F as it relates to F'. Now by Boolean logic, we have that $F = (F' \wedge F_1) \vee (\overline{F'} \wedge F_0)$.



In essence, we have "balanced" the formula F by replacing F' with 0 in one part and 1 in the other, and gained 2 more in depth by this process, since we can readily check that

$$\text{depth}(F) \leq 2 + \max\{\text{depth}(F'), \text{depth}(F_0), \text{depth}(F_1)\}.$$

Recursively apply the same balancing procedure to each of F', F_0, and F_1, and define $D(s)$ to be the maximum possible depth obtainable from this procedure over all size $\leq s$ formulas. Then we have that $D(s) \leq 2 + D(2s/3)$, and so

$$D(s) \leq O(\log s) = O(\log \text{size}(F)),$$

which is what we set out to prove. ■

3 Full-basis formulas

Until now, our discussion has been focused on Boolean formulas in the de Morgan basis with $\wedge, \vee$, and $\neg$ operations. However, *full-basis* formulas allow any two-variable binary gate $g : \{0,1\}^2 \to \{0,1\}$. It is easy to show that full-basis formulas are the same in power as gates that use only binary XOR ($\oplus$) and AND ($\wedge$) gates:

Claim 5. *Given a full-basis formula with ℓ variable leaves, there always exists an equivalent formula with ℓ variable leaves using only binary XOR ($\oplus$) and AND ($\wedge$) gates (and negations, which need not be at the leaves).*

Proof. Note that we can obtain all the binary functions with two variable leaves:

1. Negating variables, i.e. $\neg x_i \equiv x_i \oplus 1$.
2. Gates where two (out of four) of the input combinations map to 1's are either a single variable or the binary XOR function (or their negations).
3. Gates where (one or three, by way of negation) of the input combinations map to 1 can be constructed in the following way:

x	y	$\overline{x} \wedge \overline{y}$	$\overline{x} \wedge y$	$x \wedge \overline{y}$	$x \wedge y$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

So we can replace any full-basis formula with an equivalent formula using only binary XOR ($\oplus$) and AND ($\wedge$) gates. ■

What else happens in this model? Do we need to reprove everything? Fortunately, no:

- Shannon's $\Omega(2^n/\log n)$ bound still holds, as it was obtained via a counting argument on Boolean formulas;
- The $\Omega(n^{1.5})$ lower bound for PAR_n breaks down—we have full-basis formula size n for PAR_n since we can construct the tree of parities we saw in a previous lecture and compute PAR_i on all the i variables at each level.

In fact, Nečiporuk [Nec66] showed that a full basis formula for an explicit function over n variables has size $\Omega(n^2/\log n)$. We will prove a looser version of this result via Andre'ev-like method (but without using random restrictions).
Let $b = \log n$ and let $m = n/b = n/\log n$. Define a $2n$-variable function

$$\begin{aligned} A(x,y) &= A\left(x_1, x_2, \ldots, x_m, y_1, y_2, \ldots, y_n\right) \\ &= \underbrace{f_y(x_1 \wedge \ldots \wedge x_m, x_{m+1} \wedge \ldots \wedge x_{2m}, \ldots)}_{b \text{ blocks, } m \text{ variables per block}}. \end{aligned}$$

Claim 6. *Any full-basis formula for $A(x,y)$ must have size*

$$\geq \Omega\left(\frac{n^2}{\log n \log\log n}\right).$$

Proof. Let F be a minimum size formula for A with $\text{size}(F) = s$. Pick $\log n$ blocks of $n/\log n$ variables each thus: for $m = n/\log n$, define

$$\begin{aligned} B_1 &= x_1 x_2 \cdots x_m, \\ B_2 &= x_{m+1} x_{m+2} \cdots x_{2m}, \\ &\vdots \\ B_{\log n} &= x_{(m\log n - 1)m+1} x_{(m\log n-1)m+2} \cdots x_{m\log n}. \end{aligned}$$

