Lower Bounds **(CS294-152, UC Berkeley, Fall'18)**

PARITY $\notin$ AC^0

Notes for 10/8/18

Scribe: *Jonathan Shafer*

Contents

1 Introduction

In previous lectures we discussed P/poly – decision problems with a poly-sized circuit family over AND, OR, and NOT gates. In this lecture we focus on the class AC^0 ("Alternating Circuits"[1]), that captures the subset of these problems that have constant-depth circuit families. Specifically, we present one of the best known results in circuit complexity: Poly-sized circuits

[1] See section 3.3 below for the meaning of the name.

of constant depth cannot compute PARITY.

Definition 1.

$$\mathsf{AC}^0 = \Big\{ f : \{0,1\}^* \to \{0,1\} : \begin{array}{l} \textit{There exist a polynomial } p(n), \textit{ a constant } d, \textit{ and a circuit family} \\ \{C_n\}_{n=1}^{\infty} \textit{ for } f \textit{ over AND, OR and NOT gates of unlimited fan-in,} \\ \textit{such that } C_n \textit{ is of size at most } p(n) \textit{ and depth at most } d \end{array} \Big\}.$$

Note: we do not count NOT gates towards the depth or the size of a circuit.

AC^0 can do interesting things! Examples of some AC^0 functions, where $x_i \in \{0,1\}^n$ for all i:

- $f(x_1, x_2) = \mathbb{1}(x_1 \leq x_2)$.
- $f(x_1, \ldots, x_n) = \max\{x_1, \ldots, x_n\}$.
- $f(x_1, x_2) = x_1 + x_2$. Furthermore, AC^0 can even add $O(\log(n))$ n-bit numbers, see Vollmer (2013). But adding $O(n)$ n-bit numbers is not in AC^0 as we will see below, because that would compute PARITY.

Note: When the output is multi-bit, we consider each output bit to be its own function, and each of these functions is a member of AC^0.

Definition 2. PARITY *is the XOR function. That is, if* $x = (x_1, \ldots, x_n)$ *is an* n*-bit number, then* $\mathsf{PARITY}(x) = \sum_{i=1}^{n} x_i \pmod 2$.

Our objective in this lecture is to show the following lower bound:

Theorem 1.1 (Ajtai, 1983; Furst, Saxe, & Sipser, 1984)**.** $\mathsf{PARITY} \notin \mathsf{AC}^0$.

In fact, we will show a more explicit bound:

Theorem 1.2 (Håstad, 1986)**.** *For all* $d \geq 2$ *there exists* $\delta > 0$ *such that any circuit family* $\{C_n\}_{n=1}^{\infty}$ *of constant depth* d *that computes* PARITY *satisfies* $\mathsf{SIZE}(C_n) \geq 2^{\delta n^{1/(d-1)}}$.

Note: This bound is almost tight, as attested by Theorem 2.1 below.

2 Warm Up: The Case $d = 2$

For a start, we show the theorem holds for the case $d = 2$.

Lemma 2.1. *The depth-2 circuit size complexity of* PARITY *(and its negation) is* $2^{n-1} + 1$.

Proof of lemma. Let PARITY_n denote the parity function restricted to inputs of length n. Notice that by induction PARITY_n is unbiased, meaning that precisely half of the inputs of any fixed length n have an output value of 1. Thus, if $\{z_1, \ldots, z_{2^{n-1}}\} = \mathsf{PARITY}_n^{-1}(1)$, we may write

$$\mathsf{PARITY}_n(x) = \bigvee_{k=1}^{2^{n-1}} [x = z_k] = \bigvee_{k=1}^{2^{n-1}} \bigwedge_{i=1}^{n} [x_i = z_{k_i}] = \bigvee_{k=1}^{2^{n-1}} \left[\left(\bigwedge_{i:\, z_{k_i}=1} x_i \right) \wedge \left(\bigwedge_{i:\, z_{k_i}=0} \neg x_i \right) \right].$$

This is an OR over 2^{n-1} AND clauses, so it can be implemented using a depth 2 circuit with one OR gate and 2^{n-1} AND gates of fan-in n, which proves the upper bound.

For the lower bound, note that for $d = 2$, it suffices to show a lower bound for DNF and CNF circuits (because if there are two gates of the same type such that the output of one is an input to the other, then they can be combined into a single gate, producing a smaller circuit).[2] We now prove the desired lower bound for DNFs in the following claim, and an analogous argument can be made for CNFs.

Claim 2.1. *Let C be a DNF of minimal size that computes* PARITY_n*. Then every AND in C contains all n input variables (has x_i or $\neg x_i$ but not both for all $i \in [n]$), and C has $\geq 2^{n-1}$ AND gates.*

Proof of claim. Assume there exists some AND gate g in C that is not affected by some input i. We know $g \not\equiv 0$ because C is minimal. So there exists $x \in \{0,1\}^n$ such that $g(x) = 1$, and taking $\tilde{x}$ to be x with the ith bit flipped, we have that $g(\tilde{x}) = 1$ as well. But then $C(x) = C(\tilde{x}) = 1$, in contradiction to the assumption that C computes PARITY_n.

