Lower Bounds for ACC Circuits

Samuel Schlesinger

March 9, 2020

Abstract

I discuss recent breakthroughs in the theory of circuit lower bounds.

Introduction

Circuit complexity has developed for a few years now, and very recently there have been a couple of major advances [**?**] [**?**], the background of which I spent the semester studying. These results each hack into **ACC**, the class of circuits of constant depth constructed via unbounded fan-in AND, OR, NOT, and MOD_m gates, where m is an arbitrary natural number. Williams gives a non-uniform lower bound for **ACC** circuits, showing that $\mathbf{NTIME}[2^n]$ does not have non-uniform **ACC** circuits of polynomial size [**?**], and Chen and Papakonstantinou improve this bound by giving a strategy for depth reduction of **ACC** circuits which accounts for composite moduli [**?**], matching the size of the construction given in [**?**]. For the rest of this discussion, I'll assume standard definitions and notations for circuits and circuit families.

First I'll discuss the context of [**?**], as these methods go back quite a while. The study of circuits over the basis $\{\wedge, \vee, \neg\}$ goes back to just about as long as we've been studying circuits at all, the earliest reference I can think of being Shannon in 1938 [**?**], wherein the first hints of the research program under discussion can be seen. He defines and discusses a class of circuits he calls series-parallel, the class of circuits which can be formed via the operations of laying circuits out in series or in parallel, and he discusses whether or not the circuit is closed, as that would determine whether or not there is current flowing through it. Finding the connections we all take for granted between digital circuits and logic, with the laws which accompany that understanding, he writes the following:

> To find the circuit requiring the least number of contacts, it is therefore necessary to manipulate the expression into the form in which the least number of letters appear. The theorems given above are always sufficient to do this. A little practice in the manipulation of these symbols is all that is required. Fortunately most of the theorems are exactly the same as those of numerical algebra – the associative, commutative, and distributive laws of algebra hold here. [**?**]

How close Shannon was to a much more difficult to understand model is not indicated by the statement above, as without modification, the class of circuits he discusses, AC, is actually quite docile to analysis. In [**?**] this is foreshadowed when he discusses a function which is quite hard for series-parallel circuits, the parity function. Specifically, Shannon shows:

Theorem 1 (Shannon, 1938) The two functions of n variables which require the most elements in a series-parallel realization are parity and the negation of parity, each of which require $3 \times 2^{n-1} - 2$ gates.

Proof of Theorem 1 [**?**] The proof of this is by induction, mirroring the previous proofs in the paper which were by "perfect" induction, which is what we'd call "brute force". It's a mechanical task to note that this is true for $n = 2$, as there are only 10 functions involving two variables. Assuming that the theorem is true for $n - 1$, we can decompose our function $f(X_1, X_2, ..., X_n)$ via the nth variable:

$$f(X_1, X_2, ..., X_n) = X_n \wedge f(X_1, ..., X_{n-1}, 1) \vee X'_n \wedge f(X_1, ..., X_{n-1}, 0)$$

This is so because if $X_n = 1$, then the above equation is true if and only if $f(X_1, ..., X_{n-1}, 1) = 1$, whereas if $X_n = 1$, we have that $f(X_1, ..., X_{n-1}, X_n) = f(X_1, ..., X_{n-1}, 1)$. The case where $X_n = 0$ follows from the same reasoning. Now that we have our two functions in terms of two functions of $n-1$ variables, and we know that we can do this decomposition to any two functions, we can surmise that if $f(X_1, ..., X_{n-1}, 1)$ and $f(X_1, ..., X_{n-1}, 0)$ each require the most elements, then we can infer that $f(X_1, ..., X_n)$ does as well, as long as there's no other more concise way to represent f. We can take advantage of this, as our inductive hypothesis supplies us with the two hardest functions on $n - 1$ inputs and if we let f be the parity function or its negation we can easily use this decomposition as well as the symmetry of parity to show that there is no way to expand this function in a way which will reduce the number of elements. This relies upon the idea that all simplifications can be performed using the laws shown earlier in the paper. The second part of the proof requires a similar induction, but the structure of it is built out for us already, and we can use the decomposition in order to retrieve a recurrence for the size of the circuit of the form $s(n) = 2 \times s(n - 1) + 2$, as the irreducible formula we've retrieved for parity consists of two and gates as well as two computations of parity on $n - 1$ inputs. If the brute force method is followed on $n = 2$ one finds that parity can be written as $X_1 \wedge X'_2 \vee X'_1 \wedge X_2$, which has four elements, and solving for this recurrence using the initial condition $s(2) = 4$, one gets the desires result of $3 \times 2^{n-1} - 2$. □

The above theorem is an indication that adding parity to this model of series-parallel circuits could substantially increase their strength, seemingly up to an exponential amount. The idea of adding circuits like this was further explored in the same paper, when Shannon discussed what he called symmetric functions, characterizing them by the fact that if you permute the inputs, the output stays the same. He presented a few theorems for thinking about these functions, in particular the idea that you can think of any symmetric boolean function as merely being a set of natural numbers, each less than the number of free variables, and the function then is a test of membership in this set. Shannon presents a way of synthesizing circuits to compute such functions and shows a very efficient circuit to compute parity using this method [**?**]. This work is not normally thought of as the precursor to the new work, but the similarities are too glaring not to discuss it briefly.

