Chapter 13

Circuit lowerbounds

Complexity theory's Waterloo

We believe that **NP** does not have polynomial-sized circuits. We've seen that if true, this implies that $\mathbf{NP} \neq \mathbf{P}$. In the 1970s and 1980s, many researchers came to believe that the route to resolving **P** versus **NP** should go via circuit lowerbounds, since circuits seem easier to reason about than Turing machines. The success in this endeavor was mixed.

Progress on general circuits has been almost nonexistent: a lowerbound of n is trivial for any function that depends on all its input bits. We are unable to prove even a superlinear circuit lowerbound for any **NP** problem—the best we can do after years of effort is $4.5n - o(n)$.

To make life (comparatively) easier, researchers focussed on restricted circuit classes, and were successful in proving some decent lowerbounds. We prove some of the major results of this area and indicate where researchers are currently stuck. In Chapter 23 we'll explain some of the inherent obstacles that need to be overcome to make further progress.

13.1 $\mathbf{AC}^0$ and Håstad's Switching Lemma

As we saw in Chapter 6, $\mathbf{AC}^0$ is the class of languages computable by circuit families of constant depth, polynomial size, and whose gates have unbounded fanin. (Constant depth circuits with fanin 2 can only compute functions depending on a constant number of input bits.) The burning question in the late 1970s was whether problems like Clique and TSP have $\mathbf{AC}^0$ circuits. However, in 1981, Furst, Saxe and Sipser and independently, Ajtai, proved a lowerbound for a much simpler function:

Complexity Theory: A Modern Approach. © 2006 Sanjeev Arora and Boaz Barak.
References and attributions are still incomplete.
DRAFT

THEOREM 13.1 ([?, ?])
Let $\bigoplus$ be the parity function. That is, for every $x \in \{0,1\}^n$, $\bigoplus(x_1,\ldots,x_n) = \sum_{i=1}^n x_i \pmod 2$. Then $\bigoplus \notin \mathbf{AC}^0$.

Often courses in digital logic design teach students how to do "circuit minimization" using Karnaugh maps. Note that circuits talked about in those courses are depth 2 circuits, i.e. CNF or DNF. Indeed, it is easy to show (using for example the Karnaugh map technique studied in logic design) that the parity function requires exponentially many gates if the depth is two. However, those simple ideas do not seem to generalize to even depth 3 circuits.

The main tool in the proof of Theorem 13.1 is the concept of *random restrictions*. Let f be a function computable by a depth d circuit and suppose that we choose at random a vast majority (i.e., $n-n^\epsilon$ for some constant $\epsilon > 0$ depending on d) of the input variables and assign to each such variable either 0 or 1 at random. We'll prove that with positive probability, the function f subject to this restriction is *constant* (i.e., either always zero or always one). Since the parity function cannot be made a constant by fixing values to a subset of the variables, it follows that it cannot be computed by a constant depth circuit.

13.1.1 The switching lemma

Now we prove the main lemma about how a circuit simplifies under a random restriction. A k-DNF (resp. k-CNF) formula is an OR of AND's (resp. AND or OR's) where each AND (resp. OR) involves at most k variables.

LEMMA 13.2 (HÅSTAD'S SWITCHING LEMMA [?])
Suppose f is expressible as a k-DNF, and let ρ denote a random restriction that assigns random values to t randomly selected input bits. Then for every $s \geq 2$.

$$\mathbf{Pr}_\rho[f|_\rho \text{ is not expressible as } s\text{-CNF }] \leq \left(\frac{(n-t)k^{10}}{n}\right)^{s/2} \qquad (1)$$

where $f|_\rho$ denotes the function f restricted to the partial assignment ρ.

We'll typically use this lemma with k, s constant and $t \approx n - \sqrt{n}$ in which case the guaranteed bound on the probability will be n^{-c} for some constant c. Note that by applying the lemma to the function $\neg f$, we can get the same result with the terms DNF and CNF interchanged.

DRAFT

Proving Theorem 13.1 from Lemma 13.2. Now we show how Håstad's lemma implies that parity is not in $\mathbf{AC}^0$. We start with any $\mathbf{AC}^0$ circuit and assume that the circuit has been simplified as follows (the simplifications are straightforward to do and are left as Exercises 1 and 2): **(a)** All fanouts are 1; the circuit is a tree **(b)** All *not* gates to the input level of the circuit; equivalently, the circuit has $2n$ input wires, with the last n of them being the negations of the first n **(c)** $\vee$ and $\wedge$ gates alternate —at worst this assumption doubles the depth of the circuit **(d)** The bottom level has $\wedge$ gates of fanin 1.

We randomly restrict more and more variables, where each step with high probability will reduce the depth of the circuit by 1 and will keep the bottom level at a constant fanin. Specifically, letting n_i stand for the number of unrestricted variables after step i, we restrict $n_i - \sqrt{n_i}$ variables at step $i+1$. Since $n_0 = n$, we have $n_i = n^{1/2^i}$. Let n^b denote an upper bound on the number of gates in the circuit and let $k_i = 10b2^i$. We'll show that with high probability, after the i^{th} restriction we're left with a depth-$d-i$ circuit with at most k^i fanin in the bottom level. Indeed, suppose that the bottom level contains $\wedge$ gates and the level above it contains $\vee$ gates. The function each such $\vee$ gate computes is a k_i-DNF and hence by Lemma 13.2, with probability $1 - \left(\frac{k_i^{10}}{n^{1/2^{i+1}}}\right)^{k_{i+1}/2}$, which is at least $1 - 1/(10n^b)$ for large enough n, the function such a gate computes will be expressible as a k_{i+1}-CNF. We can then merge this CNF with the $\wedge$-gate above it, reducing the depth of the circuit by one (see Figures 13.1 and 13.2). The symmetric reasoning applies in the case the bottom level consists of $\vee$ gates— in this case we use the lemma to transform the k_i-CNF of the level above it into a k_{i+1}-DNF. Note that we apply the lemma at most once per each of the at most n^b gates of the original circuit. By the union bound, with probability $9/10$, if we continue this process for $d-2$ steps, we'll get a depth two circuit with fanin $k = k_{d-2}$ at bottom level (i.e., a k-CNF or k-DNF formula). If we then choose to restrict each variable with probability half (i.e., restrict about half of the variables to a random value), this circuit will be reduced to a constant function with probability at least 2^{-k}. Since the parity function is not constant under any restriction of less than n variables, this proves Theorem 13.1. ■

Figure unavailable in pdf file.

Figure 13.1: Circuit before Håstad switching transformation.

Web draft 2006-09-28 18:10
DRAFT

Figure unavailable in pdf file.

Figure 13.2: Circuit after Håstad switching transformation. Notice that the new layer of $\wedge$ gates can be collapsed with the single $\wedge$ parent gate, to reduce the number of levels by one.

13.1.2 Proof of the switching lemma (Lemma 13.2)

Now we prove the Switching Lemma. The original proof was more complicated; this one is due to Razborov. Let f be expressible as a k-DNF on n variables. Let t be as in the lemma and let $\mathcal{R}_t$ denote the set of all restrictions to t variables (note we can assume $t > n/2$). We have that $|\mathcal{R}_t| = \binom{n}{t}2^t$. Let $K_{t,s}$ denote the set of restrictions ρ such that $f|_\rho$ is not a s-CNF. We need to bound $|K_{t,s}|/|\mathcal{R}_t|$ by the right hand side of (1) to prove the lemma. We'll do that by showing a one-to-one function mapping $K_{t,s}$ into the set $Z \times S$ where Z is the set of restrictions of at least $t+s$ variables (i.e. $Z = \cup_{t' \geq t+s}\mathcal{R}_{t'}$) and S is some set of size 3^{2ks}. This will prove the lemma since at he range $t' \gg n/2$, $\binom{n}{t'} \approx \left(\frac{n}{n-t'}\right)^{n-t'}$ and hence Z will be of size bounded by roughly $n2^s\left(\frac{n-t}{n}\right)^s|\mathcal{R}_t|$. We leave verifying the exact bound as Exercise 3.