To see that C has at least 2^{n-1} AND gates, observe that since every AND gate contains all n variables, each AND gate can be satisfied by at most 1 input string. But in order to compute parity, at least one AND gate must be satisfied for each of the 2^{n-1} strings in $\mathsf{PARITY}_n^{-1}(1)$. Thus, 2^{n-1} AND gates are necessary and the proof is complete. □

Before proceeding to prove Theorem 1.2, we also want to see that it is almost tight:

[2] And of course, PARITY cannot be computed by a depth 1 circuit because it does not equal AND or OR (possibly modified with NOT gates).

Theorem 2.1. *There exists a constant δ such that for all d, PARITY_n and its negation have AC^0 circuits of depth d and size at most $2^{\delta n^{1/(d-1)}}$.*[3]

Proof. The case $d = 2$ was covered in Lemma 2.1. For $d > 2$, we explicitly construct a circuit for PARITY_n as follows. We start with a circuit that uses XOR gates of fan-in at most $n^{1/(d-1)}$. Clearly, if we consider an almost-balanced tree of XOR gates, in which all gates have precisely $n^{1/(d-1)}$ children except for the lowest layer of gates in which some might have less, then this circuit computes PARITY.

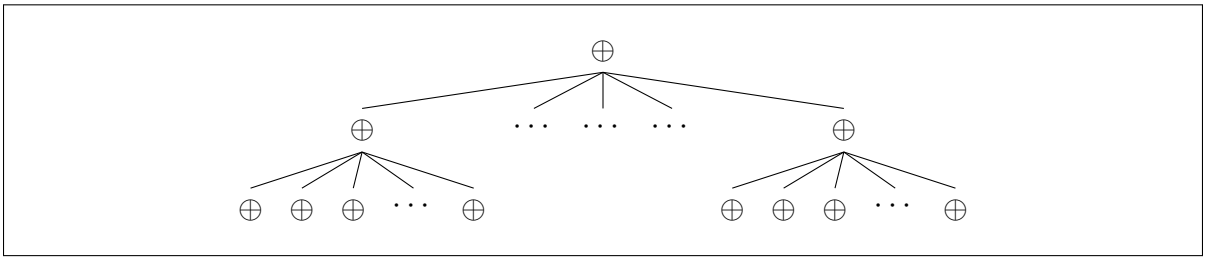

Figure 1: Balanced tree of XOR gates of depth 3.

The circuit has depth at most $d - 1$, and its size is at most

$$\sum_{k=0}^{d-2} (\text{XOR fan-in})^k = \sum_{k=0}^{d-2} \left(n^{1/(d-1)}\right)^k = \frac{n-1}{n^{1/(d-1)}-1} \leq n.$$

Next, we replace each XOR gate with a functionally-equivalent DNF or CNF circuit (the details of these circuits are discussed below). On even layers of the tree, we replace each XOR gate by a CNF that computes XOR or its negation, and on odd layers we do the same using DNF circuits. We replace the gates one layer at a time starting from the root layer, and at each step choose whether to use a circuit that computes XOR or its negation according to whether the output of the gate being replaced is negated or not in the layer above. In this manner, we avoid using NOT gates anywhere in the circuit except in the bottommost layer (between the inputs and the first AND or OR gates).

We now discuss the construction of the replacement circuits. Recall that each XOR gate had fan-in at most $n^{1/(d-1)}$, and from the proof of Lemma 2.1 there exists an explicit DNF circuit for PARITY_k of size $2^{k-1} + 1$, entailing that we can replace a XOR gate with a DNF circuit of size

$$2^{(n^{1/(d-1)})-1} + 1 = 2^{O(n^{1/(d-1)})}.$$

[3]There remains a gap between the lower bound of Theorem 1.2 and the upper bound here, in that the lower bound allows for a δ that may depend on d, while here δ is kept constant.

Moreover, by replacing the first input to this circuit with its negation, we obtain a DNF circuit of the same size that computes the negation of XOR. Consequently, negating these formulas and applying De Morgan's law yields CNF formulas of the same size that compute XOR and its negation.

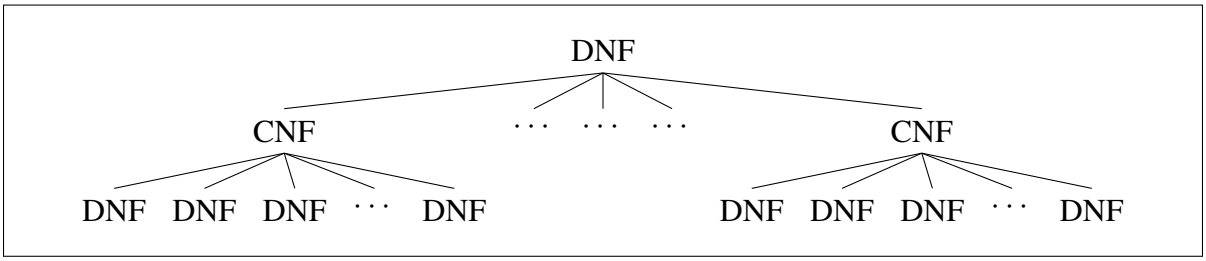

Figure 2: After replacing XOR gates with CNF and DNF circuits.