Smolensky's Algebraic Methods

From the preceding, it's clear that augmenting traditional circuits with symmetric functions can give them a lot more power. That being said, the work of Shannon shows that parity is quite hard, but doesn't really give us the ability to compare it to how hard it is in comparison to things like conjunction and disjunction, as these things are simply built into our model. To this effect, we must look forward a few decades [**?**] [**?**]. By finding a way to do so, it can be shown that it is provably quite hard to compute MOD_r in ACC_p, for $r \neq p^n$, p prime. The proofs used come almost directly, with some slight modifications, from [**?**].

The key observation is that, given a field F and a boolean function, we can construct a function $\mathbb{B}^n \to F$ such that this function evaluates to 0 if and only if the boolean function is false, and 1 if and only if the boolean function is true. Understanding this, one can then show that this algebra is generated as an F-algebra via the relations $X_i^2 = X_i$. It is clear that each of our functions satisfies this condition, but to prove this, we use a similar strategy as used in [**?**] to decompose functions and note that if we fix an input $x \in \mathbb{B}^n$, we can construct the following function:

$$f(X_i) = \prod_{X_i(x)=1} X_i \times \prod_{X_j(x)=0} (1 - X_j)$$

Clearly, f takes 1 on x and 0 elsewhere, and with this tool we can construct all of the other functions we want as a linear combination of ones like this. If we request that none of the X_i appear with degree greater than one, we get that this representation is unique, as the number of monomials of this form is equal to the dimension of the F-algebra we've constructed, 2^n. This is true because to form such monomials we choose a subset of our variables to include, thus we must include all 2^n of these [**?**].

Definition: F-Easy We'll say that a boolean function is F-easy if it can be represented in its own variables as a polynomial of constant degree.

For any field F, it is clear that NOT is F-easy, as $NOT(g) = 1 - g$ in our algebra. When $F = Z_p$, it's clear that MOD_p is F-easy via Fermat's litle theorem. One struggle we immediately come up against however is that AND is clearly not F-easy, as we can see that it has linear size. Clearly this leaves us with the same problem for OR via duality and the fact that NOT is F-easy, so clearly this stern notion of F-easiness is not exactly what we mean when we discuss efficient computation [**?**].

Definition: nearly F-easy A function f^n is nearly F-easy if for any choice of boolean functions $g_1, g_2, ..., g_m$, and any l, there exists a quotient of our original F-algebra A^n with dimension at least $2^n - 2^{n-l}$ such that $f^n(g_1, g_2, ..., g_n)$ can be written in A^n as a polynomial in g's of degree at most $\lambda \times l$, where λ is a constant.

Lemma 1 (Smolensky, 1987) OR is a nearly F-easy operation for any field F of characteristic $p \neq 0$.

Proof of Lemma 1 [?] In order to show Theorem 2, one needs to find a polynomial of low degree and show that it is equivalent to our old polynomial on some quotient of the original algebra. In order to do this, we consider polynomials over Z_p, as this is sufficient given that our field is of characteristic p. We consider a set $S = \{OR_{j=1}^{l}(\sum_{i=1}^{m} C_{ij} g_i)^p - 1\}$. Clearly if for some assignment d, $f(d) = 0$, then we have that $g_i(d) = 0$, so $\forall s \in S, s(d) = 0$. In the positive case, we have that if $f(d) = 1$, for some i_0, $g_{i_0}(d) = 1$, by a simple linear algebra argument, for any choice of the C_{ij} where $i \neq i_0$, we have that there's only one setting of $C_{i_0 j}$s such that this expression is zero on d, and thus we can show that for any random element, the probability that $f(d) \neq s(d)$ is $\leq p^{-l}$. From here we can use a counting argument to show there must exist an element $k \in S$ such that $k(d) \neq f(d)$ on at most $2^n - l$ assignments, as if there weren't then we can show that the probability of our random circuit being correct is lower than it is.

The above lemma shows that, though OR is not F-easy in the same sense as NOT and MOD_p, clearly it is easier than something for which this is not the case, as we can actually use our polynomial approximation of OR to get good results, whereas if we could only have bad approximators for some function, clearly that is actually a much harder problem. In the next lemma, we show that if we have a whole circuit of constant depth consisting of F-easy and nearly F-easy operations, and it isn't too large, there's a quotient algebra of our original of size nearly as large as the original wherein we have that all outputs of C^n have degree $o(\sqrt{n})$.

