ACC lower bounds (lecture notes)

Course: Circuit Complexity, Autumn 2024, University of Chicago
Instructor: William Hoza (williamhoza@uchicago.edu)

Recall that "$\mathsf{AC}^0[m]$ circuits" can use AND gates, OR gates, and MOD_m gates, all with unbounded fan-in. There are constants and literals at the bottom. When m is not a power of a prime, the class $\mathsf{AC}^0[m]$ is poorly understood. As mentioned previously in this course, it is an open problem to show $\mathsf{NP} \not\subseteq \mathsf{AC}^0[6]$.

On the bright side, there is a line of work showing that there are "somewhat explicit" functions that cannot be computed by small $\mathsf{AC}^0[6]$ circuits and similar models. In particular, Murray and Williams showed $\mathsf{NQP} \not\subseteq \mathsf{ACC}$ [MW18], where NQP is nondeterministic quasipolynomial time and $\mathsf{ACC} = \bigcup_m \mathsf{AC}^0[m]$.

The full proof that $\mathsf{NQP} \not\subseteq \mathsf{ACC}$ is beyond the scope of this course. In these lecture notes, we will sketch the proof of the weaker theorem $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}$, where E^{NP} denotes the class of functions that can be computed in $2^{O(n)}$ time using an NP oracle.

1 Depth reduction for ACC circuits

In this section, we will show that ACC circuits can be simulated by low-degree polynomials in a certain sense, despite the fact that we do not know how to construct low-degree probabilistic polynomials computing the MOD_6 function. Recall that $\mathbb{Z}[x_1, \ldots, x_n]$ is the set of n-variate polynomials with integer coefficients.

Definition 1 (L_1 norm of a polynomial)**.** If $h \in \mathbb{Z}[x_1, \ldots, x_n]$, then we define $L_1(h)$ to be the sum of the absolute values of the coefficients.

Definition 2 (SYM^+)**.** We define $\mathsf{SYM}^+[k]$ to be the class of functions $C \colon \{0,1\}^n \to \{0,1\}$ of the form $C(x) = g(h(x))$, where $h \in \mathbb{Z}[x_1, \ldots, x_n]$ satisfies $\deg(h) \leq k$ and $L_1(h) \leq 2^k$. Note that h is multilinear without loss of generality. The function $g \colon \mathbb{Z} \to \{0,1\}$ can be arbitrary, but we emphasize that it is a function of just one integer variable.

You can double check that each function in $\mathsf{SYM}^+[k]$ can be computed by a "SYM of AND of literals," where the AND gates have fan-in at most k and the SYM gate has fan-in at most $2^{O(k)}$. Consequently, each function in $\mathsf{SYM}^+[k]$ can be computed by a TC^0_3 circuit of size $2^{O(k)}$. The following theorem is a key step in the proof that $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}$, as well as being interesting in its own right.

Theorem 1 (Simulating $\mathsf{AC}^0[m]$ circuits using SYM^+ circuits)**.** *Let $m, d \in \mathbb{N}$ be constants. If $C \colon \{0,1\}^n \to \{0,1\}$ is an $\mathsf{AC}^0_d[m]$ circuit of size $S \geq n$, then $C \in \mathsf{SYM}^+[\operatorname{polylog} S]$.*

When d and m are growing parameters, the best bound known is $C \in \mathsf{SYM}^+[(\log S)^{O(d \cdot s)}]$, where s is the number of distinct prime factors of m [CP19]. In these lecture notes, we assume d and m are constant for simplicity.

1.1 Simulating MOD_m gates using MOD_p gates

Lemma 1. *Let $p, e \in \mathbb{N}$ be constants, where p is prime and $e \geq 1$. Then $\mathsf{MOD}_{p^e} \in \mathsf{AC}^0[p]$.*

Proof. We prove it by induction on e. The base case $e = 1$ is trivial. For the inductive step, let $e \geq 2$, let $x \in \{0,1\}^n$, and let N be the Hamming weight of x. We claim that[1]

$$\mathsf{MOD}_{p^e}(x) = \mathsf{MOD}_p(x) \vee \mathsf{MOD}_{p^{e-1}}\left(\bigwedge_{i \in S_1} x_i, \ldots, \bigwedge_{i \in S_{\binom{n}{p}}} x_i \right), \tag{1}$$