The last step in the construction is to notice that because it alternates between layers of CNF and DNF, the circuit now has two consecutive layers of AND gates, followed by two consecutive layers of OR gates, and so forth (except at the edges). By merging each set of consecutive layers of gates of the same type, we obtain a final circuit of depth d (which is 1 more than the original depth, because all layers except the root are merged with their parents).

To compute an upper bound on the size of the final circuit, we multiply the number of XOR gates in the original by the number of gates in each DNF and CNF, yielding that the final circuit size is at most

$$n \cdot 2^{O(n^{1/(d-1)})} = 2^{O(n^{1/(d-1)})},$$

as desired. □

3 Preliminaries

Here we introduce three main ingredients that will figure predominantly in the proof of the general case (Theorem 1.2), and explain how each of them contributes to the argument.

3.1 Random Restrictions

Imagine performing the following "experiment" on the PARITY function, and separately on an AC^0 circuit. Pick an input variable x_i at random, and fix all other inputs to $\{0, 1\}$ values chosen independently and uniformly.

What happens to the PARITY function? It simplifies to either

- $f(x_i) = x_i$, with probability $\frac{1}{2}$, or
- $f(x_i) = \neg x_i$ with probability $\frac{1}{2}$.

In contrast, Ajtai (1983), Furst et al. (1984) and Håstad (1986) show that, with high probability, performing the same operation on a small AC^0 circuit causes the circuit to simplify to a constant function (the papers differ in how small the circuit needs to be in order to achieve this effect). The conclusion is that small AC^0 circuits cannot compute the PARITY function.

With this motivation in mind, we now define random restrictions.

Definition 3. *Let $f : \{0,1\}^n \to \{0,1\}$ and $v \in [n]$. A random restriction ρ of v variables of f, denoted $f|_\rho$, is a random variable whose value is a function $\{0,1\}^{n-v} \to \{0,1\}$ chosen as following. Uniformly select some subset $I \in \binom{[n]}{v}$, and for each $i \in I$, set the input x_i to a bit chosen uniformly and independently. $f|_\rho$ is the remaining function over the unfixed input bits.*

3.2 Switching Lemma

The proof we present below for Theorem 1.2 does not actually apply a random restriction to all variables except one. Instead, it obtains a quantitatively better result by applying a sequence of smaller random restrictions until the circuit is of depth 2, at which point we can derive a lower bound from Claim 2.1.

For this we need the following famous lemma, which shows that applying random restrictions indeed makes the depth decrease with high probability.

Definition 4. *A k-CNF is an AND of ORs of literals and their negations, where each OR has fan-in at most k. k-DNF is defined analogously, with OR of ANDs.*

Lemma 3.1 (Switching Lemma, Håstad, 1986)**.** *Let $p \in (0, \frac{1}{5})$. Suppose f is a k-CNF or k-DNF over n variables. Pick a random restriction ρ of $(1-p)n$ variables of f. Then for all t,*

$$\mathbb{P}\left[f|_\rho \text{ is equivalent to a } t\text{-DNF and also to a } t\text{-CNF}\right] > 1 - (10pk)^t.$$

That is, after applying a random restriction, we can "switch" a k-CNF to a t-DNF and vice versa, with high probability.

Example. Take $p = \frac{1}{20k}$. The Lemma says: if you randomly set all but np of the variables in a k-CNF formula, the remaining formula on $\frac{n}{20k}$ variables is equivalent to a t-DNF formula, with probability at least $1 - 2^{-t}$. So, the remaining formula has very high probability of being equivalent to a 100-DNF for example.

Why would this be useful? Suppose I have a depth-3 circuit that is an AND of ORs of ANDs, and the ANDs have fan-in k. We can think of this as an AND of k-DNFs. We can apply the switching lemma to the k-DNFs, and convert all of them to t-CNFs. This means the depth-3 circuit is equivalent to an AND of AND of ORs of fan-in t – which is just a t-CNF itself! We have successfully reduced the depth by 1.

3.3 AC^0 Normal Form

An important property of AC^0 circuits is that they admit a certain normal form, in which the circuit alternates repeatedly between layers of AND gates and layers of OR gates.

