Chapter 23

Why are circuit lowerbounds so difficult?

Why have we not been able to prove strong lower bounds for circuits? In 1994 Razborov and Rudich formalized the notion of a "natural mathematical proof," for a circuit lowerbound. They pointed out that current lowerbound arguments involve "natural" mathematical proofs, and show that obtaining strong lowerbound with such techniques would violate a widely believed cryptographic assumption (namely, that factoring integers requires time 2^{n^ϵ} for some fixed $\epsilon > 0$). Thus presumably we need to develop mathematical arguments that are not natural. This result may be viewed as a modern analogue of the Baker, Gill, Solovay result from the 1970s (see Chapter 4) that showed that diagonalization alone cannot resolve P versus NP and other questions.

Basically, a natural technique is one that proves a lowerbound for a random function and is "constructive." We formalize "constructive" later but first consider why lowerbound proofs may need to work for random functions.

23.1 Formal Complexity Measures

Let us imagine at a high level how one might approach the project of proving circuit lower bounds. For concreteness, focus on formulas, which are boolean circuits where gates have indegree 2 and outdegree 1. It is tempting to use some kind of induction. Suppose we have a function like the one in Figure 23.1 that we believe to be "complicated." Since the function computed at the output is "complicated", intuition says that at least one

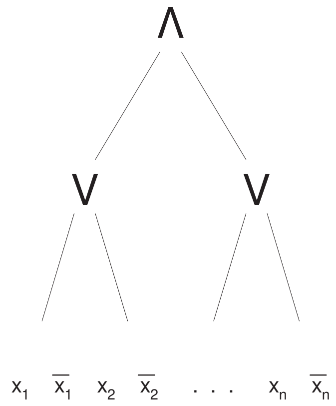

Figure 23.1: A formula for a hard function.

of the functions on the incoming edges to the output gate should also be "pretty complicated" (after all those two functions can be combined with a single gate to produce a "complicated" function). Now we try to formalize this intuition, and point out why one ends up proving a lowerbound on the formula complexity of random functions.

The most obvious way to formalize a "complicatedness" is as a function μ that maps every boolean function on $\{0,1\}^n$ to a nonnegative integer. (The input to μ is the truth table of the function.) We say that μ is a *formal complexity measure* if it satisfies the following properties: First, the measure is low for trivial functions: $\mu(x_i) \leq 1$ and $\mu(\bar{x}_i) \leq 1$ for all i. Second, we require that

- $\mu(f \wedge g) \leq \mu(f) + \mu(g)$ for all f, g; and
- $\mu(f \vee g) \leq \mu(f) + \mu(g)$ for all f, g.

For instance, the following function ρ is trivially a formal complexity measure

$$\rho(f) = 1 + \text{the smallest formula size for } f. \tag{23.1}$$

In fact, it is easy to prove the following by induction.

THEOREM 23.1
If μ is any formal complexity measure, then $\mu(f)$ is a lowerbound on the formula complexity of f.

Thus to formalize the inductive approach outlined earlier, it suffices to define a measure μ such that μ(CLIQUE) is high (say superpolynomial). For example, one could try "fraction of inputs for which the function agrees with the CLIQUE function" or some suitably modified version of this. In general, one imagines that defining a measure that lets us prove a good lowerbound for CLIQUE would involve some deep observation about the CLIQUE function. The next lemma seems to show, however, that even though all we care about is the CLIQUE function, our lowerbound necessarily must reason about random functions.

Lemma 23.2
Suppose μ is a formal complexity measure and there exists a function $f : \{0,1\}^n \to \{0,1\}$ such that $\mu(f) \geq c$ for some large number c. Then for at least $1/4$ of all functions $g : \{0,1\}^n \to \{0,1\}$ we must have $\mu(g) \geq c/4$.

Proof: Let $g : \{0,1\}^n \to \{0,1\}$ be any function. Write f as $f = h \oplus g$ where $h = f \oplus g$. So $f = (\bar{h} \wedge g) \vee (h \wedge \bar{g})$ and $\mu(f) \leq \mu(g) + \mu(\overline{g}) + \mu(h) + \mu(\overline{h})$.