Mapping $K_{t,s}$ into $Z \times S$. Let $\rho \in K_{t,s}$ be a restriction fixing t variables such that $f|_\rho$ is not an s-CNF. We need to map ρ in a one-to-one way into some restriction ρ^* of at least $t+s$ variables, and some additional element in a set S of size at most 3^{2ks}.

Special case: each term has at most one "live" variable. To get some intuition for the proof, consider first the case that for each term t in the k-DNF formula for f, ρ either fixed t to the value 0 or left a *single* unassigned variable in t, in which case we say that t's value is `?` (ρ can't fix a term to the value 1 since we assume $f|_\rho$ is not constant). We denote by $x_1, \ldots, x_s$ denote the first s such unassigned variables, according to some canonical ordering of the terms for the k-DNF formula of f (there are more than s since otherwise $f|_\rho$ would be expressible as an s-CNF). For each such variable x_i, let $\texttt{term}_i$ be the `?`-valued term in which x_i appears. Let R_i be the operation of setting x_i to the value that ensures $\texttt{term}_i$ is true. We'll map ρ to $\tau_1 = R_1R_2\cdots R_s\rho$. That is, apply R_s to ρ, then apply R_{k-1} to ρ, $\cdots$, then apply R_1 to ρ. The crucial insight is that given τ_1, one can deduce $\texttt{term}_1$: this is the first term that is true in $f|_{\tau_1}$. One might think that the

DRAFT

second term that is true in $f|_{\tau_1}$ is $\texttt{term}_2$ but that's not necessarily the case, since the variable x_1 may have appeared several times, and so setting it to R_1 may have set other terms to true (it could not have set other terms to false, since this would imply that $f|_\rho$ includes an OR of x_i and $\neg x_i$, and hence is the constant one function). We thus supply as part of the mapping a string $w_1 \in \{0,1,\star\}^s$ that tells us the assignment of the k variables of $\texttt{term}_1$ in $\tau_2 = R_2 \cdots R_s \rho$. Given that information we can "undo" R_1 and move from τ_1 to τ_2. Now in τ_2, $\texttt{term}_2$ is the first satisfied term. Continuing on this way we see that from τ_1 (which is an assignment of at least $t+s$ variables) and strings $w_1, \ldots, w_s$ that are defined as above, we can recover ρ, implying that we have a one-to-one mapping that takes ρ into an assignment of at least $t+s$ variables and a sequence in $\{0,1,\star\}^{ks}$.

The general case. We now consider the general case, where some terms might have more than one unassigned variable in them. We let $\texttt{term}_1$ be the first ?-valued term in $f|_\rho$ and let x_1 be the first unassigned variable in $\texttt{term}_1$. Once again, we have an operation R_1 that will make $\texttt{term}_1$ true, although this time we think of R_1 as assigning to all the k variables in $\texttt{term}_1$ the unique value that makes the term true. We also have an operation L_1 assigning a value to x_1 such that $f|_{L_1\rho}$ cannot be expressed by an $s-1$-CNF. Indeed, if for both possible assignments to x_1 we get an $s-1$-CNF then $f|_\rho$ is an s-CNF. We note that it's not necessarily the case that x_1's value under $L_1\rho$ is different from its value under $R_1\rho$, but it is the case that $\texttt{term}_1$'s value is either ? or FALSE under $L_1\rho$ (since otherwise $f|_{L_1\rho}$ would be constant). We let $\texttt{term}_2$ be the first ?-valued term in $f|_{L_1\rho}$ (note that $\texttt{term}_2 \geq \texttt{term}_1$) and let x_2 be the first unassigned variable in $\texttt{term}_2$. Once again, we have an operation R_2 such that $\texttt{term}_2$ is the first true term in $f|_{R_2L_1\rho}$ and operation L_2 such that $f|_{L_2L_1\rho}$ is not a $s-2$-CNF. Continuing in this way we come up with operations $L_1, \ldots, L_s, R_1, \ldots, R_s$ such that if we let ρ_i be the assignment $L_i \cdots L_1 \rho$ (with $\rho_0 = \rho$) then for $1 \leq i \leq s$:

- $\texttt{term}_i$ is the first ?-valued term in $f|_{\rho_{i-1}}$.
- $\texttt{term}_i$ is the first true-valued term in $f|_{R_i\rho_{i-1}}$.
- L_i agrees with ρ_{i-1} on all variables assigned a value by ρ_{i-1}.
- R_i agrees with ρ_i on all variables assigned a value by ρ_i.

For $1 \leq i \leq s$, define τ_i to be $R_i R_{i+1} \cdots R_s \rho_s$, and define $\tau_{s+1} = \rho_s$. We have that $\texttt{term}_i$ is the first true term in $f|_{\tau_i}$: indeed, all the operations in τ_i

Web draft 2006-09-28 18:10
DRAFT

do not change variables assigned values by ρ_{i-1} and there $\texttt{term}_i$ is the first ?-valued term. Thus τ_i cannot make any earlier term true. However, since the last operation applied is R_i, $\texttt{term}_i$ is true in $f|_{\tau_i}$.

Let $z_1, \ldots, z_s$ and $w_1, \ldots, w_s$ be $2s$ strings in $\{0,1,\star\}^s$ defined as follows: z_i describes the values assigned to the k variables appearing in $\texttt{term}_i$ by ρ_{i-1} and w_i describes the value assigned to $\texttt{term}_i$'s variables by τ_{i+1}. Clearly, from $\texttt{term}_i$, z_i and the assignment ρ_i one can compute ρ_{i-1} and from $\texttt{term}_i$, w_i and the assignment τ_i one can compute τ_{i+1}. We'll map ρ to τ_1 and the sequence $z_1, \ldots, z_s, w_1, \ldots, w_s$. Note that τ_1 does assign values to at least s variables not assigned by ρ, and that from τ_1 we can find $\texttt{term}_1$ (as this is the first true term in $f|_{\tau_1}$) and then using w_1 recover τ_2 and continue in this way until we recover the original assignment ρ. Thus this mapping is a one-to-one map from $T_{t,s}$ to $Z \times \{0,1,\star\}^{2ks}$. ■

13.2 Circuits With "Counters":ACC

One way to extend the $\mathbf{AC}^0$ lowerbounds of the previous section was to define a more general class of circuits. What if we allow more general gates? The simplest example is a parity gate. Clearly, an $\mathbf{AC}^0$ circuit provided with parity gates can can compute the parity function. But are there still other functions that it cannot compute? Razborov proved the first such lowerbound using his *Method of Approximations*. Smolensky later extended this work and clarified this method for the circuit class considered here.

Normally we think of a modular computation as working with numbers rather than bit, but it is sufficient to consider modular gates whose output is always 0/1.

Definition 13.3 (modular gates)
For any integer m, the MOD_m gate outputs 0 if the sum of its inputs is 0 modulo m, and 1 otherwise.

Definition 13.4 (**ACC**)
For integers $m_1, m_2, \ldots, m_k > 1$ we say a language L is in $\mathbf{ACC}^0[m_1, m_2, \ldots, m_k]$ if there exists a circuit family $\{C_n\}$ with constant depth and polynomial size (and unbounded fan-in) consisting of $\wedge$, $\vee$, $\neg$ and $MOD_{m_1}, \ldots, MOD_{m_k}$ gates accepting L.

The class $\mathbf{ACC}^0$ contains every language that is in $\mathbf{ACC}^0(m_1, m_2, \ldots, m_k)$ for some $k \geq 0$ and $m_1, m_2, \ldots, m_k > 1$.

Good lowerbounds are known only when the circuit has one kind of modular gate.

DRAFT

THEOREM 13.5 (RAZBOROV,SMOLENSKY)
For distinct primes p and q, the function MOD_p is not in $\mathbf{ACC}^0(q)$.

We exhibit the main idea of this result by proving that the parity function cannot be computed by an $\mathbf{ACC}^0(3)$ circuit.
PROOF: The proof proceeds in two steps.