Lemma 2 (Smolensky, 1987) If C^n is a depth K circuit with an arbitrary number of F-easy gates and 2^r nearly F-easy gates, where r is $o(n^{1/2k})$, then there exists A^n, a quotient of our algebra, with dimension just a tad below 2^n, $2^n - o(2^n)$, such that all outputs of C^n have degree $o(\sqrt{n})$ in A^n.

Proof of Lemma 2 [?] Essentially what we do is we inject each of these operations into the sum of all of the ideals which we know exist given that each operation is F-easy or nearly F-easy, and then we see that each gate computes a function which can be expressed as a polynomial of degree $o(n^{1/2k})$. As the depth of C^n is k, each output will then have a degree of $o(\sqrt{n})$ in the quotient algebra generated by the sum of these ideals.

After this, Smolensky introduces the key notion of U_F^n completeness, for our F-algebra U_F^n.

Definition: U_F^n-complete A set of elements v_i^n is U_F^n-complete if for any quotient A of U_F^n and any polynomial $u \in U_F^n$, $deg_A(u) \leq n/2 + max(deg_A(v_i^n)$.

Lemma 3 (Smolensky, 1987) Let $h \in F$, $h \neq 0, 1$. Take $Y_i = (h-1)x_i + 1$ then $\prod_{i=1}^n Y_i$ is U_F^n-complete.

Proof of Lemma 3 [?] Clearly $Y_i(0) = 1, Y_i(1) = h$, so we can see that $X_i = (h-1)^{-1}(Y_i - 1), Y_i^{-1} = (h^{-1} - 1)X_i + 1$. We can write u as a polynomial in Y_i's using these facts, and means that its enough for us to show that for any monomial in the Y_is, $\prod_{i \in W} Y_i$, where $W \subset \{1, 2, ..., n\}$, $deg_A(\prod_{i \in W} Y_i) \leq \frac{n}{2} + deg_A(\prod_{i=1}^n Y_i)$ in all F-algebra A. If $|W| \leq \frac{n}{2}$ this is clear, and when $|W| > \frac{n}{2}$ we note that $\prod_{i \in W} Y_i$ is equal to $\prod_{i=1}^n Y_i \times \prod_{i \in complement(W)} Y_i^{-1}$, and then, noting that $|complement(W)| \leq \frac{n}{2}$, we have this case as the first.

To comment on the above proof, the key fact that we used was that the nature of the Y_i allowed us to rewrite arbitrary polynomials in X_i in terms of Y_i, and then use simple facts like linearity to complete the proof. This ability to express arbitrary polynomials is considerably akin to the notion of completeness in the classical setting, and presumably is the reason for writing this definition as such. We will soon see the power that thinking about completeness in this new setting gives us.

Definition: Roots of Unity An element $a \in F$ is a q-th root of unity if $a \neq 1$ and $a^q = 1$.

Corrolary of Lemma 3 [?] The above is interesting because it can allow us to see that $MOD_{s,q}$ are U_F^n-complete. In order to see this we let h be a q-th root of unity, $q \neq p$, and we note that we can write $\prod_{i=1}^n Y_i = \sum_{s=0}^{q-1} h^s \times MOD_{s,q}(X_1, X_2, ..., X_n)$, as when $k \equiv s(mod(q))$, we can see that for d where d

contains k ones, $\prod Y_i(d) = h^k = h^s$. Thus we know that in any algebra $\prod Y_i$ is the maximal necessary degree of $MOD_{s,q}$, and we have our result from Lemma 3.

The next result Smolensky shows is quite beautiful, and it brings everything together in a way which makes the more classical results that will follow very easy to show. The essence of the following statement is that if we have a U_F^n-complete set, and we can represent it efficiently in some algebra, then that algebra is necessarily quite small. Combining this with Lemma 2 allows us to see that representing our circuit in this algebraic way and showing that our basis is F-easy can allow us to give lower bounds in a very elegant framework, optionally divorced from the classical setting. The statement is as follows:

Lemma 4 (Smolensky, 1987) If a set v_i^n is U_F^n-complete, and in some algebra A^n, a quotient of U_F^n, its degree is $o(\sqrt{n})$, then $dim(A^n) \leq 2^{n-1} + o(2^n)$.

Proof of Lemma 4 [?] An arbitrary element $a \in A^n$ can be written as a polynomial of degree at most $\frac{n}{2}+o(\sqrt{n})$, implying that A^n is equal to the span of all monomials of this degree, as this is a way to produce all such polynomials and they are closed under the field operations anyways. The number of monomials of this form is $\sum_{i=0}^{\frac{n}{2}+o(\sqrt{n})} C(n,i)$, which is equal to $2^{\frac{n}{2}+o(\sqrt{n})}$ which can be approximated as $2^{n-1} + o(2^n)$.