Now suppose for contradiction's sake that $\{g : \mu(g) < c/4\}$ contains more than $3/4$ of all boolean functions on n-bit inputs. If we pick the above function g randomly, then $\overline{g}, h, \overline{h}$ are also random (though not independent). Using the trivial union bound we have $\Pr[\text{All of } h, \bar{h}, g, \bar{g} \text{ have } \mu < c/4] > 0$. Hence $\mu(f) < c$, which contradicts the assumption. Thus the lemma is proved. $\square$

In fact, the following stronger theorem holds:

Theorem 23.3
If $\mu(f) > c$ then for all $\epsilon > 0$ and for at least $1 - \epsilon$ of all functions g we have that,

$$\mu(g) \geq \Omega\left(\frac{c}{(n + \log(1/\epsilon))^2}\right).$$

The idea behind the proof of the theorem is to write f as the boolean combination of a small number of functions and then proceed similarly as in the proof of the lemma.

23.2 Natural Properties

Moving the above discussion forward, we think of a lowerbound proof as identifying some property of "hard" functions that is not shared by "easy" functions.

DEFINITION 23.1 *A* property Φ *is a map from boolean functions to* $\{0,1\}$. *A* P-natural property useful against *P/poly is a property* Φ *such that:*

1. $\Phi(f) = 1$ *for at least a* $1/2^n$ *fraction of all boolean functions on* n *bits (recall that there are* 2^{2^n} *functions on* n *bits);*

2. $\Phi(f) = 1$ *implies that* $f \notin$ *P/poly (or more concretely, that* f *has circuit complexity at least* $n^{\log n}$*, say); and*

3. Φ *is computable on* n*-bit functions in* $2^{O(n)}$ *time (i.e., polynomial in the length of the function's truth table).*

The term P-natural refers to requirement (3). The property is useful against P/*poly* because of requirement (2). (Note that this requirement also ensures that Φ is not trivial, since it must be 0 for functions in P/*poly*.) Requirement (1) corresponds to our above intuition that circuit lowerbounds should prove the hardness of a random function.

By suitably modifying (2) and (3) we can analogously define, for any complexity class $\mathcal{C}_1$ and circuit class $\mathcal{C}_2$, a $\mathcal{C}_1$-natural property that is useful against circuit class $\mathcal{C}_2$. We emphasize that when the property is computed, the input is the truth table of a function, whose size is 2^n. Thus a P-natural property is computed in time 2^{cn} for some constant $c > 1$ and a PSPACE-natural property is computed in space 2^{cn}.

EXAMPLE 23.1 The result that PARITY is not computable in AC^0 (Section 13.1) involved the following steps. (a) Show that every AC^0 circuit can be simplified by restricting at most $n - n^{\epsilon}$ input bits so that it then becomes a constant function. (b) Show that the PARITY function does not have this property.

Thus the natural property lurking in this proof is the following: $\Phi(f) = 1$ iff for every way of assigning values to at most $n - n^{\epsilon}$ input bits the function does not become a constant function. Clearly, if $\Phi(f) = 1$ then $f \notin \mathrm{AC}^0$, so f is useful against AC^0. Furthermore, Φ can be computed in $2^{O(n)}$ time — just enumerate all possible choices for the subsets of variables and all ways of setting them to 0/1. This running time is polynomial in the length of the truth-table, so Φ is P-natural. Finally, requirement (1) is also met since almost all boolean functions satisfy $\Phi(f) = 1$ (easy to check using a simple probability calculation; left as exercise).

Thinking further, we see that Φ is a AC^0-natural property that is useful against AC^0.