Step 1. In the first step, we show (using induction on h) that for any depth h MOD_3 circuit on n inputs and size S, there is a polynomial of degree $(2l)^h$ which agrees with the circuit on $1 - S/2^l$ fraction of the inputs. If our circuit C has depth d then we set $2l = n^{1/2d}$ to obtain a degree $\sqrt{n}$ polynomial that agrees with C on $1 - S/2^{n^{1/2d}/2}$ fraction of inputs.

Step 2 We show that no polynomial of degree $\sqrt{n}$ agrees with MOD_2 on more than $49/50$ fraction of inputs.

Together, the two steps imply that $S > 2^{n^{1/2d}/2}/50$ for any depth d circuit computing MOD_2, thus proving the theorem. Now we give details.
Step 1. Consider a node g in the circuit at a depth h . (The input is assumed to have depth 0.) If $g(x_1, \cdots, x_n)$ is the function computed at this node, we desire a polynomial $\tilde{g}(x_1, \cdots, x_n)$ over $GF(3)$ with degree $(2l)^h$, such that $g(x_1, \ldots, x_n) = \tilde{g}(x_1, \ldots, x_n)$ for "most" $x_1, \ldots, x_n \in \{0, 1\}$. We will also ensure that on every input in $\{0, 1\}^n \subseteq GF(3)$, polynomial $\tilde{g}$ takes a value in $\{0, 1\}$. This is without loss of generality since we can just square the polynomial. (Recall that the elements of $GF(3)$ are $0, -1, 1$ and $0^2 = 0$, $1^2 = 1$ and $(-1)^2 = 1$.)

We construct the approximator polynomial by induction. When $h = 0$ the "gate" is an input wire x_i, which is exactly represented by the degree 1 polynomial x_i. Suppose we have constructed approximators for all nodes up to height $h - 1$ and g is a gate at height h.

1. If g is a NOT gate, then $g = \neg f_1$ for some other gate f_1 that is at height $h - 1$ or less. The inductive hypothesis gives an approximator $\tilde{f}_1$ for f_1. Then we use $\tilde{g} = 1 - \tilde{f}_1$ as the approximator polynomial for g; this has the same degree as $\tilde{f}_1$. Whenever $\tilde{f}_1 = f_1$ then $\tilde{g} = g$, so we introduced no new error.

2. If g is a MOD_3 gate with inputs $f_1, f_2, \ldots, f_k$, we use the approximation $\tilde{g} = (\sum_{i=0}^{k} \tilde{f}_i)^2$. The degree increases to at most $2 \times (2l)^{h-1} < (2l)^h$. Since $0^2 = 0$ and $(-1)^2 = 1$, we introduced no new error.

Web draft 2006-09-28 18:10
DRAFT

3. If g is an AND or an OR gate, we need to be more careful. Suppose $g = \wedge_{i=0}^{k} f_i$. The naive approach would be to replace g with the polynomial $\Pi_{i \in I} \tilde{f}_i$. For an OR gate $g = \vee_{i=0}^{k} f_i$ De Morgan's law gives a similar naive approximator $1 - \prod_{i \in I}(1 - \tilde{f}_i)$. Unfortunately, both of these multiply the degree by k, the fanin of the gate, which could greatly exceed $2l$.

   The correct solution involves introducing some error. We give the solution for OR; De Morgan's law allows AND gates to be handled similarly.

   If $g = \vee_{i=0}^{k} f_i$, then $g = 1$ if and only if at least one of the $f_i = 1$. Furthermore, by the *random subsum principle* (see Section **??** in the Appendix) if any of the $f_i = 1$, then the sum (over $GF(3)$) of a random subset of $\{f_i\}$ is nonzero with probability at least $1/2$.

   Randomly pick l subsets $S_1, \cdots, S_l$ of $\{1, \ldots, k\}$. Compute the l polynomials $(\sum_{j \in S_i} \tilde{f}_j)^2$, each of which has degree at most twice that of the largest input polynomial. Compute the OR of these l terms using the naive approach. We get a polynomial of degree at most $2l \times (2l)^{h-1} = (2l)^h$. For any x, the probability over the choice of subsets that this polynomial differs from $OR(\tilde{f}_1, \ldots, \tilde{f}_k)$ is at most $\frac{1}{2^l}$. So, by the probabilistic method, there *exists* a choice for the l subsets such that the probability over the choice of x that this polynomial differs from $OR(\tilde{f}_1, \cdots, \tilde{f}_k)$ is at most $\frac{1}{2^l}$. We use this choice of the subsets to construct the approximator.

Applying the above procedure for each gate gives an approximator for the output gate of degree $(2l)^d$ where d is depth of the entire circuit. Each operation of replacing the gate by its approximator polynomial introduces error on at most $1/2^l$ fraction of all inputs, so the overall fraction of erroneous inputs for the approximator is at most $S/2^l$. (Note that errors at different gates may affect each other. Error introduced at one gate may be cancelled out by errors at another gate higher up. We are being pessimistic in applying the union bound to *upperbound* the probability that any of the approximator polynomials anywhere in the circuit miscomputes.)

Step 2. Suppose that a polynomial f agrees with the MOD_2 function for all inputs in a set $G' \subseteq 0, 1^n$. If the degree of f is bounded by $\sqrt{n}$, then we show $|G'| < \left(\frac{49}{50}\right) 2^n$.

Consider the change of variables $y_i = 1 + x_i \pmod 3$. (Thus $0 \to 1$ and $1 \to -1$.) Then, G' becomes some subset G of $\{-1, 1\}^n$, and f becomes some other polynomial, say $g(y_1, y_2, \ldots, y_n)$, which still has degree $\sqrt{n}$. Moreover,

DRAFT

$$MOD_2(x_1, x_2, \ldots, x_n) = \begin{cases} 1 & \Rightarrow \Pi_{i=1}^n y_i = -1 \\ 0 & \Rightarrow \Pi_{i=1}^n y_i = 1 \end{cases} \quad (2)$$

Thus $g(y_1, y_2, \ldots, y_n)$, a degree $\sqrt{n}$ polynomial, agrees with $\Pi_{i=1}^n y_i$ on G. This is decidedly odd, and we show that any such G must be small. Specifically, let F_G be the set of all functions $S : G \to \{0, 1, -1\}$. Clearly, $|F_G| = 3^{|G|}$, and we will show $|F_G| \leq 3^{\left(\frac{49}{50}\right)2^n}$, whence Step 2 follows.

LEMMA 13.6
For every $S \in F_G$, there exists a polynomial g_S which is a sum of monomials $a_I \prod_{i \in I} y_i$ where $|I| \leq \frac{n}{2} + \sqrt{n}$ such that $g_S(x) = S(x)$ for all $x \in G$.

PROOF: Let $\hat{S} : GF(3)^n \to GF(3)$ be any function which agrees with S on G. Then $\hat{S}$ can be written as a polynomial in the variables y_i. However, we are only interested in its values on $(y_1, y_2, \ldots, y_n) \in \{-1, 1\}^n$, when $y_i^2 = 1$ and so every monomial $\Pi_{i \in I} y_i^{r_i}$ has, without loss of generality, $r_i \leq 1$. Thus $\hat{S}$ is a polynomial of degree at most n. Now consider any of its monomial terms $\Pi_{i \in I} y_i$ of degree $|I| > n/2$. We can rewrite it as

$$\Pi_{i \in I} y_i = \Pi_{i=1}^n y_i \Pi_{i \in \bar{I}} y_i, \quad (3)$$

which takes the same values as $g(y_1, y_2, \ldots, y_n)\Pi_{i \in \bar{I}} y_i$ over $\{-1, 1\}^n$. Thus every monomial in $\hat{S}$ has degree at most $\frac{n}{2} + \sqrt{n}$. ■

To conclude, we bound the number of polynomials whose every monomial with a degree at most $\frac{n}{2} + \sqrt{n}$. Clearly this number is #polynomials $\leq 3^{\#\text{monomials}}$, and

$$\#\text{monomials} \leq \left| \{N \subseteq \{1 \cdots n\} | \, |N| \leq \frac{n}{2} + \sqrt{n} \right| \quad (4)$$

$$\leq \sum_{i \leq \frac{n}{2} + \sqrt{n}} \binom{n}{i} \quad (5)$$

Using knowledge of the tails of a binomial distribution (or alternatively, direct calculation),

$$\leq \frac{49}{50} 2^n \quad (6)$$

■

Web draft 2006-09-28 18:10
DRAFT

13.3 Lowerbounds for monotone circuits

A Boolean circuit is *monotone* if it contains only AND and OR gates, and no NOT gates. Such a circuit can only compute monotone functions, defined as follows.