With the above four lemmas in hand, Smolesnky then moved to attack the main theorems of his paper:

Theorem 1 (Smolensky, 1987) Given a depth k circuit C^n which uses $exp(o(n^{\frac{1}{2k}}))$ nearly F-easy gates and an arbitrary number of F-easy gates. Then the output g of C^n will differ from any U_F^n-complete element f on $2^{n-1} - o(2^n)$ assignments.

Proof of Theorem 1 [?] By lemma 2 we have an algebra A^n such that g has degree $o(\sqrt{n})$ in A^n and the dimension of A^n is $2^n - o(2^n)$. Ignoring where g differs from f, we get a smaller algebra B^n. In A^n, g_i's coincide with f_i's and thus have degree $o(\sqrt{n})$, and thus we know that by lemma 4 the dimension of B^n must be $2^{n-1} + o(2^n)$, and thus we ignored $2^n - 2^{n-1} + o(2^n) = 2^{n-1} - o(2^n)$ assignments.

The above proof is quite terse, and it gives a very great way to give a great bound on how bad it goes when you try to compute parity in AC^0.

Corollary of Theorem 1 (Smolensky, 1987) The output of any depth k size $exp(o(n^{\frac{1}{2k}}))$ circuit with AND, OR, NOT as a basis differs from the MOD_2 function on $2^{n-1} - o(2^n)$ assignments.

Proof of Corrolary to Theorem 1 Noting the corrolary to lemma 3 which gave us the U_F^n-completeness of MOD_a, $NOT(MOD_a)$, and the F-easiness of AND, OR, we apply theorem 1 and get the desired result.

Theorem 2 (Smolensky, 1987) For p prime, $\nexists a, r \neq p^a$, computing MOD_r by depth k circuits in ACC_p requires $exp(O(n^{\frac{1}{2k}}))$ AND and OR gates.

Proof of Theorem 2 [?] Let $q|r$, q prime, $q \neq p$. Let $F = F_{p^{q-1}}$, then clearly MOD_p is F-easy, but by the corrolary to lemma 3 we can see that $MOD_{i,q}$ is U_F^n-complete and thus requires a large circuit. As this set is AC^0 reducible to MOD_r, as we chose $q|r$, we know that MOD_r requires a large circuit as well.

These methods have clearly simplified this proof a whole lot, and it's not clear whether or not the results would even be achievable from a direct, classical complexity attack on the problem. The thing I'd like to emphasize the most is that these methods of analysis are far different than those that people had used before, and there's a lot of evidence that says that this will necessarily be the case for many of the complexity classes we want to attack still today, so this was a great step forward. The geometric perspective this gives us is invaluable as well, and Smolensky comments on this in the next section, discussing the possibility that lemma 1 holds for a field of characteristic 0, so that analytic methods can be used, as "small" fields of characteristic zero can all be embedded in the complex field [?].

Nondeterministic Time Hierarchy

In [?], a contradiction of the nondeterministic time hierarchy is used to separate ACC and NEXP, so I also wanted to cover that here. The statement is as follows [?]:

Nondeterministic Time Hierarchy Theorem Let t_1, t_2 be time-constructible functions, then if $t_1(n+1) = o(t_2(n))$, then $NTIME(t_1(n)) \subset NTIME(t_2(n)))$, where $\subset$ denotes strict subset.

Proof [?] Let M_i be an enumeration of nondeterministic Turing machines. We define a nondeterministic machine M such that on input $w = 1^i01^m0y$, if $|y| < t_1(i+m+2)$, then accept if M_i accepts both 1^i01^m0y0 and 1^i01^m0y1 in $t_2(|w|)$ steps. If $|y| = t_1(i+m+2)$, then accept if M_i rejects input 1^i01^m0 on the computation path described by y. This machine uses time $O(t_2(n)$, but if our theorem is contradicted there would exist an equivalent machine M^* using time $O(t_1(n))$. Since $t(n+1) = o(t_2(n))$, if we choose sufficiently large m, then we can show that $1^i01^m0 \in L(M)$ if and only if $1^i01^m0y \in L(M)$ for $|y| = 1$ if and only if $1^i01^m0y \in L(M)$ for $|y| = 2$ and so on to $i^i01^m01y \in L(M)$ for $|y| = t_1(i+m+2)$ which, because of the second part of the machines, definition, causes a contradiction with the first statement, as we've involved every single computation path possible in the condition before.

I really enjoy this proof as it's exactly the sort of nice, self-referential process which you tend to think of as actually being the ontological cause of these sorts of inabilities of logic and computation.