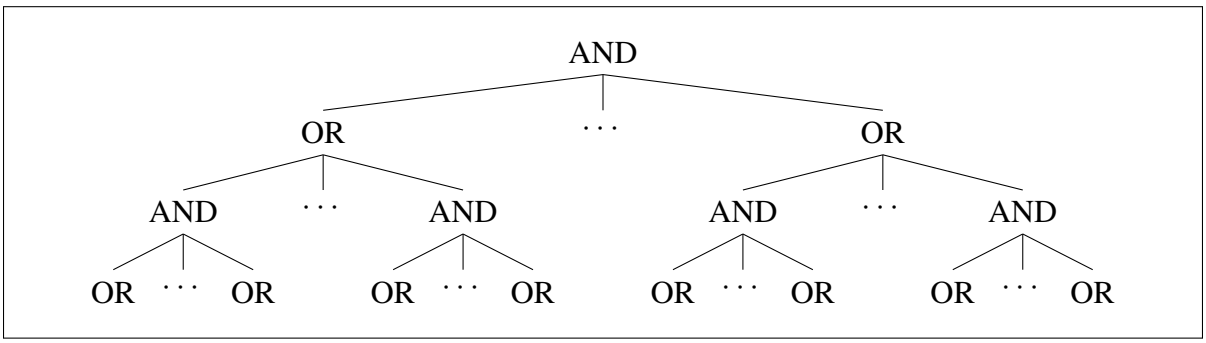

Figure 3: Alternating layers of AND and OR.

This form is perfect for using the switching lemma: We can apply a random restriction and then invoke the switching lemma to switch the bottom two layers from being, say, ANDs of ORs, to being ORs of ANDs. At this point the newly formed ORs can be merged with the untouched layer of ORs above them, making the whole circuit decrease its depth by 1. We may apply this process repeatedly to obtain a circuit of depth 2, for which we have a lower bound.

Definition 5. *We say that a circuit is in alternating circuit (AC) normal form if it satisfies the following conditions:*

- *The gates and inputs can be divided into layers, such that layer 0 contains only input bits and their negations, and for all $k > 0$, layer k contains only gates for which all the inputs are elements from layer $k - 1$.*
- *Except for layer 0, even layers contain only AND gates and odd layers contain only OR gates, or vice versa.*
- *The circuit is a tree. (All AND and OR gates have fan-out 1. Inputs and their negations may have larger fan-outs.)*

The name "AC^0" is justified by the following claim, which says that every AC^0 function has a circuit family in AC normal form.

Claim 3.1. *Let f be a function in AC^0 with a family of circuits $\{C_n\}_{n=1}^{\infty}$ of depth d and size at most $p(n)$. Then there exists a family of circuits $\{C'_n\}_{n=1}^{\infty}$ in AC normal form of depth d and size at most $2p(n) + 4d(p(n) + n)^2$ that computes the same function.*

Proof of claim. We perform a sequence of steps on each circuit C_n to transform it to an appropriate C'_n in AC form. Let s be the size of C_n.

- **Move all NOT gates to the bottom.** To achieve this, we first ensure there is a NOT gate that computes $\neg x_i$ for each input bit x_i, by adding at most n NOT gates. Next, we traverse a topological ordering of the AND and OR gates of the circuit $g_1, g_2, \ldots, g_s$ such that g_i depends only on input bits and their negations and on gates g_j with $j < i$. For each g_i in the list, we add a new gate g'_i that computes $\neg g_i$ using De Morgan's law.
  - If $g_i = \text{AND}(x_{i_1}, \ldots, x_{i_r}, g_{j_1}, \ldots, g_{j_t})$, then $g'_i = \text{OR}(\neg x_{i_1}, \ldots, \neg x_{i_r}, g'_{j_1}, \ldots, g'_{j_t})$.
  - If $g_i = \text{OR}(x_{i_1}, \ldots, x_{i_r}, g_{j_1}, \ldots, g_{j_t})$, then $g'_i = \text{AND}(\neg x_{i_1}, \ldots, \neg x_{i_r}, g'_{j_1}, \ldots, g'_{j_t})$.

  At this point we may remove all the NOT gates from the circuit except those of the form $\text{NOT}(x_i)$, because whenever $\text{NOT}(g_i)$ is used (as an input to some gate, or as the output of the whole circuit), we may use g'_i instead. Note that we have added at most s gates of AND and OR in this step.
- **Remove consecutive gates of the same type.** If there exists an OR gate that is an input to another OR gate, we replace the two with a single OR gate over the union of their inputs. In the same manner, we merge consecutive AND gates. We perform these operations repeatedly until there are no more consecutive gates of the same type. Notice that the size and depth of the circuit cannot increase during this step.
- **Enforce a layered structure.** We say that all inputs and their negations belong to layer 0, and for each AND and OR gate g, the layer of g is the number of wires in the longest path from any layer 0 element to g (not counting incoming wires to NOT gates). Because of the previous step, all gates in each layer are of the same type, with the layers alternating between AND and OR. Whenever there is an element a and a gate g such that a is an input to g and the difference in layers between a and g is $k > 1$, we replace the wire connecting them with a sequence of $k - 1$ alternating AND and OR gates of fan-in 1 (maintaining the alternation between layers of AND gates and layers of OR). We repeat this operation until every gate of layer ℓ takes only elements from layer $\ell - 1$ as input. Note that before this step, the circuit contained at most $2s$ gates and $2n$ inputs and negated inputs, making for at most $(2s + 2n)^2$ wires that might need to be replaced. Because k is always at most d, we have added at most $d(2s + 2n)^2$ gates in this step, for a total circuit size upper bound of $2s + 4d(s + n)^2$.

