Preliminaries and goals

The goal of complexity theory is to show that natural problems are hard. By “natural,” we usually mean problems that we encounter in complexity classes like $L$, $P$, $NP$, or $PSPACE$. We formalize “hard” as not in smaller complexity classes, such as $L$, $P$, $BPP$, or $P/poly$—polynomial-sized circuit families.

The Razborov-Smolensky circuit lower bound from 1987 says that the parity function is not in $AC^0 + \text{mod}_p$ for any odd prime where $AC^0$ is, roughly speaking, the class of languages decidable in constant parallel time (this is formalized below) and $\text{mod}_p$ being the function that outputs 0 if the sum of its inputs is a multiple of $p$ and 1 otherwise.

Ryan Williams recent result shows that nondeterministic exponential time is not a subset of $AC^0 + \text{mod}_N$ for any integer $N$.

This course is going to prove William’s result completely without omitting detail, including all of the results necessary for the proof.

$AC^0$

“Sipser’s program”\(^1\) is to look at special cases of Boolean circuits and prove lower bounds for stronger and stronger cases. So let’s start by looking at simple cases and then generalize.

- Probably the most simple, nontrivial special case of a Boolean circuit is one in conjunctive/disjunctive normal form (CNF/DNF). A CNF circuit has depth two. It is an AND of a number of OR clauses. For example $x_1 \land (x_2 \lor \lnot x_3) \land (x_4 \lor \lnot x_5 \lor \lnot x_6)$. A DNF circuit is similar except that the outermost gates are OR gates and the innermost are AND gates.

---

\(^1\)Sipser claims this is not his idea at all.
The number of inputs to each gate — the *fanin* — is unbounded and we allow some of the inputs to gates to be negated. Note that the parity function requires an exponential number of gates to be expressed in CNF or DNF.

- We can generalize CNF and DNF by adding another level of gates. We can assume that the levels alternate between AND and OR gates. (Otherwise, the levels could be collapsed.) To formalize this, let \( AC^0_d \) be the set of functions \( f \) such that the function \( f_n \) which computes \( f \) on \( n \) bit inputs has a depth \( d \), polynomial-sized circuit with unbounded fanin AND and OR gates.

- Finally, \( AC^0 = \bigcup_d AC^0_d \). Thus \( AC^0 \) is the class of languages decided by constant-depth, polynomial-sized, Boolean circuits consisting of alternating levels of AND and OR gates with unbounded fanin and possibly negated inputs.

That parity is not in \( AC^0 \) was proved by Ajtai, FSS (Furst, Saxe, and Sipser), Yao, and Hastad using a *switching lemma*. The Razborov-Smolensky result used an entirely different method which we’ll call the *polynomial approximation method*. Since William’s result does not use switching lemmata, we will not cover them. Instead, we will focus on the polynomial approximation method.

**Polynomial approximation method**

The basic idea underlying the polynomial approximation method is to replace a Boolean circuit with a polynomial over a finite field that computes the same (Boolean) function as the circuit when the polynomial is evaluated (in the finite field) on Boolean inputs. More formally, let \( p > 2 \) be prime and \( \mathbb{F}_p \) be the finite field of integers modulo \( p \). We can convert any Boolean expression of \( n \) literals to a multivariate polynomial in \( \mathbb{F}_p[x_1, x_2, \ldots, x_n] \) inductively using three simple rules:

- \( x \land y = xy \);
- \( \lnot x = 1 - x \); and using De Morgan’s law
- \( x \lor y = x + y - xy \).
For \( x \in \{0, 1\} \), \( x^k = x \). Therefore, we can replace \( x^k \) with \( x \) in the polynomial that results from the conversion. After this, the resulting polynomial is multilinear.

We now turn to the question of which Boolean functions have a small-degree polynomial over \( \mathbb{F}_p \), which computes the same function on almost all of its inputs. To answer this, we prove the following characterization lemma.

**Lemma 1.** If \( f : \{0, 1\}^n \to \{0, 1\} \) is computable by a size \( m \), depth \( d \) Boolean circuit (with \( \text{mod}_p \) gates), then there exists a degree \( D = O[(\log m)^d] \) polynomial \( g \in \mathbb{F}_p[x_1, x_2, \ldots, x_n] \) such that

\[
\Pr_{x_1, x_2, \ldots, x_n \in \{0, 1\}} \left[ f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n) \right] \geq 1 - \frac{1}{m}.
\]

**Proof.** Without loss of generality, we assume the circuit contains only OR and NOT gates. We proceed by induction. For a circuit \( C = C_1 \lor C_2 \lor \cdots \lor C_m' \), we have polynomials \( g_1, g_2, \ldots, g_m' \) with \( g_i \) corresponding to \( C_i \) and the “polynomial” \( g_1 \lor g_2 \lor \cdots \lor g_m' \).

Our goal is to produce a polynomial \( g \) such that on input \( x \in \{0, 1\}^m' \), if \( g_1(x) = g_2(x) = \cdots = g_m'(x) = 0 \), then \( g(x) = 0 \). If any \( g_i(x) = 1 \), then \( g(x) = 1 \), almost always. To do this, we are going to look at sums of random subsets of \( \{g_1, g_2, \ldots, g_m'\} \) and then bound the probability that on input \( x \) the sum is 0. Then, we’re going to take a disjunction of these sums using the rule \( x \lor y = x + y - xy \) and apply the union bound.