DEFINITION 13.7
For $x, y \in \{0,1\}^n$, we denote $x \preccurlyeq y$ if every bit that is 1 in x is also 1 in y. A function $f:\{0,1\}^n \to \{0,1\}$ is *monotone* if $f(x) \leq f(y)$ for every $x \preccurlyeq y$.

REMARK 13.8
An alternative characterization is that f is *monotone* if for every input x, changing a bit in x from 0 to 1 cannot change the value of the function from 1 to 0.

It is easy to check that every monotone circuit computes a monotone function, and every monotone function can be computed by a (sufficiently large) monotone circuit. CLIQUE is a monotone function since adding an edge to the graph cannot destroy any clique that existed in it. In this section we show that the CLIQUE function can not be computed by polynomial (and in fact even subexponential) sized monotone circuits:

THEOREM 13.9 ([?, ?])
Denote by $\mathsf{CLIQUE}_{k,n} : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ *be the function that on input an adjacency matrix of an* n*-vertex graph* G *outputs* 1 *iff* G *contains a* k*-vertex clique.*

There exists some constant $\epsilon > 0$ *such that for every* $k \leq n^{1/4}$*, there's no monotone circuit of size less than* $2^{\epsilon\sqrt{k}}$ *that computes* $\mathsf{CLIQUE}_{k,n}$*.*

We believe CLIQUE does not have polynomial-size circuits even allowing NOT gates (i.e., that $\mathbf{NP} \not\subseteq \mathbf{P}/\mathrm{poly}$). In fact, a seemingly plausible approach to proving this might be to show that for *every* monotone function f, the monotone circuit complexity of f is polynomially related to the general (non-monotone) circuit complexity. Alas, this conjecture was refuted by Razborov ([?], see also [?]).

13.3.1 Proving Theorem 13.9

Clique Indicators

To get some intuition why this theorem might be true, lets show that $\mathsf{CLIQUE}_{k,n}$ can't be computed (or even approximated) by subexponential monotone circuits of a very special form. For every $S \subseteq [n]$, let C_S denote the

DRAFT

function on $\{0,1\}^{\binom{n}{2}}$ that outputs 1 on a graph G iff the set S is a clique in G. We call C_S the *clique indicator* of S. Note that $\mathsf{CLIQUE}_{k,n} = \bigvee_{S\subseteq[n],|S|=k} \mathsf{C}_S$. We'll now prove that $\mathsf{CLIQUE}_{k,n}$ can't be computed by an OR of less than $n^{\sqrt{k}/20}$ clique indicators.

Let $\mathcal{Y}$ be the following distribution on n-vertex graphs: choose a set $K \subseteq [n]$ with $|K| = k$ at random, and output the graph that has a clique on K and no other edges. Let $\mathcal{N}$ be the following distribution on n-vertex graphs: choose a function $c : [n] \to [k-1]$ at random, and place an edge between u and v iff $c(u) \neq c(v)$. With probability one, $\mathsf{CLIQUE}_{n,k}(\mathcal{Y}) = 1$ and $\mathsf{CLIQUE}_{n,k}(\mathcal{N}) = 0$. The fact that $\mathsf{CLIQUE}_{n,k}$ requires an OR of at least $n^{\sqrt{k}/20}$ clique indicators follows immediately from the following lemma:

LEMMA 13.10
Let n be sufficiently large, $k \leq n^{1/4}$ and $S \subseteq [n]$. Then either $\mathbf{Pr}[\mathsf{C}_S(\mathcal{N}) = 1] \geq 0.99$ *or* $\mathbf{Pr}[\mathsf{C}_S(\mathcal{Y}) = 1] \leq n^{-\sqrt{k}/20}$

PROOF: Let $\ell = \sqrt{k-1}/10$. If $|S| \leq \ell$ then by the birthday bound, we expect a random $f : S \to [k-1]$ to have less than 0.01 collisions and hence by Markov the probability f is one to one is at least 0.99. This implies that $\mathbf{Pr}[\mathsf{C}_S(\mathcal{N}) = 1] \geq 0.99$.

If $|S| > \ell$ then $\mathbf{Pr}[\mathsf{C}_S(\mathcal{Y}) = 1]$ is equal to the probability that $S \subseteq K$ for a random $K \subseteq [n]$ of size k. This probability is equal to $\binom{n-\ell}{k-\ell}/\binom{n}{k}$ which is at most $\binom{n}{k-\sqrt{k-1}/10}/\binom{n}{k}$ which, by the formula for the binomial coefficients, is less than $\left(\frac{2k}{n}\right)^{\ell} \leq n^{-0.7\ell} < n^{-\sqrt{k}/20}$ (for sufficiently large n). ■

Approximation by clique indicators.

Together with Lemma 13.10, the following lemma implies Theorem 13.9:

LEMMA 13.11
Let C be a monotone circuit of size s. Let $\ell = \sqrt{k}/10$. Then, there exist sets $S_1, \ldots, S_m$ with $m \leq n^{\sqrt{k}/20}$ such that

$$\mathbf{Pr}_{G\in_R\mathcal{Y}}[\bigvee_i \mathsf{C}_{S_i}(G) \geq C(G)] > 0.9 \tag{7}$$

$$\mathbf{Pr}_{G\in_R\mathcal{N}}[\bigvee_i \mathsf{C}_{S_i}(G) \leq C(G)] > 0.9 \tag{8}$$

$$\tag{9}$$

Web draft 2006-09-28 18:10
DRAFT

PROOF: Set $\ell = \sqrt{k}/10$, $p = 10\sqrt{k}\log n$ and $m = (p-1)^\ell \ell!$. Note that $m \ll n^{\sqrt{k}/20}$. We can think of the circuit C as the sequence of s monotone functions $f_1, \ldots, f_s$ from $\{0,1\}^{\binom{n}{2}}$ to $\{0,1\}$ where each function f_k is either the AND or OR of two functions $f_{k'}, f_{k''}$ for $k', k'' < k$ or is the value of an input variable $x_{u,v}$ for $u, v \in [n]$ (i.e., $f_k = \mathsf{C}_{\{u,v\}}$). The function that C computes is f_s. We'll show a sequence of functions $\tilde{f}_1, \ldots, \tilde{f}_s$ such that each function $\tilde{f}_k$ is **(1)** an OR of at most m clique indicators $\mathsf{C}_{S_1}, \ldots, \mathsf{C}_{S_m}$ with $|S_i| \leq \ell$ and **(2)** $\tilde{f}_k$ approximates f_k in the sense of (7) and (8). We call a function $\tilde{f}_k$ satisfying **(1)** an (ℓ, m)-*function*. The result will follow by considering the function $\tilde{f}_s$.