This completes the proof of the claim. □

4 Proof of $\mathsf{PARITY} \notin \mathsf{AC}^0$

We now have all the ingredients to prove Theorem 1.2, under the assumption that the switching lemma is true.

Proof of Theorem 1.2. Let C_n be a circuit that computes PARITY_n and let C'_n be an equivalent circuit in AC normal form of size s and depth d.

We apply a sequence of steps with the objective of transforming C'_n into a depth 2 circuit.

Step 1: Ensure the bottom gates have small fan-in. Assume w.l.o.g. that the bottom layer consists of AND gates (the case of OR gates is symmetric). Initially, these gates can have any fan-in. Consider each of them to be a 1-CNF, and apply the switching lemma to each of them, with $p = \frac{1}{20}$, $t = \log(s)$, and $k = 1$: For each gate, with probability strictly greater than $1 - \left(\frac{10}{20}\right)^{-\log s} = 1 - \frac{1}{s}$, the gate can be converted to a $\log(s)$-DNF. From the union bound, there is a positive probability that after fixing all but a $\frac{1}{20}$-fraction of the inputs, all the layer 2 gates are equivalent to $\log(s)$-DNFs simultaneousy. Therefore, there exists some specific restriction of that size that satisfies this requirement. We apply this restriction, and then, by merging the OR gates of the new $\log(s)$-DNFs into the layer of ORs above them, we obtain a new circuit C^0_n in AC normal form with the same depth as C'_n. In C^0_n the lowest layer consists of gates with fan-in at most $\log(s)$, there are at most s gates above the lowest layer, and there are $\frac{n}{20}$ inputs.

Step 2: Reduce the depth to 2. W.l.o.g. assume layer 1 of C^i_n contains AND gates (the case of OR gates is symmetric). View each gate in layer 2 as forming a $\log(s)$-DNF together with its children from layer 1. Apply a random restriction to a all but a p-fraction of the inputs, for $p = \frac{1}{20\log(s)}$. Again, by the switching lemma, with probability strictly greater than $1 - (10 \cdot (1/20\log(s)) \cdot \log(s))^{-\log(s)} = 1 - \frac{1}{s}$, each of these $\log(s)$-DNFs can be switched after the restriction to a $\log(s)$-CNF without altering its functionality. And from the union bound, there exists a restriction in which this holds for all DNFs simultaneously. By applying that restriction, switching to CNFs and merging layer 2 with layer 3, we obtain circuit C^{i+1}_n.

We repeat this operation $d - 2$ times, yielding C^{d-2}_n. Observe that for every i, C^i_n is of depth $d - i$, the gates of the lowest layer have fan-in at most $\log(s)$, there are at most s gates above the lowest layer, and there are $\frac{n}{20}p^i$ inputs.

In particular, C^{d-2}_n is a depth 2 circuit that calculates PARITY on $\frac{n}{20}p^{d-2}$ inputs, with bottom fan-in at most $\log(s)$. From the proof of Lemma 2.1 (Claim 2.1), we know that every depth 2

circuit for parity must have a bottom fan-in that equals the number of inputs. Thus,

$$\frac{n}{20}\left(\frac{1}{20\log(s)}\right)^{d-2} = (\text{bottom fan-in}) \leq \log(s),$$

entailing that

$$s \geq 2^{\left(\frac{1}{20}n^{1/(d-1)}\right)}. \tag{1}$$

This lower bound applies to C'_n, the circuit in AC normal form. Let $S = \mathsf{SIZE}(C_n)$. We know from Claim 3.1 that

$$s \leq 2S + 4d(S+n)^2 \leq 10dS^2$$

and so

$$\mathsf{SIZE}(C_n) \geq \sqrt{\frac{1}{10d}2^{\left(\frac{1}{20}n^{1/(d-1)}\right)}} = 2^{\left(\frac{1}{10}n^{1/(d-1)} - \frac{1}{2}\log(10d)\right)}.$$

Finally,

$$\mathsf{SIZE}(C_n) \geq 2^{\left(\delta n^{1/(d-1)}\right)}$$

for δ that depends on d, as desired. □

Remark. *The constant of* 20 *that appears in (1) is not tight, and can be made smaller using other switching lemmata. Ryan is not sure whether it can be replaced with* $(1+\varepsilon)$ *for all* $\varepsilon > 0$*. That is the case at least for* $d = 2$ *and* $d = 3$ *(see Paturi, Pudlák, & Zane, 1997 for the latter). The constants that result from the conversion between the original and the AC normal circuits can definitely be made tighter.*

5 Proof of the Switching Lemma

We now complete the proof of $\mathsf{PARITY} \notin \mathsf{AC}^0$ by proving the switching lemma, on which we relied above. We will actually prove a slightly stronger variant:

Lemma 5.1 (Switching Lemma – Alternative Formulation)**.** *Let* $p \in (0, \frac{1}{2})$*. Suppose* f *is a* k*-CNF or* k*-DNF over* n *variables. Pick a random restriction* ρ *of* $(1-p)n$ *variables of* f*. Then for all* t*,*

$$\mathbb{P}\left[f|_\rho \textit{ is equivalent to a } t\textit{-DNF and also to a } t\textit{-CNF}\right] > 1 - \left(\frac{8pk}{1-p}\right)^t.$$

Note that the only difference is in the parameters, and that this statement implies our original switching lemma. In particular, when $p < \frac{1}{5}$, the current bound of $1-(8pk/(1-p))^t$ is greater than the the original bound of $1-(10pk)^t$.