[1] Recall that we defined $\mathsf{MOD}_m(x) = 1 \iff x_1 + \cdots + x_n \not\equiv 0 \pmod m$, which is opposite to the way many sources define it.

where $S_1, S_2, \ldots, S_{\binom{n}{p}}$ is an enumeration of all size-p subsets of $[n]$. If N is not a multiple of p, this is trivial: $\mathsf{MOD}_{p^e}(x) = \mathsf{MOD}_p(x) = 1$. Now assume N is a multiple of p. In this case, observe that

$$\binom{N}{p} = \frac{N \cdot (N-1) \cdots (N-p+1)}{p \cdot (p-1) \cdots 1}.$$

In both the numerator and the denominator, only the first term is a multiple of p. Therefore, the exponent of p in the prime factorization of $\binom{N}{p}$ is one less than the exponent of p in the prime factorization of N. That is, $p^e \mid N$ if and only if $p^{e-1} \mid \binom{N}{p}$. Eq. (1) follows. By induction, Eq. (1) shows $\mathsf{MOD}_{p^e} \in \mathsf{AC}^0[p]$; note that $\mathrm{poly}(\binom{n}{p}) = \mathrm{poly}(n)$ since p is a constant. □

More generally, let m be an arbitrary positive integer, with prime factorization $m = p_1^{e_1} \cdot p_2^{e_2} \cdots p_s^{e_s}$. Then

$$\mathsf{MOD}_m(x) = \mathsf{MOD}_{p_1^{e_1}}(x) \vee \cdots \vee \mathsf{MOD}_{p_s^{e_s}}(x).$$

Thus, we can simulate an $\mathsf{AC}^0[m]$ circuit using AND gates, OR gates, MOD_{p_1} gates, MOD_{p_2} gates, …, and MOD_{p_s} gates. The depth blows up by a constant factor and the size blows up polynomially, assuming m is a constant.

1.2 Eliminating one layer of MOD_p gates

Lemma 2 (Modulus-amplifying polynomials)**.** *For every $k \in \mathbb{N}$, there exists a polynomial $M_k \in \mathbb{Z}[x]$ such that $\deg(M_k) = O(k)$, $L_1(M_k) = 2^{O(k)}$, and for every $x \in \mathbb{Z}$ and every $p \in \mathbb{N}$,*

$$\begin{aligned} x \equiv 0 \pmod p &\implies M_k(x) \equiv 0 \pmod{p^k} \\ x \equiv 1 \pmod p &\implies M_k(x) \equiv 1 \pmod{p^k}. \end{aligned} \tag{2}$$

Proof. Define

$$M_k(x) = \sum_{i=0}^{k-1} \binom{2k-1}{i} \cdot x^{2k-1-i} \cdot (1-x)^i.$$

The degree and L_1 bounds are straightforward. Observe that $M_k(x)$ is a multiple of x^k, which proves Eq. (2). Now suppose $x \equiv 1 \pmod p$. Then $1 - x$ is a multiple of p, so $(1-x)^i \equiv 0 \pmod{p^k}$ whenever $i \geq k$. Consequently,

$$\begin{aligned} M_k(x) &\equiv \sum_{i=0}^{2k-1} \binom{2k-1}{i} \cdot x^{2k-1-i} \cdot (1-x)^i \pmod{p^k} \\ &= (x + 1 - x)^{2k-1} && \text{by the binomial theorem} \\ &= 1. \end{aligned}$$

□

Lemma 3 (SYM^+ can simulate $\mathsf{SYM}^+ \circ \mathsf{MOD}_p$)**.** *Let $n, k \in \mathbb{N}$, let p be prime, and let $C \colon \{0,1\}^n \to \{0,1\}$ be a formula consisting of variables feeding into MOD_p gates feeding into a $\mathsf{SYM}^+[k]$ gate. Then $C \in \mathsf{SYM}^+[O(k^3 \cdot p \cdot \log n)]$.*