We construct the functions $\tilde{f}_1, \ldots, \tilde{f}_s$ by induction. For $1 \leq k \leq s$, if f_k is an input variable then we let $\tilde{f}_k = f_k$. If $f_k = f_{k'} \vee f_{k''}$ then we let $\tilde{f}_{k'} \sqcup \tilde{f}_{k''}$ and if $f_k = f_{k'} \wedge f_{k''}$ then we let $\tilde{f}_{k'} \sqcap \tilde{f}_{k''}$, where the operations $\sqcup, \sqcap$ will be defined below. We'll prove that for every $f, g : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$ **(a)** if f and g are (m,ℓ)-functions then so is $f \sqcup g$ (resp. $f \sqcap g$) and **(b)** $\mathbf{Pr}_{G \in_R \mathcal{Y}}[f \sqcup g\ (G) < f \cup g\ (G)] < 1/(10S)$ (resp. $\mathbf{Pr}_{G \in_R \mathcal{Y}}[f \sqcap g\ (G) < f \cap g\ (G)] < 1/(10S)$) and $\mathbf{Pr}_{G \in_R \mathcal{N}}[f \sqcup g\ (G) > f \cup g\ (G)] < 1/(10S)$ (resp. $\mathbf{Pr}_{G \in_R \mathcal{Y}}[f \sqcap g\ (G) < f \cap g\ (G)] < 1/(10S)$). The lemma will then follow by showing using the union bound that with probability ≥ 0.9 the equations of Condition **(b)** hold for all $\tilde{f}_1, \ldots, \tilde{f}_s$. We'll now describe the two operations $\sqcup, \sqcap$. Condition **(a)** will follow from the definition of the operations, while Condition **(b)** will require a proof.

The operation f $\sqcup$ g. Let f, g be two (m,ℓ)-functions: that is $f = \bigvee_{i=1}^m \mathsf{C}_{S_i}$ and $g = \bigvee_{j=1}^m \mathsf{C}_{T_j}$ (if f or g is the OR of less than m clique indicators we can add duplicate sets to make the number m). Consider the function $h = \mathsf{C}_{Z_1} \cup \cdots \cup \mathsf{C}_{Z_{2m}}$ where $Z_i = S_i$ and $Z_{m+j} = T_j$ for $1 \leq i, j \leq m$. The function h is not an (m,ℓ)-function since it is the OR of $2m$ clique indicators. We make it into an (m,ℓ)-function in the following way: as long as there are more than m distinct sets, find p subsets $Z_{i_1}, \ldots, Z_{i_p}$ that are in a *sunflower* formation. That is, there exists a set $Z \subseteq [n]$ such that for every $1 \leq j, j' \leq p$, $Z_{i_j} \cap Z_{i,j'} = Z$. Replace the functions $\mathsf{C}_{Z_{i_1}}, \ldots, \mathsf{C}_{Z_{i_p}}$ in the function h with the function C_Z. Once we obtain an (m,ℓ)-function h' we define $f \sqcup g$ to be h'. We won't get stuck because of the following lemma (whose proof we defer):

LEMMA 13.12 (SUNFLOWER LEMMA [**?**])
Let $\mathcal{Z}$ be a collection of distinct sets each of cardinality at most ℓ. If $|\mathcal{Z}| > (p-1)^\ell \ell!$ then there exist p sets $Z_1, \ldots, Z_p \in \mathcal{Z}$ and set Z such that $Z_i \cap Z_j = Z$ for every $1 \leq i, j \leq p$.

DRAFT

The operation f ⊓ g. Let f, g be two (m, ℓ)-functions: that is $f = \bigvee_{i=1}^{m} \mathsf{C}_{S_i}$ and $g = \bigvee_{j=1}^{m} \mathsf{C}_{T_j}$. Let h be the function $\bigvee_{1 \leq i,j \leq m} \mathsf{C}_{S_i \cup T_j}$. We perform the following steps on h: **(1)** Discard any function C_Z for $|Z| > \ell$. **(2)** Reduce the number of functions to m by applying the sunflower lemma as above.

Proving Condition (b). To complete the proof of the lemma, we prove the following four equations:

- $\mathbf{Pr}_{G \in_R \mathcal{Y}}[f \sqcup g\ (G) < f \cup g\ (G)] < 1/(10S)$.

  If $Z \subseteq Z_1, \ldots, Z_p$ then for every i, $\mathsf{C}_{Z_i}(G)$ implies that $\mathsf{C}_Z(G)$ and hence the operation $f \sqcup g$ can't introduce any "false negatives".

- $\mathbf{Pr}_{G \in_R \mathcal{N}}[f \sqcup g\ (G) > f \cup g\ (G)] < 1/(10S)$.

  We can introduce a "false positive" on a graph G only if when we replace the clique indicators for a sunflower $Z_1, \ldots, Z_p$ with the clique indicator for the common intersection Z, it is the case that $\mathsf{C}_Z(G)$ holds even though $\mathsf{C}_{Z_i}(G)$ is false for every i. Recall that we choose $G \in_R \mathcal{N}$ by choosing a random function $c : [n] \to [k-1]$ and adding an edge for every two vertices u, v with $c(u) \neq c(v)$. Thus, we get a false positive if c is one-to-one on Z (we denote this event by B) but *not* one-to-one on Z_i for every $1 \leq i \leq p$ (we denote these events by $A_1, \ldots, A_p$). We'll show that the intersection of B and $A_1, \ldots, A_p$ happens with probability at most 2^{-p} which (by the choice of p) is less than $1/(10m^2 s)$. Since we apply the reduction step at most m times the equation will follow.

  Since $\ell < \sqrt{k-1}/10$, for every i, $\mathbf{Pr}[A_i|B] < 1/2$ (the probability that there'll be a collision on the at most ℓ elements of $Z_i \setminus Z$ is less than half). Conditioned on B, the events $A_1, \ldots, A_p$ are independent, since they depend on the values of c on disjoint sets, and hence we have that $\mathbf{Pr}[A_1 \wedge \cdots \wedge A_p \wedge B] \leq \mathbf{Pr}[A_1 \wedge \cdots \wedge A_p|B] = \prod_{i=1}^{p} \mathbf{Pr}[A_p|B] \leq 2^{-p}$.

- $\mathbf{Pr}_{G \in_R \mathcal{Y}}[f \sqcap g\ (G) < f \cap g\ (G)] < 1/(10S)$.

  By the distributive law $f \cap g = \bigvee_{i,j}(\mathsf{C}_{S_i} \cap \mathsf{C}_{T_j})$. A graph G in the support of $\mathcal{Y}$ consists of a clique over some set K. For such a graph $\mathsf{C}_{S_i} \cap \mathsf{C}_{T_j}$ holds iff $S_i, T_j \subseteq K$ and thus $\mathsf{C}_{S_i} \cap \mathsf{C}_{T_j}$ holds iff $\mathsf{C}_{S_i \cup T_j}$ holds. We can introduce a false negative when we discard functions of the form C_Z for $|Z| > \ell$, but by Lemma 13.10, for such sets Z, $\mathbf{Pr}[\mathsf{C}_Z(\mathcal{Y}) = 1] < n^{-\sqrt{k}/20} < 1/(10sm^2)$. The equation follows since we discard at most m^2 such sets.

Web draft 2006-09-28 18:10
DRAFT

- $\mathbf{Pr}_{G\in_R\mathcal{N}}[f\sqcap g\ (G) > f\cap g\ (G)] < 1/(10S)$.

  Since $\mathsf{C}_{S\cup T}$ implies both C_S and C_T, we can't introduce false positives by moving from $f\cap g$ to $\bigvee_{i,j}\mathsf{C}_{S_i\cup T_j}$. We can't introduce false positives by discarding functions from the OR. Thus, the only place where we can introduce false positives is where we replace the clique indicators of a sunflower with the clique indicator of the common intersection. We bound this probability in the same way as this was done for the $\sqcup$ operator.

■

Proof of the sunflower lemma (Lemma 13.12). The proof is by induction on ℓ. The case $\ell = 1$ is trivial since distinct sets of size 1 must be disjoint. For $\ell > 1$ let $\mathcal{M}$ be a maximal subcollection of $\mathcal{Z}$ containing only disjoint sets. Because of $\mathcal{M}$'s maximality for every $Z \in \mathcal{Z}$ there exists $x \in \cup\mathcal{M} = \cup_{M\in\mathcal{M}}M$ such that $x \in Z$. If $|\mathcal{M}| \geq p$ we're done, since such a collection is already a sunflower. Otherwise, since $|\cup\mathcal{M}| \leq (p-1)\ell$ by averaging there's an $x \in \cup\mathcal{M}$ that appears in at least a $\frac{1}{\ell(p-1)}$ fraction of the sets in $\mathcal{Z}$. Let $Z_1,\ldots,Z_k$ be the sets containing x, and note that $k > (p-1)^{\ell-1}(\ell-1)!$. Thus, by induction there are p sets among the $\ell-1$-sized sets $Z_1\setminus\{x\},\cdots,Z_k\setminus\{x\}$ that form a sunflower, adding back x we get the desired sunflower among the original sets. Note that the statement (and proof) assume nothing about the size of the universe the sets in $\mathcal{Z}$ live in. ■

13.4 Circuit complexity: The frontier

Now we sketch the "frontier" of circuit lowerbounds, namely, the dividing line between what we can prove and what we cannot. Along the way we also define multi-party communication, since it may prove useful for proving some new circuit lowerbounds.