The original proof was complicated. The alternative one presented here is due to Razborov, and is a very clever counting argument. In general, the switching lemma and its proof remain very important in the modern study of circuit complexity. There are still many questions being answered about AC^0, using new switching lemmata (e.g., see the work of Ben Rossman).

5.1 Reduction to a combinatorial claim

We demonstrate that in order to prove Lemma 5.1, it suffices to prove the following.

Claim 5.1 (Combinatorial Claim)**.** *Using the notation of Lemma 5.1, let $v = (1-p)n$ be the number of variables to be fixed, and let R_m denote the set of all restrictions of m variables out of n. Then*

$$|\{\rho \in R_v : \ f|_\rho \textit{ is not equivalent to a } t\textit{-CNF}\}| \leq |R_{v+t}| \cdot (4k)^t.$$

To see that Claim 5.1 entails the switching lemma (Lemma 5.1), observe that the total number of possible restrictions is $|R_v| = \binom{n}{v}2^v$ (because there are $\binom{n}{v}$ ways to choose which set of v variables to fix, and for each set there are 2^v ways to assign values to the variables in the set). Now a straightforward calculation yields:

$$\begin{aligned}
\mathbb{P}\left[f|_\rho \text{ is not equivalent to a } t\text{-CNF}\right] &= \frac{|\{\rho \in R_v : \ f|_\rho \text{ is not equivalent to a } t\text{-CNF}\}|}{|R_v|} \\
&\leq \frac{|R_{v+t}| \cdot (4k)^t}{|R_v|} \\
&= \frac{\binom{n}{v+t}2^{v+t} \cdot (4k)^t}{\binom{n}{v}2^v} \\
&= \frac{\binom{n}{v+t}2^t(4k)^t}{\binom{n}{v}} \\
&= \frac{\binom{n}{(1-p)n+t}(8k)^t}{\binom{n}{(1-p)n}} \\
&= \frac{((1-p)n)!(pn)!}{((1-p)n+t)!\,(pn-t)!}(8k)^t \\
&= \frac{(pn-t+1)(pn-t+2)\cdots(pn)}{((1-p)n+1)((1-p)n+2)\cdots((1-p)n+t)}(8k)^t \\
&< \left(\frac{pn}{(1-p)n}\right)^t (8k)^t \\
&= \left(\frac{8pk}{1-p}\right)^t,
\end{aligned}$$

as desired.

5.2 Strategy for proving the combinatorial claim

Let $B \subseteq R_v$ denote the set of "bad restrictions" that appears in the LHS of the inequality in the combinatorial claim. Consider the set $S = R_{v+t} \times \{0,1\}^\ell$ where $\ell = t\log(4k)$. We can think of S as being a set of all ordered pairs (ρ, s), where ρ is a restriction of $v+t$ variables, and s is a binary string of length ℓ. Notice that the expression on the RHS of the combinatorial claim is the cardinality of S. Therefore, to prove the combinatorial claim, it suffices to construct a one-to-one function $h : B \to S$. We think of this as constructing an encoding that encodes a restriction $\rho \in R_v$ using a pair (ρ', s), where s is an "advice string".

Consider the cardinalities of the two sets:

$$|R_v| = \binom{n}{v} 2^v; \quad |R_{v+t}| = \binom{n}{v+t} 2^{v+t},$$

where $v \geq \frac{n}{2}$. As illustrated in the following graph, the first set may be larger or smaller then the second one, depending on the exact values of v and t.

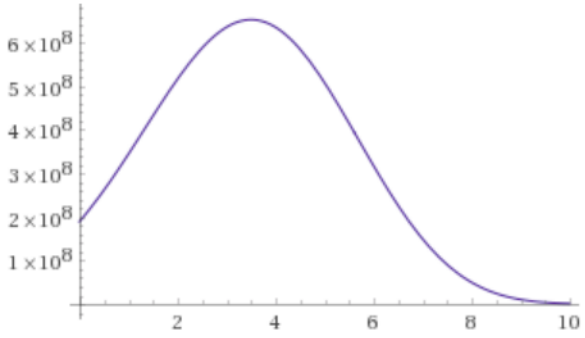

Figure 4: Graph of the function $f(m) = \binom{20}{10+m} 2^{10+m}$. Source: WolframAlpha.

Of course, if $|R_{v+t}| \geq |R_v|$ then a one-to-one function $R_v \to R_{v+t}$ exists and there is nothing to prove. The rest of the proof will be concerned with the case $|R_{v+t}| < |R_v|$, where we will use the advice string s to construct a one-to-one function h as above.

The construction will use the notion of a *max-term*, to be defined in the next section.