Proof. By introducing dummy variables if necessary, we can write

$$C(x) = g\left(\left(\sum_{i=1}^{L} c_i \prod_{j=1}^{k} \mathsf{MOD}_p(x_{ij1}, \ldots, x_{ij\ell})\right) \bmod p^{k+2}\right),$$

where $g\colon \mathbb{Z} \to \{0,1\}$, each $c_i \in \mathbb{Z}$, we have $\sum_{i=1}^{L} |c_i| \le 2^k$, and $\ell \le n$. (Reducing mod p^{k+2} doesn't destroy any information, because the sum lies between -2^k and 2^k.) Therefore,

$$
\begin{aligned}
C(x) &= g\left(\left(\sum_{i=1}^{L} c_i \prod_{j=1}^{k} \mathbb{1}\left[\sum_{t=1}^{\ell} x_{ijt} \not\equiv 0 \mod p\right]\right) \bmod p^{k+2}\right) && \text{by definition of } \mathsf{MOD}_p \\
&= g\left(\left(\sum_{i=1}^{L} c_i \cdot \mathbb{1}\left[\prod_{j=1}^{k}\sum_{t=1}^{\ell} x_{ijt} \not\equiv 0 \mod p\right]\right) \bmod p^{k+2}\right) && \text{because a product of nonzero elements is nonzero in any field, including } \mathbb{F}_p \\
&= g\left(\left(\sum_{i=1}^{L} c_i \cdot M_{k+2}\left(\left(\prod_{j=1}^{k}\sum_{t=1}^{\ell} x_{ijt}\right)^{p-1}\right)\right) \bmod p^{k+2}\right) && \text{by Fermat's little theorem and modulus amplification.}
\end{aligned}
$$

The expression above has the format of SYM^+: first we apply a multivariate polynomial, and then we apply a univarate function ("reduce mod p^{k+2}, then apply g"). The degree of the polynomial is at most $k\cdot(p-1)\cdot\deg(M_{k+2}) = O(p\cdot k^2)$. The L_1 norm of this polynomial is at most $2^k \cdot L_1(M_{k+2}) \cdot (\ell^{k\cdot(p-1)})^{\deg(M_{k+2})} = n^{O(p\cdot k^3)}$. □

1.3 Simulating the entire circuit

Proof sketch of Theorem 1. There are several steps, but none is too difficult, given the tools that we have developed.

1. Replace each MOD_m gate with AND gates, OR gates, MOD_{p_1} gates, ..., and MOD_{p_s} gates, as described in Section 1.1.

2. Replace each AND/OR gate with a probabilistic polynomial over the field $\mathbb{F}_2$ with error $0.1/S$ and degree $\ell = O(\log S)$. Note that a degree-ℓ polynomial over $\mathbb{F}_2$ is a $\mathsf{MOD}_2 \circ \mathsf{AND}_\ell$ circuit, where the MOD_2 gate has fan-in at most $S^{O(\log S)}$. Let $\mathcal{D}$ be the resulting distribution over circuits.

3. Independently sample $t = O(n)$ circuits $C_1, \ldots, C_t \sim \mathcal{D}$ and set $C(x) = \mathsf{MAJ}_t(C_1(x), \ldots, C_t(x))$. By Hoeffding's inequality and the union bound over all $x \in \{0,1\}^n$, there is some fixing of $C_1, \ldots, C_t$ such that C computes f. Note that each C_i consists of MOD_2 gates, MOD_{p_1} gates, MOD_{p_2} gates, ..., MOD_{p_s} gates, and AND_ℓ gates (with literals and constants at the bottom).

4. By introducing dummy gates if necessary, we can ensure that *all gates at the same level are of the same type*. In other words, we can compute f using a circuit of the following form:

$$\mathsf{MAJ}_t \circ (\mathsf{MOD}_2 \circ \mathsf{MOD}_{p_1} \circ \cdots \circ \mathsf{MOD}_{p_s} \circ \mathsf{AND}_\ell)^{O(1)}.$$

5. Note that $\mathsf{MAJ}_t \in \mathsf{SYM}^+[\log t]$. We eliminate the layers underneath the SYM^+ gate one by one to get a $\mathsf{SYM}^+[k]$ circuit. To handle MOD_p layers, we use Lemma 3. To handle AND_ℓ layers, we use the trivial fact $\mathsf{SYM}^+[k] \circ \mathsf{AND}_\ell \subseteq \mathsf{SYM}^+[k \cdot \ell]$. Since the number of layers is $O(1)$, we get $f \in \mathsf{SYM}^+[\mathrm{polylog}(S)]$.