Now select the least frequently occurring variable in each block (without loss of generality, say x_m in B_1, x_{2m} in B_2, and so on), and let restriction ρ be that which fixes all other variables equal to 1. This causes each block B_i to collapse to x_{im}, and so $A_{\restriction\rho}$ is equivalent to $f_y(x_m, x_{2m}, \ldots, x_{bm})$.
Nowe we can pick $y \in \{0,1\}^n$ so that f_y is any $(\log n)$-variable function. By Shannon's counting argument, we have that most f_y will require size $\Omega(2^{\log n}/\log\log n) = \Omega(n/\log\log n)$. Then F will have $\geq \Omega(n/\log\log n)$ occurrences of the variables x_m, x_{2m}, $\ldots, x_{bm}$, and so the total number of occurrences of $x_1, \ldots, x_n$ in F must be

$$\geq m \cdot \Omega(n/\log\log n) = \Omega\left(\frac{n^2}{\log n \log\log n}\right),$$

as desired. ∎

Question: Given an arbitrary (potentially non-binary) complete basis, does the same thing hold?
▷ No; there indeed exist complete bases that do not trivially convert to something like what we have seen, or are not equivalent to a de Morgan-type basis.

Some other known lower bounds for full-basis formulas include the following:

- The best lower bound for MAJ_n is $\Omega(n \log n)$ full-basis formula size (proved via an argument based on Ramsey theory);
- An argument via multi-party communication complexity gives $\Omega(n \log^2 n)$ for any explicit function.

These are both non-Andre'ev/non-Nečiporuk methods, and a bit involved.

4 Constant-depth circuits

Constant-depth circuits (which we will call $\mathbf{AC}^0$ or $\mathbf{AC}^0_{d,s}$ for a (depth, size) pair (d, s)) are a natural computation model in which the standard gates have unbounded fanin (i.e. can take any number of inputs). A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is in $\mathbf{AC}^0$ if there exists a polynomial $p(n)$, a constant d, and a circuit family $\{\mathcal{C}_n\}_{n=1}^{\infty}$ for the function f over $\wedge, \vee, \neg$ gates of unbounded fanin such that $\mathcal{C}_n$ is of size at most $p(n)$ and depth at most d (considered to be constant, although we will also give lower bounds for the case where d is a slow-growing function of n).[2]

Motivation. We can think of $\mathbf{AC}^0_{d,s}$ as a very general parallel computation model. Each level of the circuit can be thought of as a timestep, and thus a depth-d constant-depth circuit is like d-timesteps on processors that can only perform $\wedge, \vee, \neg$ operations. Sufficiently strong lower bounds for constant-depth circuits would also imply nontrivial lower bounds for more general circuit models—a classic example is due to Valiant:

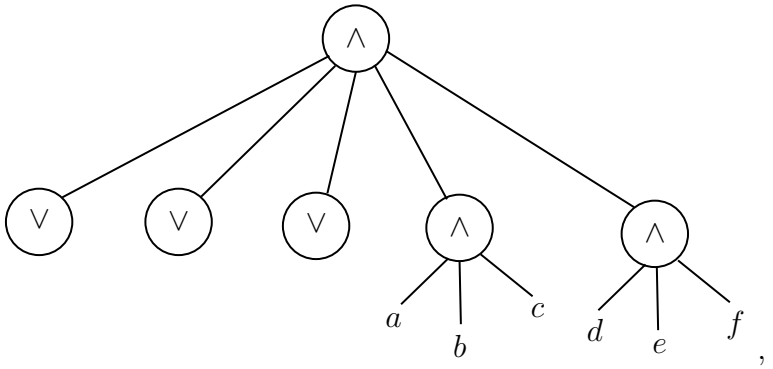
Theorem 7 (Valiant, [Val77])**.** *If a Boolean function $f : \{0,1\}^n \to \{0,1\}$ requires any depth-3 circuit to have size $2^{\omega\left(\frac{n}{\log \log n}\right)}$, then f cannot be computed by a circuit that has both $O(n)$ size and $O(\log n)$ depth.*

Strong enough lower bounds for depth-3 will hopefully tell us something nontrivial about richer models with logarithmic depth and linear size—although we don't have any of those results yet.