5.3 Max-terms and t-CNFs

The crucial property about "not being a t-CNF" that we will use to construct the encoding can be expressing in the following language:

Definition 6. *Let $g : \{0,1\}^n \to \{0,1\}$ be some function. A max-term for g of size m is a restriction ρ of m variables such that*

- $g|_\rho \equiv 0$.
- *ρ is **minimal** with respect to set containment, i.e., there exist no restriction ρ' such that $g|_{\rho'} \equiv 0$ and the fixed variables of ρ' are a strict subset of the fixed variables of ρ.*

Claim 5.2. *If f is not equivalent to a t-CNF, then there exists a max-term of f of size at least $t+1$.*

Proof of Claim 5.2. We show that if every max-term for f has size at most t then f is equivalent to a t-CNF: Let M be the set of max-terms for f. For each $\rho \in M$, write ρ as the assignment $(x_i = b_{\rho,i})_{i \in I_\rho}$ for some set of indices $I_\rho \subseteq [n]$ and bits $b_{\rho,i}$. Consider the following formula.

$$F(x) = \bigwedge_{\rho \in M} \bigvee_{i \in I_\rho} \mathbb{1}\left(x_i \neq b_{\rho,i}\right).$$

We argue:

- **The formula is a t-CNF.** It is a CNF because it is and AND of ORs of inequalities, and each inequality can be written as an input variable or its negation. Furthermore, each OR clause has fan-in at most t, from the assumption that all max-terms are of size at most t.
- **It computes f.** We show that for all $a \in \{0,1\}^n$, $F(a) = 0 \iff f(a) = 0$.

  $\Longrightarrow$. Assume $F(a) = 0$. Then there exists $\rho \in M$ such that $\vee_{i \in I_\rho} \mathbb{1}\left(a_i \neq b_{\rho,i}\right) = 0$. Then a agrees with a restriction ρ that is a max-term, so $f(a) = 0$.

  $\Longleftarrow$. Assume $f(a) = 0$. Then viewing a as a restriction where $(x_i = a_i)_{i \in [n]}$, we see that $f|_a \equiv 0$. Therefore, there exists some $\rho \in M$ that agrees with a (either a itself is a minimal assignment that forces the output to be 0, or else some subset of a is). Then the clause $\vee_{i \in I_\rho} \mathbb{1}\left(a_i \neq b_{\rho,i}\right) = 0$ for this ρ, making $F(a) = 0$.

We conclude that if f is not equivalent to a t-CNF, it must have a max-term of size at least $t+1$. □

5.4 Constructing the encoding restriction: $\rho \mapsto \rho'$

Write f as a k-DNF

$$F = \bigwedge_{i \in I} C_i,$$

where each C_i is an AND clause over at most k literals (input variables and their negations).

Recall that we want to identiy a mapping $\rho \mapsto (\rho', s)$, where $\rho \in R_v$ is "bad" (making $f|_\rho$ not equivalent to a t-CNF), and $\rho' \in R_{v+t}$. We use Claim 5.2 to build this mapping as following. Because $f|_\rho$ is not equivalent to a t-CNF, the claim entails that $f|_\rho$ has a max-term π of size at least $t+1$. Notice that the variables restricted by π are disjoint from those restricted by ρ, so the "combined" restriction $\rho \cup \pi$ in which both restrictions are applied simultaneously is well defined. Furthermore,

- $f|_{\rho\cup\pi} \equiv 0$.
- $f|_{\rho\cup\pi'} \not\equiv 0$ for all $\pi' \subsetneq \pi$ (i.e., all π' that agree with π and fix only a strict subset of the variables fixed by π).

Assume π assigns bit $x_i = b_i$ for all i. We define ρ' using an iterative process, in such a way that ρ' will contain all the assignments of ρ, together with exactly t additional assignments that came from π.

$\pi' \leftarrow \varnothing$ (will hold a sub-restriction of π)
$\rho' \leftarrow \rho$ (the output being constructed)
While $|\rho'| < v + t$: (want output to be of size $v + t$)
    Find C_j with minimal j such that $C_j|_{\rho\cup\pi'} \not\equiv 0$
    $I \leftarrow \{i : x_i$ or $\neg x_i$ appear in C_j, and x_i is not set by $\pi'\}$ (variables to be added to the restrictions)
    Remove elements from I until $|I| \le v + t - |\rho'|$ (ensure output size $\le v + t$)
    Find bits $\{b'_i : i \in I\}$ such that $C_j|_{\rho\cup\pi'\cup\{b'_i\}} \not\equiv 0$
    $\pi' \leftarrow \pi' \cup \{(x_i = b_i) : i \in I\}$ (make $C_j|_{\rho\cup\pi'} \equiv 0$)
    $\rho' \leftarrow \rho' \cup \{(x_i = b'_i) : i \in I\}$ (make $C_j|_{\rho\cup\rho'} \not\equiv 0$)

Output ρ'

Notice that in every iteration of the loop, π' is a strict sub-assignment of π, so $f|_{\rho\cup\pi'} \not\equiv 0$, and so a suitable clause C_j always exists. Moreover, $C_j|_{\rho\cup\pi'} \not\equiv 0$, and yet we know that $C_j|_{\rho\cup\pi} \equiv 0$. Consequently, π must contain some assignments to variables of C_j not yet set in π', so at each iteration $|\pi'|$ will strictly increase.