□

2 A nontrivial satisfiability algorithm for ACC circuits

The next step in the proof that $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}$ is to design a nontrivial algorithm that *analyzes a given* ACC *circuit* and determines whether it is satisfiable. The algorithm is based on the following lemma, which shows how to evaluate a polynomial h on all points in $\{0,1\}^n$ using amortized time $\mathrm{poly}(n, \log L_1(h))$ per evaluation.

Lemma 4 (Fast multipoint evaluation of multilinear polynomials)**.** *Let $h\colon \{0,1\}^n \to \mathbb{Z}$ be a multilinear polynomial with integer coefficients. Given h, represented as a list of 2^n coefficients, it is possible to compute $h(x)$ for all $x \in \{0,1\}^n$ in time $2^n \cdot \operatorname{poly}(n, \log L_1(h))$.*[2]

Proof. Let $s = L_1(f)$. If $n = 0$, then the problem is trivial. Otherwise, there are polynomials h_0, h_1 such that

$$h(x_1, \ldots, x_n) = h_0(x_2, \ldots, x_n) + x_1 \cdot h_1(x_2, \ldots, x_n). \tag{3}$$

Considered as lists of coefficients, h_0 and h_1 are simply the first half and the second half of h, respectively. In particular, $L_1(h_0) \le s$ and $L_1(h_1) \le s$. We recursively compute $h_0(x)$ and $h_1(x)$ for all $x \in \{0,1\}^{n-1}$, and then we use Eq. (3) to compute $h(x)$ for all $x \in \{0,1\}^n$. The time complexity of this algorithm, $T(n,s)$, satisfies

$$T(n,s) \le 2T(n-1,s) + 2^n \cdot \operatorname{poly}(n, \log s),$$

because $|h_0(x)| \le s$ and $|h_1(x)| \le s$ for all $x \in \{0,1\}^{n-1}$. This implies $T(n,s) \le 2^n \cdot \operatorname{poly}(n, \log s)$. □

Theorem 2 (Nontrivial satisfiability algorithm for ACC)**.** *There exists an algorithm that determines whether a given $\mathsf{AC}^0_d[m]$ circuit $C\colon \{0,1\}^n \to \{0,1\}$ is satisfiable, and if m and d are held constant, then the algorithm's time complexity is $2^{n-n^{\Omega(1)}} + 2^{\operatorname{polylog} S}$, where S is the size of C.*

Proof sketch. First, let us design a satisfiability algorithm that runs in time $2^n \cdot \operatorname{poly}(n) + 2^{\operatorname{polylog}(S)}$. If $S > 2^n$ or $S < n$, then the trivial algorithm (try all possible inputs) is fast enough. If $n \le S \le 2^n$, then we use the following algorithm:

1. By Theorem 1, it is possible to write C in the form $C(x) = g(h(x))$ where $h\colon \{0,1\}^n \to \mathbb{Z}$ is a multilinear polynomial satisfying $L_1(h) \le 2^{\operatorname{polylog} S}$. We only showed that such a representation *exists*, but it turns out that it can be *constructed* in quasipolynomial ($2^{\operatorname{polylog} S}$) time, if we represent h as a list of monomials with nonzero coefficients and we represent g as a list of $2^{\operatorname{polylog}(S)}$ output values. We omit the proof that g and h can be efficiently constructed.
2. Rewrite h as a list of 2^n coefficients, many of which may be zero. This can be done in time $2^{\operatorname{polylog} S} + 2^n \cdot \operatorname{poly}(n)$.
3. Compute $h(x)$ for all $x \in \{0,1\}^n$. By Lemma 4, this can be done in time $2^n \cdot \operatorname{poly}(n)$.
4. Check whether there is some $x \in \{0,1\}^n$ such that $g(h(x)) = 1$. This can be done in time $2^n \cdot \operatorname{poly}(n)$.