EXAMPLE 23.2 The lowerbound for ACC^0 circuits described in Section 13.2 is not natural *per se*. Razborov and Rudich show how to *naturalize* the proof, in other words change it —while retaining its essence—so that it does use a natural property. Recall that every boolean function on n bits can be represented by a multilinear polynomial over $GF(3)$. The space of all n-variate multilinear polynomials forms a vector space, whose dimension is $N = 2^n$. Then all multilinear polynomials in n variables of total degree less than $n/2$ form a subspace of dimension $N/2$ (this assumes n is even), and we denote this space by L. For a boolean function f let $\hat{f}$ be a multilinear polynomial over $GF(3)$ that represents f. Then define $\Phi(F) = 1$ iff the dimension of the space

$$\left\{\hat{f}l_1 + l_2 : l_1, l_2 \in L\right\}$$

is at least $3N/4$. It can be checked that Φ is 1 for the parity function, as well as for most random functions. Furthermore, rank computations can be done in NC^2 so it is NC^2-natural. The technique of Section 13.2 can be used to show that if $\Phi(f) = 1$ then $f \notin$ ACC$^0[3]$; thus Φ is useful against ACC$^0[3]$.

EXAMPLE 23.3 The lowerbound for monotone circuits in Section 13.3 does use constructive methods, but it is challenging to show that it applies to a random function since a random function is not monotone. Nobody has formulated a good definition of a random monotone function.

In the definition of natural proofs, requirement (3) is the most controversial in that there is no inherent reason why mathematical proofs should go hand in hand with efficient algorithms.

REMARK 23.1 "Constructive mathematics" was a movement within mathematics that rejected any proofs of existence that did not yield an algorithm for constructing the object. Today this viewpoint is considered quaint; nonconstructive proofs are integral to mathematics.

In our context, "constructive" has a stricter meaning, namely the proof has to yield a polynomial-time algorithm. Many proofs that would be "constructive" for a mathematician would be nonconstructive under our definition. Surprisingly, even with this stricter definition, proofs in combinatorial mathematics are usually constructive, and —as Razborov and Rudich are pointing out —the same is true of current circuit lowerbounds as well.

In a few cases, combinatorial results initially proved "nonconstructively" later turned out to have constructive proofs: a famous example is the Lovàsz Local Lemma (discovered in 1974; algorithmic version is in Beck [Bec91]). The same is true for several circuit lowerbounds—cf. the "naturalized" version of the Razborov-Smolensky lowerbound for $\mathrm{ACC}^0[q]$ mentioned earlier, and Raz's proof [Raz00] of the Babai-Nisan-Szegedy [BNS] lowerbound on multiparty communication complexity.

23.3 Limitations of Natural Proofs

The following theorem by Razborov and Rudich explains why we have not been able to use the same techniques to obtain an upper bound on $\mathrm{P}/poly$: constructing a P-natural property useful against $\mathrm{P}/poly$ violates widely believed cryptographic assumptions.

Theorem 23.4 (Razborov, Rudich [RR97])
Suppose a P-natural property Φ exists that is useful against $P/poly$. Then there are no strong pseudorandom function generators. In particular, factoring and discrete log have subexponential algorithms.

We recall that pseudorandom function generators were defined in Section 9.4.3. The definition used a distinguisher polynomial-time machine that is given oracle access to either a truly random function or a function from the pseudorandom family. The family is termed *pseudorandom* if the distinguisher cannot distinguish between the two oracles. Now we tailor that more general definition for our narrow purposes in this section. We allow the distinguisher $2^{O(n)}$ time and even allow it to examine the truth table of the function! (This is without loss of generality since in $2^{O(n)}$ time the distinguisher could construct the truth table using 2^n queries to the oracle.)

Definition 23.2 *A* pseudorandom function generator *is a function $f(k, x)$ computable in polynomial time where the input x has n bits and the "key" k has n^c bits, where $c > 1$ is a fixed constant. Denoting by F_n the function* *obtained by uniformly selecting $k \in \{0,1\}^{n^c}$ and setting F_n to $f(k,\cdot)$, we have the property that the function ensemble $F = \{F_n\}_{n=1}^{\infty}$ is "pseudorandom," namely, for each Turing machine M running in time $2^{O(n)}$, and for all sufficiently large n,*

$$|\Pr[M(F_n) = 1] - \Pr[M(H_n) = 1]| < \frac{1}{2^{n^2}},$$

where H_n is a random function on $\{0,1\}^n$.

We will denote $f(k,\cdot)$ by f_k.