13.4.1 Circuit lowerbounds using diagonalization

We already mentioned that the best lowerbound on circuit size for an **NP** problem is $4.5n - o(n)$. For **PH** better lowerbounds are known: one of the exercises in Chapter 6 asked you to show that some for every $k > 0$, some language in **PH** (in fact in $\mathbf{\Sigma}_2^p$) requires circuits of size $\Omega(n^k)$. The latter

DRAFT

lowerbound uses diagonalization, and one imagines that classes "higher up" than **PH** should have even harder languages.

Frontier 1: Does **NEXP** have languages that require super-polynomial size circuits?

If we go a little above **NEXP**, we can actually prove a super-polynomial lowerbound: we know that $\mathbf{MA_{EXP}} \not\subseteq \mathbf{P}/\text{poly}$ where $\mathbf{MA_{EXP}}$ is the set of languages accepted by a one round proof with an all powerful prover and an exponential time *probabilistic* verifier. This follows from the fact that if $\mathbf{MA_{EXP}} \subseteq \mathbf{P}/\text{poly}$ then in particular $\mathbf{PSPACE} \subseteq \mathbf{P}/\text{poly}$. However, by $\mathbf{IP} = \mathbf{PSPACE}$ (Theorem 9.13) we have that in this case $\mathbf{PSPACE} = \mathbf{MA}$ (the prover can send in one round the circuit for computing the prover strategy in the interactive proof). However, by simple padding this implies that $\mathbf{MA_{EXP}}$ equals the class of languages in *exponential space*, which can be directly shown to not contain $\mathbf{P}/\text{poly}$ using diagonalization. Interestingly, this lower bound does not relativize (i.e., there's an oracle under which $\mathbf{MA_{NEXP}} \subseteq \mathbf{P}/\text{poly}$ [**?**]).

13.4.2 Status of ACC versus P

The result that PARITY is not in $\mathbf{AC}^0$ separates $\mathbf{NC}^1$ from $\mathbf{AC}^0$. The next logical step would be to separate $\mathbf{ACC}^0$ from $\mathbf{NC}^1$. Less ambitiously, we would like to show even a function in **P** or **NP** that is not in $\mathbf{ACC}^0$.

The Razborov-Smolenksy method seems to fail when we allow the circuit even two types of modular gates, say MOD_2 and MOD_3. In fact if we allow the bounded depth circuit modular gates that do arithmetic mod q, when q is not a prime —a prime power, to be exact— we reach the limits of our knowledge. (The exercises ask you to figure out why the proof of Theorem 13.5 does not seem to apply when the modulus is a composite number.) To give one example, it it is consistent with current knowledge that the majority of n bits can be computed by linear size circuits of constant depth consisting entirely of MOD_6 gates. The problem seems to be that low-degree polynomials modulo m where m is composite are surprisingly expressive [**?**].

Frontier 2: Show Clique is not in $\mathbf{ACC}^0(6)$.

Or even less ambitiously:

Frontier 2.1: Exhibit a language in **NEXP** that is not in $\mathbf{ACC}^0(6)$.

It is worth noting that thus far we are talking about *nonuniform* circuits (to which Theorem 13.5 also applies). Stronger lower bounds are known for

DRAFT

Figure unavailable in pdf file.

Figure 13.3: The depth 2 circuit with a symmetric output gate from Theorem 13.13.

uniform circuits: Allender and Gore [**?**] have shown that a decision version of the Permanent (and hence the Permanent itself) requires exponential size "Dlogtime-uniform" $\mathbf{ACC}^0$ circuits. (A circuit family $\{C_n\}$ is *Dlogtime uniform* if there exists a deterministic Turing machine M that given a triple (n, g, h) determines in linear time —i.e., $O(\log n)$ time when $g, h \leq \text{poly}(n)$— what types of gates g and h are and whether g is h's parent in C_n.)

But going back to nonuniform $\mathbf{ACC}^0$, we wish to mention an alternative representation of $\mathbf{ACC}^0$ circuits that may be useful in further lowerbounds. Let a *symmetric* gate be a gate whose output depends only on the number of inputs that are 1. For example, majority and mod gates are symmetric. Yao has shown that $\mathbf{ACC}^0$ circuits can be simplified to give an equivalent depth 2 circuits with a symmetric gate at the output (figure **??**). Beigel and Tarui subsequently improved Yao's result:

THEOREM 13.13 (YAO [**?**], BEIGEL AND TARUI [**?**])
If $f \in \mathbf{ACC}^0$, then f can be computed by a depth 2 circuit C with a symmetric gate with quasipolynomial (i.e., $2^{\log^k n}$) fan-in at the output level and $\vee$ gates with polylogarithmic fan-in at the input level.

We will revisit this theorem below in Section 13.5.1.

13.4.3 Linear Circuits With Logarithmic Depth

When we restrict circuits to have bounded fanin we necessarily need to allow them to have nonconstant (in fact, $\Omega(\log n)$) depth to have any reasonable power. With this in mind, the simplest interesting circuit class seems to be one of circuits wth linear size and logarithmic depth.

Frontier 3: Find an explicit function that cannot be computed by circuits of linear size and logarithmic depth.

(Note that by counting one can easily show that *some* function on n bits requires superpolynomial size circuits and hence bounded fan-in circuits with more than logarithmic depth; see the exercises on the chapter on circuits. Hence we want to show this for an explicit function, e.g. CLIQUE.)

Valiant thought about this problem in the '70s. His initial candidates for lowerbounds boiled down to showing that a certain graph called a *su-*

DRAFT

perconcentrator needed to have superlinear size. He failed to prove thisand instead ended up proving that such superconcentrators do exist!

Another sideproduct of Valiant's investigations was the following important lemma concerning depth-reduction for such circuits.

LEMMA 13.14 (VALIANT)
In any circuit with m edges and depth d, there are $km/\log d$ edges whose removal leaves a circuit with depth at most $d/2^{k-1}$.

This lemma can be applied as follows. Suppose we have a circuit C of depth $c \log n$ with n inputs $\{x_1, \ldots, x_n\}$ and n outputs $\{y_1, \ldots, y_n\}$, and suppose $2^k \sim c/\epsilon$ where $\epsilon > 0$ is arbitrarily small. Removing $O(n/\log\log n)$ edges from C then results in a circuit with depth at most $\epsilon \log n$. But then, since C has *bounded* fan-in, we must have that each output y_i is connected to at most $2^{\epsilon \log n} = n^\epsilon$ inputs. So each output y_i in C is completely determined by n^ϵ inputs and the values of the omitted edges. So we have a "dense" encoding for the function $f_i(x_1, \ldots, x_n) = y_i$. We do not expect this to be the case for any reasonably difficult function.

13.4.4 Branching Programs

Just as circuits are used to investigate time requirements of Turing Machines, *branching programs* are used to investigate space complexity.

A branching program on n input variables $x_1, x_2, \ldots, x_n$ is a directed acyclic graph all of whose nodes of nonzero outdegree are labeled with a variable x_i. It has two nodes of outdegree zero that are labeled with an output value, ACCEPT or REJECT. The edges are labeled by 0 or 1. One of the nodes is designated the start node. A setting of the input variables determines a way to walk on the directed graph from the start node to an output node. At any step, if the current node has label x_i, then we take an edge going out of the node whose label agrees with the value of x_i. The branching program is *deterministic* if every nonoutput node has exactly one 0 edge and one 1 edge leaving it. Otherwise it is *nondeterministic*. The *size* of the branching program is the number of nodes in it. The branching program complexity of a language is defined analogously with circuit complexity. Sometimes one may also require the branching program to be *leveled*, whereby nodes are arranged into a sequence of levels with edges going only from one level to the next. Then the *width* is the size of the largest level.