Now we explain how to improve the time complexity to $2^{n-n^{\Omega(1)}} + 2^{\operatorname{polylog} S}$. Given C, define $C'\colon \{0,1\}^{n-n^\varepsilon} \to \{0,1\}$ by the rule

$$C'(x) = \bigvee_{y \in \{0,1\}^{n^\varepsilon}} C(xy).$$

Then C is satisfiable if and only if C' is satisfiable. The circuit C' is an $\mathsf{AC}^0_{d+1}[m]$ circuit of size $2^{n^\varepsilon} \cdot S$, so using the algorithm described above, we can decide whether it is satisfiable in time $2^{n-n^\varepsilon} \cdot \operatorname{poly}(n) + 2^{\operatorname{poly}(n^\varepsilon, \log S)}$. The theorem follows by picking a small enough ε. □

3 Nontrivial satisfiability algorithms imply circuit lower bounds

We are working toward proving $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}$. The last part of the proof consists of showing that nontrivial satisfiability algorithms, such as the one we constructed in the last section, imply circuit lower bounds. This last part of the proof is not specific to ACC; it could also be applied to a hypothetical nontrivial satisfiability algorithm for (say) TC^0 circuits.

Our "hard problem" will be a version of the classic 3-SAT problem. Basically, the problem is to construct a satisfying assignment for an *exponential-size* 3-SAT instance, given a circuit that constructs the 3-SAT instance. To be more precise, we make the following definitions.

[2] We assume a *random access* model of computation throughout these lecture notes.

Definition 3 (Succinct representation of a 3-CNF formula)**.** Let $\phi\colon \{0,1\}^{2^b} \to \{0,1\}$ be a 3-CNF formula with 2^a clauses. We say that a circuit $C\colon \{0,1\}^a \to \{0,1\}^{3\cdot(b+1)}$ over the full binary basis *succinctly represents* ϕ if $C(i)$ describes the i-th clause of ϕ by providing the names of the three variables in the clause, along with three additional bits to indicate the presence/absence of negations. Note that the formula ϕ does not necessarily use all 2^b available variables.

Definition 4 (The problem in E^{NP} that we will show is not in ACC)**.** We define $h\colon \{0,1\}^* \to \{0,1\}$ as follows. Let $C\colon \{0,1\}^a \to \{0,1\}^{3\cdot(b+1)}$ be a circuit that succinctly represents a 3-CNF formula $\phi\colon \{0,1\}^{2^b} \to \{0,1\}$. Let $\sigma \in \{0,1\}^{2^b}$ be the lexicographically-first satisfying assignment for ϕ, or let $\sigma = 0^{2^b}$ if ϕ is unsatisfiable. For each $i \in \{0,1\}^b$, we define

$$h(C,i) = \sigma_i.$$

Proposition 1. $h \in \mathsf{E}^{\mathsf{NP}}$.

Proof sketch. First, construct the full description of the 3-CNF formula ϕ that is succinctly represented by C. This takes time $2^{O(n)}$. Then, construct the lexicographically-first satisfying assignment σ for ϕ, bit by bit. We can extend σ by one bit using the NP oracle, so this also takes time $2^{O(n)}$. Finally, output the i-th bit of σ. □

It remains to show that $h \notin \mathsf{ACC}$. The proof is a *reduction* from the following uniform time complexity lower bound.

Theorem 3 (Nondeterministic time complexity lower bound for succinct 3-SAT)**.** *There does not exist a nondeterministic algorithm that satisfies both of the following two conditions.*

1. *Given a circuit $C\colon \{0,1\}^a \to \{0,1\}^{3\cdot(b+1)}$ that succinctly represents a 3-CNF formula ϕ, the algorithm determines whether ϕ is satisfiable.*
2. *If C has depth $O(1)$ and size $\mathrm{poly}(a)$ (i.e., $C \in \mathsf{NC}^0$), then the algorithm's runtime is $2^{a-\omega(\log a)}$.*

The proof of Theorem 3 is beyond the scope of this course, but we will briefly summarize the proof idea. The first step is something called the Nondeterministic Time Hierarchy Theorem, which tells us that there exists $f\colon \{0,1\}^* \to \{0,1\}$ that can be computed nondeterministically in time $O(2^n)$ but not $o(2^n)$. The proof of the Nondeterministic Time Hierarchy Theorem is a clever and nontrivial diagonalization argument.