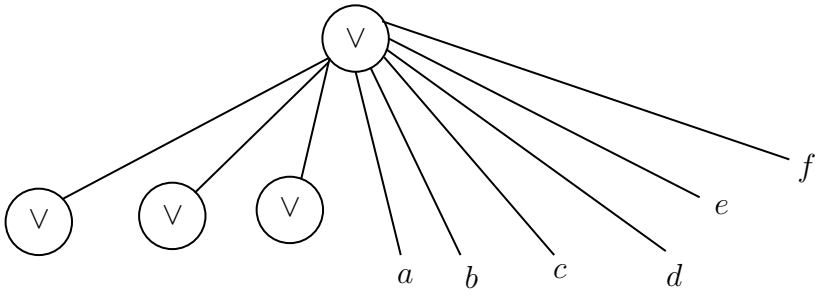
Let's now make two simplifying observations that help us clarify our thinking about $\mathbf{AC}^0$:

[2] Note that DNFs and CNFs are the $d = 2$ cases of $\mathbf{AC}^0$.

- Since d is small in some sense, we can view our size-s depth-d circuit as a size-s^d depth-d formula (via the claim we already saw) without suffering too much blowup. So we can consider only formulas instead, which are somewhat simpler because they don't reuse subcomputations.
- Given a formula, we can assume all negations are at the leaves, and that gates alternate between $\wedge$ and $\vee$. This is because it never makes sense to have a formula like



since it is equivalent to



no matter what the leaf functions are.

4.1 Intuition for the lower bound of constant-depth circuits

We will now give a high-level intuition for the lower bound of constant-depth circuits. We start by presenting function $f : \{0,1\}^n \to \{0,1\}$ that is plausibly "hard" for constant-depth circuits of depth $d = 2$. With some thought, we find that the parity function PAR_n is the ultimate hard function for DNFs.

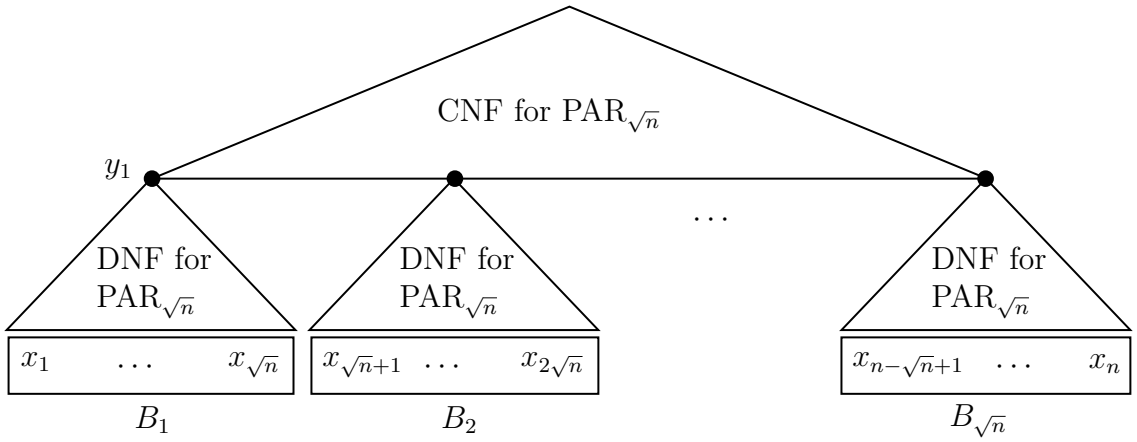
Claim 8. *Any (optimal) DNF $\mathcal{D}$ for* PAR_n *must have* 2^{n-1} AND *gates, and in fact, we must have one* AND *gate for each of the* 2^{n-1} *satisfying assignments of the input variables.*