Let \( l = 2 \log m \) — the \( 2 \log m \) will come from wanting \( 2^l = m^2 \) shortly. Then, for each \( j \in \{1, 2, \ldots, l\} \), pick a subset \( S_j \subseteq \{1, 2, \ldots, m'\} \) uniformly at random. That is, for each element, flip a coin to decide if it should be in \( S_j \). Let \( h_j = \sum_{i \in S_j} g_i \).

Clearly, if all of the \( g_i(x) = 0 \), then each \( h_j(x) = 0 \). If, for some \( i, g_i(x) = 1 \), then \( \Pr_{S_j} [h_j(x) = 0] \leq \frac{1}{2} \). To see this, condition on \( S_j \setminus \{i\} \). If \( \sum_{k \in S_j \setminus \{i\}} g_k = T_j \), then

\[
h_j = \sum_{k \in S_j} g_k = \begin{cases} T_j + g_i & \text{with probability } \frac{1}{2} \\ T_j & \text{with probability } \frac{1}{2} \end{cases}
\]

since \( i \in S_j \) with probability \( \frac{1}{2} \). Thus the probability that \( h_j(x) = 0 \) for all \( j \) is at most \( \left( \frac{1}{2} \right)^l = \frac{1}{m^2} \).

We would like to set \( g = h_1 \lor h_2 \lor \cdots \lor h_l \) but the \( h_j \) need not have Boolean outputs when their inputs are Boolean. Instead, we raise each \( h_j \)
to the power $p - 1$. Thus, if $h_j(x) = 0$, then $h_j(x)^{p-1} = 0$ and if $h_j(x) \neq 0$, then $h_j(x)^{p-1} = 1$ since all arithmetic is over a field of characteristic $p$. Therefore, we set $g = h_1^{p-1} \lor h_2^{p-1} \lor \cdots \lor h_l^{p-1}$. This has the two desired properties that $g(x) = 0$ whenever $g_i(x) = 0$ for all $i$ and if any $g_i(x) = 1$, then $g(x) = 1$ with probability $1 - \frac{1}{m^2}$. This completes the construction of the polynomial $g$ from the circuit. It remains to bound the probability that $f(x_1, x_2, \ldots, x_n) = g(x_1, x_2, \ldots, x_n)$ over the inputs and the degree of the polynomial.

**Bounding the probability.** If $\gamma$ is a gate in the circuit which computes $f$, let $C_\gamma(x_1, x_2, \ldots, x_n)$ be the value of the output of that gate when the circuit has inputs $x_1, x_2, \ldots, x_n$ and $g_\gamma(x_1, x_2, \ldots, x_n)$ be the value of the polynomial associated with gate $\gamma$ produced by the above construction on $x_1, x_2, \ldots, x_n$.

For any intermediate gate $\gamma$ which takes as input the outputs of gates $\gamma_1, \gamma_2, \ldots, \gamma_{m'}$,

$$
\Pr_{S_1, S_2, \ldots, S_l} \left[ g_\gamma(x_1, x_2, \ldots, x_n) \neq C_\gamma(x_1, x_2, \ldots, x_n) \text{ but } g_{\gamma_i}(x_1, x_2, \ldots, x_n) = C_{\gamma_i}(x_1, x_2, \ldots, x_n) \text{ for all } i \right] \leq \frac{1}{m^2}.
$$

By applying the union bound over all $m$ circuits, we get that for all inputs $x_1, x_2, \ldots, x_n$, the probability that the polynomial $g = g_{\gamma_0}$ for the output gate $\gamma_0$ differs from the output $C_{\gamma_0}$ of the gate with probability at most $\frac{1}{m}$. The probability is over the choice of sets $S_j$. This proves the probability bound.

Note that this probability is for any fixed input over random choices of the sets $S$ used to approximate the gates. But then it follows by averaging that the same bound on the failure probability holds if we pick the input at random as well. This of course is the same if we pick the sets first, and then the input. Then the best choice of sets can have at most the same bound over choices of random input. This gives us a construction that works for almost all inputs (all but $1/m$ fraction of inputs $x$).

**Bounding the degree.** In the construction of $g$ from $g_1, g_2, \ldots, g_{m'}$, let $D'$ be the maximum degree of the $g_i$. We can compute a bound on the degree of $g$ from $D'$. The construction of each $h_j$ by adding some of the $g_i$ clearly does not change the degree so $\deg h_j \leq D'$. Therefore $\deg h_j^{p-1} \leq D'(p-1)$. Then $g$ is constructed as the “disjunction” of the $2\log m$ polynomials $h_j^{p-1}$ so $\deg g \leq D'(p-1)(2\log m)$. 

If $D_d$ is the degree of the polynomials corresponding to the gates in level $d$ of the circuit, then $D_{d+1} \leq D_d(2(p - 1) \log m)$ and $D_0 = 1$. Therefore $D_d \leq (2(p - 1) \log m)^d$, as desired. \qed