The second step of the proof of Theorem 3 is a sophisticated version of the Cook-Levin theorem. Recall that the classic Cook-Levin theorem says that 3-SAT is NP-complete. It immediately follows that there is an exponential-time reduction from f to 3-SAT. It turns out that something much stronger is true: There is a *polynomial*-time reduction that converts $x \in \{0,1\}^n$ into a circuit $C\colon \{0,1\}^a \to \{0,1\}^{3\cdot(b+1)}$ that *succinctly represents* a 3-CNF formula ϕ that is satisfiable if and only if $f(x) = 1$. Furthermore, this reduction can be carried out in such a way that $a = n + O(\log n)$ and $C \in \mathsf{NC}^0$ [JMV18]. Consequently, if an algorithm as described in Theorem 3 *did* exist, then we could run it to compute $f(x)$ nondeterministically in time $\mathrm{poly}(n) + 2^{a-\omega(\log a)} = 2^{n-\omega(\log n)}$, which would contradict the Nondeterministic Time Hierarchy Theorem.

Now let us take Theorem 3 for granted and use it to complete the proof that $\mathsf{E}^{\mathsf{NP}} \not\subseteq \mathsf{ACC}$.

Theorem 4. $h \notin \mathsf{ACC}$.

Proof sketch. Assume for the sake of contradiction that $h \in \mathsf{ACC}$. We will show how to contradict Theorem 3 using the so-called "guess-and-SAT" method. Let $C\colon \{0,1\}^a \to \{0,1\}^{3\cdot(b+1)}$ be a given NC^0 circuit that succinctly represents a 3-CNF formula $\phi\colon \{0,1\}^{2^b} \to \{0,1\}$. Since $h \in \mathsf{ACC}$, the value $h(C,i)$ can be computed by an $\mathsf{AC}^0_d[m]$ circuit of size $S = \mathrm{poly}(a)$ for some constants $d, m \in \mathbb{N}$. We determine whether ϕ is satisfiable as follows.

1. Nondeterministically guess an $\mathsf{AC}^0_d[m]$ circuit $A\colon \{0,1\}^b \to \{0,1\}$ of size S. We think of A as a succinct representation of an exponentially long assignment $\widehat{\sigma} \in \{0,1\}^{2^b}$.

2. For each $i \in [2^a]$, let $D(i)$ indicate whether $\widehat{\sigma}$ violates clause i of ϕ. Construct a circuit that computes D by starting with C, feeding its output into three copies of A, then applying three XOR gates to account for negations, and finally applying an AND_3 gate. Altogether, D is an $\mathsf{AC}^0[m]$ circuit of depth $d + O(1) = O(1)$ and size $\mathrm{poly}(a)$.

3. Use the ACC satisfiability algorithm (Theorem 2) to check whether D is satisfiable. If so, reject, otherwise, accept.

If ϕ is satisfiable, then we can nondeterministically guess a circuit A computing $A(i) = h(C, i)$, in which case $\widehat{\sigma}$ satisfies ϕ, hence D is unsatisfiable. On the other hand, if ϕ is unsatisfiable, then $\widehat{\sigma}$ violates at least one clause of ϕ, hence D is satisfiable. Constructing D takes $\mathrm{poly}(a)$ time, and testing whether it is satisfiable takes time $2^{a - a^{\Omega(1)}}$. □

References

[CP19] Shiteng Chen and Periklis A. Papakonstantinou. "Depth reduction for composites". In: *SIAM J. Comput.* 48.2 (2019), pp. 668–686. ISSN: 0097-5397. DOI: 10.1137/17M1129672.

[JMV18] Hamidreza Jahanjou, Eric Miles, and Emanuele Viola. "Local reduction". In: *Inform. and Comput.* 261.part 2 (2018), pp. 281–295. ISSN: 0890-5401. DOI: 10.1016/j.ic.2018.02.009.

[MW18] Cody Murray and Ryan Williams. "Circuit lower bounds for nondeterministic quasi-polytime: an easy witness lemma for NP and NQP". In: *Proceedings of the 50th Annual Symposium on Theory of Computing (STOC)*. 2018, 890–901. DOI: 10.1145/3188745.3188910.