Web draft 2006-09-28 18:10
DRAFT

Figure unavailable in pdf file.

Figure 13.4: If f is computed by the above circuit, then f has a k-party protocol of complexity $k \log N$.

THEOREM 13.15
*If $S(n) \geq \log n$ and $L \in$ **SPACE**$(S(n))$ then L has branching program complexity at most $c^{S(n)}$ for some constant $c > 1$.*

PROOF: Essentially mimics our proof of Theorem'**??** that **SPACE**$(S(n)) \subseteq$ **DTIME**$(2^{O(S(n))})$. The nodes of the branching program correspond to the configurations of the space-bounded TM, and it is labeled with variable x_i if the configuration shows the TM reading the ith bit in the input. ■

Of course, a similar theorem is true about NDTMs and nondeterministic branching program complexity.

Frontier 4: Describe a problem in **P** (or even **NP**) that requires branching programs of size greater than $n^{1+\epsilon}$ for some constant $\epsilon > 0$.

There is some evidence that branching programs are more powerful than one may imagine. For instance, branching programs of constant width (reminiscent of a TM with $O(1)$ bits of memory) seem inherently weak. Thus the next result is unexpected.

THEOREM 13.16 (BARRINGTON [**?**])
A language has polynomial size, width 5 branching programs iff it is in $\mathbf{NC}^1$.

13.5 Approaches using communication complexity

Here we outline a concrete approach (rather, a setting) in which better lower-bounds may lead to a resolution of some of the questions above. It relates to generalizations of communication complexity introduced earlier. Mostly we will use *multiparty communication complexity*, though Section 13.5.4 will use *communication complexity of a relation*.

13.5.1 Connection to $\mathbf{ACC}^0$ Circuits

Suppose $f(x_1, \ldots, x_k)$ has a depth-2 circuit with a symmetric gate with fan-in N at the output and $\wedge$ gates with fan-in $k-1$ at the input level (figure 2). The claim is that f's k-party communication complexity is at most $k \log N$. (This observation is due to Razborov and Wigderson [**?**]). To see the claim,

DRAFT

first partition the $\wedge$ gates amongst the players. Each bit is not known to exactly one player, so the input bits of each $\wedge$ gate are known to at least one player; assign the gate to such a player with the lowest index. Players then broadcast how many of their gates output 1. Since this number has at most $\log N$ bits, the claim follows.

Our hope is to employ this connection with communication complexity in conjunction with Theorem 13.13 to obtain lower bounds on $\mathbf{ACC}^0$ circuits. For example, note that the function in Example **??** above cannot have $k < \log n/4$. However, this is not enough to obtain a lower bound on $\mathbf{ACC}^0$ circuits since we need to show that k is not polylogarithmic to employ Theorem 13.13. Thus a strengthening of the Babai Nisan Szegedy lowerbound to $\Omega(n/\mathrm{poly}(k))$ for say the CLIQUE function would close Frontier 2.

13.5.2 Connection to Linear Size Logarithmic Depth Circuits

Suppose that $f : \{0,1\}^n \times \{0,1\}^{\log n} \to \{0,1\}^n$ has bounded fan-in circuits of linear size and logarithmic depth. If $f(x,j,i)$ denotes the ith bit of $f(x,j)$, then Valiant's Lemma implies that $f(x,j,i)$ has a simultaneous 3-party protocol—that is, a protocol where all parties speak only once and write simultaneously on the blackboard (i.e., non-adaptively)—where,

- (x,j) player sends $n/\log\log n$ bits;
- (x,i) player sends n^ϵ bits; and
- (i,j) player sends $O(\log n)$ bits.

So, if we can show that a function does not have such a protocol, then we would have a lower bound for the function on linear size logarithmic depth circuits with bounded fan-in.

Conjecture: The function $f(x,j,i) = x_{j\oplus i}$, where $j \oplus i$ is the bitwise xor, is conjectured to be hard, i.e., f should not have a compact representation.

13.5.3 Connection to branching programs

The notion of multiparty communication complexity (at least the "number on the forehead" model discussed here) was invented by Chandra Furst and Lipton [**?**] for proving lowerbounds on branching programs, especially constant-width branching programs discussed in Section **??**

Web draft 2006-09-28 18:10
DRAFT

13.5.4 Karchmer-Wigderson communication games and depth lowerbounds

The result that PARITY is not in $\mathbf{AC}^0$ separates $\mathbf{NC}^1$ from $\mathbf{AC}^0$. The next step would be to separate $\mathbf{NC}^2$ from $\mathbf{NC}^1$. (Of course, ignoring for the moment the issue of separating $\mathbf{ACC}^0$ from $\mathbf{NC}^1$.) Karchmer and Wigderson [**?**] described how communication complexity can be used to prove lowerbounds on the minimum depth required to compute a function. They showed the following result about monotone circuits, which we will not prove this result.

THEOREM 13.17
Detecting whether a graph has a perfect matching is impossible with monotone circuits of depth $O(\log n)$

However, we do describe the basic *Karchmer-Wigderson* game used to prove the above result, since it is relevant for nonmonotone circuits as well. For a function $f:\{0,1\}^n \rightarrow \{0,1\}$ this game is defined as follows.

There are two players, **ZERO** *and* **ONE**. *Player* **ZERO** *receives an input x such that $f(x) = 0$ and Player* **ONE** *receives an input y such that $f(y) = 1$. They communicate bits to each other, until they can agree on an $i \in \{1, 2, \ldots, n\}$ such that $x_i \neq y_i$.*

The mechanism of communication is defined similarly as in Chapter 12; there is a protocol that the players agree on in advance before receiving the input. Note that the key difference from the scenario in Chapter 12 is that the final answer is not a single bit, and furthermore, the final answer is not unique (the number of acceptable answers is equal to the number of bits that x, y differ on). Sometimes this is described as *computing a relation*. The relation in this case consists of all triples (x, y, i) such that $f(x) = 0$, $f(y) = 1$ and $x_i \neq y_i$.

We define $C_{KW}(f)$ as the communication complexity of the above game; namely, the maximum over all $x \in f^{-1}(0), y \in f^{-1}(1)$ of the number of bits exchanged in computing an answer for x, y. The next theorem shows that this parameter has a suprising alternative characterization. It assumes that circuits don't have NOT gates and instead the NOT gates are pushed down to the inputs using De Morgan's law. (In other words, the inputs may be viewed as $x_1, x_2, \ldots, x_n, \overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$.) Furthermore, AND and OR gates have fanin 2. (None of these assumptions is crucial and affects the theorem only marginally.)

THEOREM 13.18 ([**?**])
$C_{KW}(f)$ is exactly the minimum depth among all circuits that compute f.

DRAFT

PROOF: First, we show that if there is a circuit $\mathcal{C}$ of depth K that computes f then $C_{KW}(f) \leq K$. Each player has a copy of $\mathcal{C}$, and evaluates this circuit on the input given to him. Of course, it ealuates to 0 for Player **ZERO** and to 1 for Player **ONE**. Suppose the top gate is an OR. Then at least one of the two incoming wires to this gate must be 1, and in the first round, Player **ONE** sends one bit communicating which of these wires it was. Note that this wire is 0 for Player **ZERO**. In the next round the players focus on the gate that produced the value on this wire. (If the top gate is an AND on the other hand, then in the first round Player **ZERO** speaks, conveying which of the two incoming wires was 0. This wire will be 1 for Player **ONE**.) This goes on and the players go deeper down the circuit, always maintaining the invariant that the current gate has value 1 for Player **ONE** and 0 for Player **ZERO**. Finally, after at most K steps they arrive at an input bit. According to the invariant being maintained, this bit must be 1 for Player **ONE** and 0 for Player **ZERO**. Thus they both know an index i that is a valid answer.