Proof. Suppose the existence of some AND gate $\mathcal{A}$ in the DNF $\mathcal{D}$ that is unaffected by input i. As $\mathcal{D}$ is optimal, $\mathcal{A}$ is not identically 0, and thus there must exist some $x \in \{0,1\}^n$ such that $\mathcal{A}(x) = 1$, and $\mathcal{A}(y) = 1$ for all y that differ from x only in the i-th bit. But then $\mathcal{D}(x) = \mathcal{D}(y) = 1$ for all $y \in \{0,1\}^n$, and so $\mathcal{A}$ doesn't actually compute the parity function.

Now we show that $\mathcal{D}$ has at least 2^{n-1} AND gates. As every AND gate contains all the n variables, each gate must be satisfied by no more than 1 input string. To compute the parity function, we need at least one AND gate to be satisfied for each of the 2^{n-1} strings in $\mathrm{PAR}_n^{-1}(1)$. Thus, $\mathcal{D}$ must have at least 2^{n-1} AND gates. ■

For example, if $n = 5$ and the DNF $\mathcal{D}$ has $x_1\overline{x_2}x_3\overline{x_4}$ as a term; it accepts 10100 and 10101, but PAR_n is 1 on the former and 0 on the latter. The idea is then that each term in any DNF (especially an optimal one) must have length at most n and will accept exactly one satisfying assignment, and so we need 2^{n-1} terms. By the same argument, we can show that any (optimal) CNF for PAR_n must have 2^{n-1} OR clauses, and more generally, we have DNFs and CNFs for PAR_n of size 2^{n-1} and depth 2.

What about depth-3 circuits? As it turns out, we can compute PAR_n with a depth-3 circuit of size $\approx 2^{\sqrt{n}}$, and we will show that this is true via a divide-and-conquer type approach.



Divide n into $\sqrt{n}$-many $\sqrt{n}$-sized blocks, and:

- Use a DNF of size $\approx 2^{\sqrt{n}}$ to compute the parity $\mathrm{PAR}_{\sqrt{n}}$ on each block to obtain outputs $y_1, \ldots, y_{\sqrt{n}}$;

- Use a CNF[3] to compute $\text{PAR}(y_1, \ldots, y_{\sqrt{n}})$, and by so doing, effectively collapse the two layers of adjacent OR gates into a single layer, and so we have a circuit of size $(\sqrt{n}+1) \cdot 2^{\sqrt{n}}$ and depth 3.

Indeed it is not hard to generalise this idea:

Claim 9. *There exists a circuit of size* $\approx 2^{O\left(n^{1/(d-1)}\right)}$ *and constant-depth* d *for* PAR_n.

Proof. Left as a homework exercise. ■

In fact, this is the best possible upper bound obtainable for PAR_n. We can show that any constant-depth circuit for PAR_n must have size at least $2^{\Omega\left(n^{1/(d-1)}\right)}$. This is a result due to Håstad, and we will discuss it in the next section.

4.2 Key to lower bound of Håstad: his Switching Lemma

Now the proof of the upper bound is not hard; that of the lower bound proved in the 1980s, however, is one of the crowning jewels of complexity theory and is much less obvious. The main theorem due to Håstad is the following:

Theorem 10 (Håstad, [Has86])**.** *For* $d \lessapprox \frac{\log n}{\log \log n}$, *any* $\mathbf{AC}^0_{d,s}$ *circuit for* PAR_n *must have size* $2^{\Omega\left(1/\left(n^{d-1}\right)\right)}$.