Intuitively, the above definition says that if f is a pseudorandom function generator, then for a random k, the probability is high that f_k "looks like a random function" to all Turing machines running in time $2^{O(n)}$. Note that f_k cannot look random to machines that run in $2^{O(n^c)}$ time since they can just guess the key k. Thus restricting the running time to $2^{O(n)}$ (or to some other fixed exponential function such as $2^{O(n^2)}$) is crucial.

Recall that Section 9.4.3 described the Goldreich-Goldwasser-Micali construction of pseudorandom function generators $f(k,x)$ using a pseudorandom generator g that stretches n^c random bits to $2n^c$ pseudorandom (also see Figure 23.2): Let $g_0(k)$ and $g_1(k)$ denote, respectively, the first and last n^c bits of $g(k)$. Then the following function is a pseudorandom function generator, where $\mathrm{MSB}(x)$ refers to the first bit of a string x:

$$f(k,x) = \mathrm{MSB}(g_{x_n} \circ g_{x_{n-1}} \circ \cdots \circ g_{x_2} \circ g_{x_1}(k)).$$

The exercises in Chapter 9 explored the security of this construction as a function of the security parameter of g; basically, the two are essentially the same. By the Goldreich-Levin theorem of Section 9.3, a pseudorandom generator with such a high security parameter exists if a oneway permutation exists and some $\epsilon > 0$, such that every 2^{n^ϵ} time algorithm has inversion probability less than 2^{-n^ϵ}. The discrete log problem —a permutation— is conjectured to satisfy this property. As mentioned in Chapter 9, researchers believe that there is a small $\epsilon > 0$ such that the worst-case complexity of DISCRETE LOG is 2^{n^ϵ}, which by random self-reducibility also implies the hardness of the average case. (One can also obtain pseudorandom generators using FACTORING, versions of which are also believed to be just as hard as DISCRETE LOG.) If this belief is correct, then pseudorandom function generators exist as outlined above. (Exercise.)

Now we can prove the above theorem.

Proof:[Theorem 23.4] Suppose the property Φ exists, and f is a pseudorandom function generator. We show that a Turing machine can use Φ to

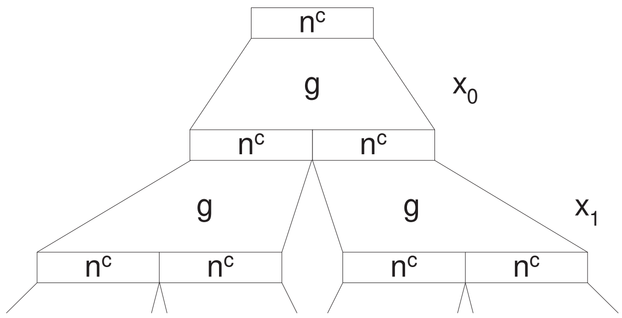

Figure 23.2: Constructing a pseudorandom function generator from a pseudorandom generator.

distinguish f_k from a random function. First note that $f_k \in$ P/*poly* for every k (just hardwire k into the circuit for f_k) so the contrapositive of property (2) implies that $\Phi(f_k) = 0$. In addition, property (1) implies that $\Pr_{H_n}[\Phi(H_n) = 1] \geq 1/2^n$. Hence,

$$\Pr_{H_n}[\Phi(H_n)] - \Pr_{k \in \{0,1\}^{n^c}}[\Phi(f_k)] \geq 1/2^n,$$

and thus Φ is a distinguisher against f. $\square$

23.4 My personal view

Discouraged by the Razborov-Rudich result, researchers (myself included) hardly ever work on circuit lowerbounds. Lately, I have begun to think this reaction was extreme. I still agree that a circuit lowerbound for say CLIQUE, if and when we prove it, will very likely apply to random functions as well. Thus the way to get around the Razborov-Rudich observation is to define properties that are not P-natural; in other words, are *nonconstructive.* I feel that this need not be such an insurmountable barrier since a host of mathematical results are nonconstructive.