For the reverse direction, we have to show that if $C_{KW}(f) = K$ then there is a circuit of depth at most K that computes f. We prove a more general result. For any two disjoint nonempty subsets $A \subseteq f^{-1}(0)$ and $B \subseteq f^{-1}(1)$, let $C_{KW}(A, B)$ be the communication complexity of the Karchmer-Wigderson game when x always lies in A and y in B. We show that there is a circuit of depth $C_{KW}(A, B)$ that outputs 0 on every input from A and 1 on every input from B. Such a circuit is called a *distinguisher* for sets A, B. The proof is by induction on $K = C_{KW}(A, B)$. The base case $K = 0$ is trivial since this means the players do not have to communicate at all to agree on an answer, say i. Hence $x_i \neq y_i$ for all $x \in A, y \in B$, which implies that either (a) $x_i = 0$ for every $x \in A$ and $y_i = 0$ for every $y \in B$ or (b) $x_i = 1$ for every $x \in A$ and $y_i = 1$ for every $y \in B$. In case (a) we can use the depth 0 circuit x_i and in case (b) we can use the circuit $\overline{x_i}$ to distinguish A, B.

For the inductive step, suppose $C_{KW}(A, B) = K$, and at the first round Player **ZERO** speaks. Then A is the disjoint union of two sets A_0, A_1 where A_b is the set of inputs in A for which Player **ZERO** sends bit b. Then $C_{KW}(A_b, B) \leq K - 1$ for each b, and the inductive hypothesis gives a circuit C_b of depth at most $K - 1$ that distinguishes A_b, B. We claim that $C_0 \wedge C_1$ distinguishes A, B (note that it has depth at most K). The reason is that $C_0(y) = C_1(y) = 1$ for every $y \in B$ whereas for every $x \in A$, $C_0(x) \wedge C_1(x) = 0$ since if $x \in A_b$ then $C_b(x) = 0$. ■

Thus we have the following frontier.

Web draft 2006-09-28 18:10
DRAFT

Frontier 5: Show that some function f in **P** (or even **NEXP**!) has $C_{KW}(f) = \Omega(\log n \log\log n)$.

Karchmer, Raz, and Wigderson [**?**] describe a candidate function that may work. It uses the fact a function on k bits has a truth table of size 2^k, and that most functions on k bits are hard (e.g., require circuit size $\Omega(2^k/k)$, circuit depth $\Omega(k)$, etc.). They define the function by assuming that part of the n-bit input encodes a very hard function, and this hard function is applied to the remaining input in a "tree" fashion.

For any function $g : \{0,1\}^k \to \{0,1\}$ and $s \geq 1$ define $g^{\circ s} : \{0,1\}^{k^s} \to \{0,1\}$ as follows. If $s = 1$ then $g^{\circ s} = g$. Otherwise express the input $x \in \{0,1\}^{k^s}$ as $x_1x_2x_3\cdots x_k$ where each $x_i \in \{0,1\}^{k^{s-1}}$ and define

$$g^{\circ s}(x_1x_2\cdots x_k) = g(g^{\circ(s-1)}(x_1)g^{\circ(s-1)}(x_2)\cdots g^{\circ(s-1)}(x_k)).$$

Clearly, if g can be computed in depth d then $g^{\circ s}$ can be computed in depth sd. Furthermore, if one fails to see how one could reduce the depth for an arbitrary function.

Now we describe the KRW candidate function $f : \{0,1\}^n \to \{0,1\}$. Let $k = \lceil \log \frac{n}{2} \rceil$ and s be the largest integer such that $k^s \leq n/2$ (thus $s = \Theta(\frac{\log n}{\log\log n})$.) For any n-bit input x, let g_x be the function whose truth table is the first 2^k bits of x. Let $x|_2$ be the string of the last k^s bits of x. Then

$$f(x) = g_x^{\circ s}(x|_2).$$

According to our earlier intuition, when the first 2^k bits of x represent a really hard function —as they must for many choices of the input— then $g_x^{\circ s}(x|_2)$ should require depth $\Omega(sk) = \Omega(\frac{\log^2 n}{\log\log n})$. Of course, proving this seems difficult.

This type of complexity questions, whereby we are asking whether s instances of a problem are s times as hard as a single instance, are called *direct sum* questions. Similar questions have been studied in a variety of computational models, and sometimes counterintuitive results have been proven for them. One example is that by a counting argument there exists an $n \times n$ matrix A over $\{0,1\}$, such that the smallest circuit computing the linear function $v \mapsto Av$ for $v \in \{0,1\}^n$ is of size $\Omega(n^2)$. However, computing this function on n instances $v_1, \ldots, v_n$ can be done significantly faster than n^3 steps using fast matrix multiplication [**?**] (the current record is roughly $O(n^{2.38})$ [**?**]).

DRAFT

Chapter notes and history

Shannon defined circuit complexity, including monotone circuit complexity, in 1949. The topic was studied in Russia since the 1950s. (See Trakhtenbrot [?] for some references.) Savage [?] was the first to observe the close relationship between time required to decide a language on a TM and its circuit complexity, and to suggest circuit lowerbounds as a way to separate complexity classes. A burst of results in the 1980s, such as the separation of **P** from $\mathbf{AC}^0$ [?, ?] and Razborov's separation of monotone **NP** from monotone $\mathbf{P}/poly$ [?] raised hopes that a resolution of **P** versus **NP** might be near. These hopes were dashed by Razborov himself [?] when he showed that his method of approximations was unlikely to apply to nonmonotone circuits. Later Razborov and Rudich [?] formalized what they called *natural proofs* to show that all lines of attack considered up to that point were unlikely to work. (See Chapter 23.)

Our presentation in Sections 13.2 and 13.3 closely follows that in Boppana and Sipser's excellent survey of circuit complexity [?], which is still useful and current 15 years later. (It omits discussion of lowerbounds on algebraic circuits; see [?] for a recent result.)

Håstad's switching lemma [?] is a stronger form of results from[?, ?, ?]. The Razborov-Smolensky method of using approximator polynomials is from [?], strengthened in[?]. Valiant's observations about superlinear circuit lowerbounds are from a 1975 paper [?] and an unpublished manuscript—lack of progress on this basic problem gets more embarrassing by the day!.

The $4.5n - o(n)$ lowerbound on general circuits is from Lachish-Raz [?].

Exercises

§1 Suppose that f is computable by an **AC** 0 circuit C of depth d and size S. Prove that f is computable by an **AC** 0 circuit C' of size $10S$ and depth d that does not contain NOT gates but instead has n additional inputs that are negations of the original n inputs.

Hint: each gate in the old circuit gets a twin that computes its negation.

§2 Suppose that f is computable by an **AC** 0 circuit C of depth d and size S. Prove that f is computable by an **AC**0 C' circuit of size $(10S)^d$ and depth d where each gate has fanout 1.

DRAFT

§3 Prove that for $t > n/2$, $\binom{n}{t+k} \leq \binom{n}{t}\left(\frac{n}{n-t}\right)^k$. Use this to complete the proof of Lemma 13.2 (Section 13.1.2).

§4 Show that $\mathbf{ACC}^0 \subseteq \mathbf{NC}^1$.

§5 Identify reasons why the Razborov-Smolensky method does not work when the circuit has mod$m$ gates, where m is a composite number.

§6 Show that representing the OR of n variables $x_1, x_2, \ldots, x_n$ exactly with a polynomial over $GF(q)$ where q is prime requires degree exactly n.

§7 The Karchmer-Wigderson game can be used to prove *upperbounds*, and not just lowerbounds. Show using this game that PARITY and MAJORITY are in $\mathbf{NC}^1$.

§8 Show that if a language is computed by a polynomial-size branching program of width 5 then it is in $\mathbf{NC}^1$.

§9 Prove Valiant's Lemma (Lemma 13.14).

Hint: A directed acyclic graph can be be turned into a leveled graph, such that if $u \to v$ is an edge then u occurs at a lower level than v. Label this edge by looking at the numbers given to the levels of u, v and remove the edges corresponding to the least popular label.

DRAFT