From the 1980s, there was a lot of work focused on finding bounds on small depth circuits for functions such as PAR. Some of the major results progressed as follows:

- Ajtai [Ajt83], Furst, Saxe, and Sipser [FSS84] proved that $\text{PAR} \notin \mathbf{AC}^0$.
- Yao [Yao85] proved the first exponential lower bounds on parity via combinatorial methods, but Håstad introduced an entirely new technique, his *switching lemma*, to prove even stronger lower bounds. Håstad's method relies on purely combinatorial and probabilistic techniques.
- In the same decade, Razborov and Smolensky were introducing algebraic techniques to prove new results for polynomials over finite fields. Smolensky in particular showed [Smo87] that depth-d circuits with gates AND, OR, and MOD_p, where p is a prime, require at least $2^{n^{\Omega(1/(2d))}}$ gates to calculate MOD_r for any $r \neq p^m$. Yao's parity result is essentially a special case of this result.

[3] If we use a DNF for $\text{PAR}(y_1, \ldots, y_{\sqrt{n}})$, we would get the total circuit size as about $(\sqrt{n}+1) \cdot 2^{\sqrt{n}}$ and circuit depth 4. This has an extra level of depth, so it's not the smartest choice for us.

Before we present the switching lemma, why *could* Theorem 10 be true? We understand the situation clearly for $d = 2$: CNFs and DNFs need large circuits to compute parity. If we could make a depth d circuit into a CNF/DNF without too much size blowup, then we'd 'beat' Theorem 10. If the bottom two layers of the circuit look like a DNF (i.e. we have $\vee$ then $\wedge$ gates), and we could efficiently switch it to a CNF (i.e. $\wedge$ then $\vee$ gates), then we'd be in business, because the circuit would then collapse one level (via our assumption that the levels alternate between $\wedge$ and $\vee$). If we can do this without paying too much in size, we can essentially repeat the process $\approx d$ times and get a small DNF/CNF for parity (and thus beat Theorem 10).

So in some sense our question is the following:

> Can we convert the constant-depth circuit $\mathcal{C}$ for PAR into a DNF/CNF for PAR, without increasing its size too much?

If so, then we are done, by our very strong DNF/CNF lower bounds. The problem though is that a CNF with n terms can blow up to a DNF with $2^{n/2}$ terms. For example, consider the CNF

$$(x_1 \vee x_2) \wedge (x_3 \vee x_4) \wedge \cdots \wedge (x_{2k-1} \vee x_{2k}).$$

By de Morgan's laws, the DNF form of this CNF would take at least 2^k terms, and hence there is exponential blowup. Our goal is then to retain the same general ideas of the above approach, but to do so in a way that avoids this blowup. For this, we go back to random restrictions. If we randomly restrict some of the variables to take on fixed values, say 0 or 1 only, then we might imagine that a lot of the terms will become much simpler and smaller in size, and therefore might possibly lead to a DNF conversion that doesn't incur an exponential blowup. So we have hope: a random restriction "kills off" longer terms, so using random restrictions can help us convert small CNFs to small DNFs.

For $0 < p < 1$ the survival probability of a Boolean variable, let $\mathcal{R}_p$ be a distribution over restrictions $\boldsymbol{\rho} : [n] \to \{0, 1, *\}$ such that

$$\boldsymbol{\rho}(x_i) = \begin{cases} * & \text{with probability } p, \\ 0 & \text{with probability } \frac{1-p}{2}, \\ 1 & \text{with probability } \frac{1-p}{2}. \end{cases}$$

Why are these random restrictions useful? Note that if we hit a k-variable and with a random restriction $\boldsymbol{\rho} \sim \mathcal{R}_p$, then the expected number of surviving variables is $p \cdot k$

(where the survival probability p is thought of as being closer to 0 than it is to 1). But in fact something much stronger happens; with high probability,

$$\Pr[\text{doesn't become the constant 0}] = \left(p + \frac{1-p}{2}\right)^k = \left(\frac{1+p}{2}\right)^k,$$

which is exponentially small in k!

Now let's take a brief detour to discuss the relationship between DNFs, CNFs, and decision trees.