Concretely, consider the question of separating NEXPTIME from ACC^0, one of the (admittedly not very ambitious) frontiers of circuit complexity outlined in Chapter 13. As observed there, NEXPTIME $\neq$ ACC^0 will follow if we can improve the Babai-Nisan-Szegedy lowerbound of $\Omega(n/2^k)$ for k-party communication complexity to $\Omega(n/\mathrm{poly}(k))$ for some function in NEXPTIME. One line of attack is to lowerbound the *discrepancy* of all large cylinder intersections in the truth table, as we saw in Raz's proof of the BNS lowerbound[1]. (In other words, the "unnatural" property we are defining is Φ where $\Phi(f) = 1$ iff f has high discrepancy and thus high multiparty communication complexity.) For a long time, I found this question intimidating because the problem of computing the discrepancy given the truth table of the function is coNP-hard (even for $k = 2$). This seemed to suggest that a proof that the discrepancy is high for an explicit function (which presumably will also show that it is high for random functions) must have a nonconstructive nature, and hence will be very difficult. Lately, I have begun to suspect this intuition.

A relevant example is Lovàsz's lowerbound of the chromatic number of the Kneser graph [Lov78]. Lowerbounding the chromatic number is coNP-complete in general. Lovàsz gives a topological proof (using the famous Borsuk-Ulam fixed point theorem) that determines the chromatic number of the Kneser graph exactly. From his proof one can indeed obtain an algorithm for solving chromatic number on all graphs([MZ02]) —but it runs in PSPACE for general graphs! So if this were a circuit lowerbound we could call it PSPACE-natural, and thus "nonconstructive." Nevertheless, Lovàsz's reasoning for the particular case of the Kneser graph is not overly complicated because the graph is highly symmetrical. This suggests we should not blindly trust the intuition that "nonconstructive $\equiv$ difficult."

I fervently hope that the next generation of researchers will view the Razborov-Rudich theorem as a guide rather than as a big obstacle!

Exercises

§1 Prove Theorem 23.3.

§2 Prove that a random function satisfies $\Phi(f) = 1$ with high probability, where Φ is the property defined in Example 23.1.

§3 Show that if the hardness assumption for discrete log is true, then pseudorandom function generators as defined in this chapter exist.

§4 Prove Wigderson's observation: P-natural properties cannot prove that DISCRETE LOG requires circuits of 2^{n^ϵ} size. (Hint: If DISCRETE LOG is hard on worst-case inputs then it is hard on most inputs, and then it can be used to construct pseudorandom functions.)

[1] Interestingly, Raz discovered this naturalization of the BNS proof after being briefly hopeful that the original BNS proof—which is not natural— may allow a way around the Razborov-Rudich result.

§5 (Razborov [Raz92]) A *submodular* complexity measure is a complexity measure that satisfies $\mu(f \vee g) + \mu(f \wedge g) \leq \mu(f) + \mu(g)$ for all functions f, g. Show that for every n-bit function f_n, such a measure satisfies $\mu(f_n) = O(n)$. (Hint: It suffices to prove this when f_n is a random function. Use induction on the number of variables, and the fact that both f_n and $\overline{f_n}$ are random functions.)

Chapter notes and history

The observation that circuit lowerbounds may unwittingly end up reasoning about random functions first appears in Razborov [Raz89]'s result about the limitations of the method of approximation.

We did not cover the full spectrum of ideas in the Razborov-Rudich paper [RR97], where it is observed that candidate pseudorandom function generators exist even in the class TC^0, which lies between ACC^0 and NC^1. Thus natural proofs will probably not allow us to separate even TC^0 from P.

Razborov's observation about submodular measures in Problem 5 is important because many existing approaches for formula complexity use submodular measures; thus they will fail to even prove superlinear lowerbounds.

In contrast with my limited optimism, Razborov himself expresses (in the introduction to [Raz03]) a view that the obstacle posed by the natural proofs observation is very serious. He observes that existing lowerbound approaches use weak theories of arithmetic such as Bounded Arithmetic. He conjectures that any circuit lowerbound attempt in such a logical system must be natural (and hence unlikely to work). But as I mentioned, several theorems even in discrete mathematics use reasoning (e.g., fixed point theorems like Borsuk-Ulam) that does not seem to be formalizable in Bounded Arithmetic. Thus is my reason for optimism.

However, somen other researchers are far more pessimistic: they fear that P versus NP may be independent of mathematics (say, of Zermelo-Fraenkel set theory). Razborov says that he has no intuition about this.