Example. *Suppose in some iteration $C_j = (x_1 \wedge x_2 \wedge \neg x_3 \wedge x_4)$, and suppose that the variables set in π but not in π' are $\{x_1, x_3\}$. Then $\rho\cup\pi$ sets $x_1 = 0$ or $x_3 = 0$ or both (because it zeros the clause), and those assignments will be added to π', and the assignment ($x_1 = 1$ and $x_3 = 0$) will be added to ρ'.*

5.5 Constructing the encoding advice: $\rho \mapsto s$

Recall that ρ' contains all the assignments of ρ, together with exactly t additional assignments that came from π. If we only see the restriction ρ', we don't know which assignments in it came from ρ and which came from π, so we can't reconstruct ρ. We will use the additional string s precisely for this: To identify which variables in the assignment came from ρ.

Naïvly, if we simply specify the indices of the variables that came from ρ, we would need $\Omega(\log n)$ bits per index. The following solution uses additional structure in order to identify all the indices with an advice string of only $\ell = t \log(4k)$ bits, which is much better.

For $i = 1, 2, \ldots$, let C_{j_i} and I_i be the clause and index set, and let π'_i, ρ'_i be the assignments added to π' and ρ' in the ith iteration of the loop. By definition, C_{j_1} is the first (minimal index) clause that is not zeroed by ρ. Interestingly, we have constructed ρ' in a way that ensures that C_{j_1} is also the minimal clause not zeroed by ρ'. To see this, note that $\rho' = \rho \cup \rho'_1 \cup \rho'_2 \ldots$, and

- By definition, ρ does not zero C_{j_1}.
- By definition, ρ'_1 does not zero C_{j_1}
- No other ρ'_q assigns variables in C_{j_1} for $q \neq 1$. This is true because in the first iteration, I contains indices of all the variables that belong to C_{j_1} and are set by π, and in all subsequent steps I does not contain indices of any variables already set in π' during previous iterations.

Hence, given ρ', we can identify C_{j_1} by finding the first clause not zeroed by ρ'.

More generally, C_{j_i} is the first clause not zeroed by

$$\rho \cup \pi'_1 \cup \cdots \cup \pi'_{i-1} \cup \rho'_i \cup \rho'_{i+1} \cup \cdots \tag{2}$$

Thus, given ρ' and $\pi'_1, \ldots, \pi'_{i-1}$ we can construct the restriction in (2) by overriding the assignments of ρ' with those of $\pi'_{<i}$, and identify C_{j_i} as the first unzeroed clause.

We use this structure to get a compressed representation of the indices of the variables that are in ρ' but not in ρ. Each of the variables belongs to C_{j_i} for some i. Hence, instead of representing the index using $O(\log(n))$, we can use just $O(\log(k))$ bits if we know the relevant C_{j_i}, because f is a k-DNF. We will use an additional bit for each variable to represent its value in π'_i, because we need π'_i in order to find $C_{j_{i+1}}$.

Accordingly, the advice string is constructed as following: There is one "block" of bits for each iteration of the loop. Each block i encodes assignments of π'_i in iteration i as follows. For each assignment, the index within the clause C_{j_i} of the variable is encoded using $\log k$ bits, followed by a bit specifying the value of the variable, followed by a control bit which is 0 iff the variable is the last one in the block.

The total length of s is thus

$$\sum_i |I_i|(\log(k)+2) = t(\log(k)+2) = t\log(4k),$$

where the first equality holds because $\sum_i |I_i| = t$.

5.6 Decoding (ρ', s)

Decoding the pair (ρ', s) works as following. First, C_{j_1} is identified using F and ρ', by virtue of being the first clause not zeroed by ρ'. Then, π'_1 is read from the first block of s. Subsequently, at iteration i, C_{j_i} is identified using F, ρ' and $\pi'_{<i}$ as being the first clause not zeroed by (2), and then π'_i is read from the ith block of s. Once all restrictions π'_i have been recovered (the end of s has been reached), we obtain ρ by simply removing from ρ' the set of t assignments to variables that appear in any of the π'_i.

This construction completes the proof of Claim 5.1, and hence of $\mathsf{PARITY} \notin \mathsf{AC}^0$. ■

References

Ajtai, M. (1983). $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, *24*(1), 1–98.

Furst, M., Saxe, J. B., & Sipser, M. (1984). Parity, circuits, and the polynomial-time hierarchy. *Mathematical systems theory*, *17*(1), 13–27.

Håstad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the eighteenth annual acm symposium on theory of computing* (pp. 6–20).

Paturi, R., Pudlák, P., & Zane, F. (1997). Satisfiability coding lemma. In *Foundations of computer science, 1997. proceedings., 38th annual symposium on* (pp. 566–574).

Vollmer, H. (2013). *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media.