Claim 11. *Suppose that F is computed by a depth-d decision tree T. Then F is computed by a width-d CNF and also computed by a width-d DNF.*

Proof. We construct the DNFs and CNFs from the decision tree by constructing the terms and clauses out of the paths in the decision tree which reach the 0s and 1s of the functions, respectively. We will show how to construct width-d DNFs; the construction for CNFs is similar. Each term in the width-d DNF will correspond to a 1-path in the decision tree, and so the width-d DNF will be true if and only if a 1-path can be followed in the decision tree. Let T be a depth-d decision tree computing F, and let P be any path from the root of T to a leaf corresponding to a 1 of F. Let $x_1, \ldots, x_\tau$ be the literals appearing on the path P, where the literal x_i is negated if we move left at the vertex and positive otherwise. For each such path P, we add a term $x_1 \wedge x_2 \wedge \ldots \wedge x_\tau$ to the DNF. If an input x is accepted by the decision tree, then the term corresponding to the accepting path in the tree will also be accepted by the DNF, and the converse direction holds as well. Obviously, for each path P in the decision tree, $\tau \leq d$, and so the resulting DNF has at most d literals per term.

The construction for the CNF is entirely analogous, except that we instead consider the paths to the 0s of the decision tree, and add a clause with the negations of the literals followed on the paths. Intuitively, this width-d CNF is true if and only if none of the 0 paths are followed in the decision tree. ■

We now state Håstad's switching lemma:

Lemma 12 (Håstad's Switching Lemma, [Has86])**.** *Let $f(x_1, \ldots, x_n)$ be computed by a width-w DNF (or CNF). Let the decision tree depth of F be denoted by* DT-depth(f)*. Then for any $t \geq 1$, and for $0 < p < 1$,*

$$\Pr_{\boldsymbol{\rho} \sim \mathcal{R}_p} [\text{DT-depth}(f\restriction_{\boldsymbol{\rho}}) \geq t] \leq (7 \cdot p \cdot w)^t$$

Let's give a few immediate remarks:

- There is no dependence on the number of variables n or the size of f in the bound, and so the lemma is a very strong statement.
- The lemma also holds for CNFs, and the proof is essentially the same.
- Note that if $p \geq 1/7w$, then the statement essentially says nothing; all probabilities are bounded above by 1. If $p < 1/7w$, then the probability of a bad restriction, one that does not signficantly reduce the decision tree depth, is small. For example with $p = 1/14w$, the probability of a bad restriction is at most $1/2^t$.

References

[Ajt83] Miklós Ajtai. 11-formulae on finite structures. *Annals of pure and applied logic*, 24(1):1–48, 1983. 4.2

[BS90] Ravi B Boppana and Michael Sipser. The complexity of finite functions. In *Algorithms and complexity*, pages 757–804. Elsevier, 1990. 1

[FSS84] Merrick Furst, James B Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, 17(1):13–27, 1984. 4.2

[Has86] John Hastad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 6–20, 1986. 10, 12

[J+12] Stasys Jukna et al. *Boolean function complexity: advances and frontiers*, volume 5. Springer, 2012. 1

[KRW95] Mauricio Karchmer, Ran Raz, and Avi Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5:191–204, 1995. 1

[Nec66] Eduard I Neciporuk. A boolean function. In *Soviet Mathematics Doklady*, volume 7, pages 999–1000, 1966. 3

[Ros14] Benjamin Rossman. Formulas vs. circuits for small distance connectivity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 203–212, 2014. 2

[Smo87] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 77–82, 1987. 4.2

[Spi71] Philip Spira. On time-hardware complexity tradeoffs for boolean functions. In *Proceedings of the 4th Hawaii Symposium on System Sciences, 1971*, pages 525–527, 1971. 3

[Val77] Leslie G Valiant. Graph-theoretic arguments in low-level complexity. In *International Symposium on Mathematical Foundations of Computer Science*, pages 162–176. Springer, 1977. 7

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles. In *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pages 1–10. IEEE, 1985. 4.2