PRINCETON U. SP'02 COS 598B: ALGORITHMS AND COMPLEXITY

Lecture 12: Nonconstructive Proof Techniques: Natural Proofs

Lecturer: *Sanjeev Arora* Scribe:*Iannis Tourlakis*

Why is proving $\mathbf{P} \neq \mathbf{NP}$ so difficult? In 1994, Razborov and Rudich [3] put forth their idea of *Natural Proofs* to explain why. Essentially, Razborov and Rudich observed that all (circuit) lower bounds proved in the 1980s had a similar proof structure. They noticed that all such lowerbound proofs are both *constructive* and *generic* in a sense we will make precise below. They then proved, more or less, that if $\mathbf{P} \neq \mathbf{NP}$ then no such constructive, generic proof could be used to prove that $\mathbf{P} \neq \mathbf{NP}$. Before outlining their argument in detail, we need to talk about circuits and circuit lower bounds.

1 Circuits

A *circuit* is an acyclic combination of $\vee$, $\wedge$ and $\neg$ gates. Note that a circuit can only compute membership for bit strings of a fixed length. So to compute a language we employ a circuit *family* with a circuit for each input length. If the circuit for inputs of size n is bounded by some polynomial in n then we say that the language computed by the circuit family has *polynomial sized circuits.* The class of languages that have polynomial size circuits is denoted by $\mathbf{P}/poly$.

It is well known $\mathbf{P} \subseteq \mathbf{P}/poly$. Hence, if $\mathbf{NP}$ does not have polynomial size circuits, then $\mathbf{P} \neq \mathbf{NP}$. On the other hand, Karp and Lipton showed in 1980 that if $\mathbf{NP} \subseteq \mathbf{P}/poly$, then we get the unlikely result that $\mathbf{PH} = \Sigma_2^p$. So it would seem that $\mathbf{NP} \not\subseteq \mathbf{P}/poly$.

In fact it was thought in the 80s that rather than proving $\mathbf{P} \neq \mathbf{NP}$ directly, it may be easier to argue $\mathbf{NP} \not\subseteq \mathbf{P}/poly$. The idea was that reasoning about a concrete combinatorial structure like circuits would on the one hand perhaps give more insight on how to proceed while on the other would get around the limitations prescribed by certain oracle results from the 70s.

Moreover, researchers were encouraged by some early successes. Let $\mathbf{AC}^0$ be the class of languages accepted by circuits of depth $O(1)$, containing $\vee$ and $\wedge$ gates of unbounded fan-in, and where the number of gates is polynomial. Furst, Saxe and Sipser in 1981 [2], and Ajtai in 1982 [1] showed that PARITY $\notin \mathbf{AC}^0$.

Let us sketch a proof of this result. The idea is to use random restrictions. Suppose some $\mathbf{AC}^0$ circuit C computes PARITY. Randomly select some subset (of prescribed size) of the inputs and then set these inputs to 0-1 at random. The intuition is that the gates with large fan-in will likely be constant now; only the gates with small fan-in will not be fixed.

THEOREM 1 ([2, 1])
If the $\mathbf{AC}^0$ circuit has size n^c and depth d, then a random restriction of $n - n^{\frac{1}{3d}}$ inputs yields a constant function with high probability.

Since PARITY clearly does not have this property, it follows that PARITY $\notin \mathbf{AC}^0$.

An important question to ask after proving a lower bound is, does the lower bound apply to a random function? This is certainly true of the PARITY lower bound: a random function subject to a random restriction will almost certainly be non-constant.

But why does the FSS/Ajtai proof apply to a random function? Fix a function f and let R be a random function. Then we can write f as the parity of two random functions: $f = (f \oplus R) \oplus R$. So intuitively any "gate by gate" proof *must* reason about random functions. In the next section we give an example to make this intuition more concrete.

2 Lowerbounds for Formulas

A *formula* is a circuit in which each gate has fan-out 1, i.e., the circuit is a tree. To measure the complexity of a formula, let us define a "formal complexity measure" $\mu : \{\text{All } n\text{-bit boolean functions}\} \to \mathcal{R}^+$. The properties this function should have are as follows:

$$\begin{aligned} \mu(x_i), \mu(\bar{x}_i) &\leq 1, \\ \mu(f \wedge g) &\leq \mu(f) + \mu(g), \\ \mu(f \vee g) &\leq \mu(f) + \mu(g). \end{aligned}$$

Theorem 2 ([3])
If there exists a function f such that $\mu(f) > c$, then $\mu(R) > c/4$ for at least $1/4$ of all functions R.

Proof: Write $f = (f \oplus R) \oplus R$ where R is a random function. Now, $a \oplus b = (a \wedge \bar{b}) \vee (\bar{a} \wedge b)$. Hence, f is a boolean expression of size 4 of 4 random functions. Suppose now that with probability at least 3/4, $\mu(R) < c/4$. But then, there exists R such that $\mu(f \oplus R)$, $\mu(\overline{f \oplus R})$, $\mu(R)$, and $\mu(\bar{R})$ are each less than $c/4$. Hence $\mu(f) < c$, a contradiction. □

3 Natural Proofs

Definition 1 *A property Φ : $\{$All n-bit boolean functions$\} \to \{0, 1\}$ is* **P**-natural *and* useful against polynomial-size circuits *if,*

1. *$\Phi(f)$ is computable in $2^{O(n)}$ time (note that the input is the truth-table of f),*
2. *$\Phi(f) = 1$ implies that f does not have polynomial-size circuits,*
3. *$\Phi(f) = 1$ for at least $1/2^n$ fraction of all functions f.*

The key observation in [3] is that all current ideas in circuit lowerbounds have such properties at their core.

Example 1 In the proof that Parity $\notin \mathbf{AC}^0$, take $\Phi(f) = 1$ if no restriction on $n - n^{\epsilon}$ inputs can make f constant. (Note that this property is useful against $\mathbf{AC}^0$ circuits rather than polynomial-size circuits.)

Theorem 3 ([3])
If Discrete Log *and* Factoring *do not have $2^{n^{\epsilon}}$-time algorithms for some $\epsilon > 0$, then there is no* **P***-natural property useful against* **P***/poly.*

(Note that the best algorithms for DISCRETE LOG and FACTORING take time $2^{n^{1/3}}$.)
PROOF: We require the idea of a *pseudorandom function generator*. Fix $c > 2$. A pseudorandom function generator takes a key k of n^c bits and gives the truth-table of a function $g_k : \{0,1\}^n \to \{0,1\}$ such that the truth-table of g_k looks random to any polynomial-time algorithm (polynomial in the length of the truth-table, i.e., $2^{O(n)}$). Assuming that factoring n^c bit integers takes 2^{n^2} time, such objects exist.

Now suppose there exists a $\mathbf{P}$-natural property Φ useful against $\mathbf{P}/poly$. Then $\Phi(g_k) = 0$ since g_k is polynomial-time computable. On the other hand, $\Phi(R) = 1$ for a random function with probability at least $1/2^n$. But then, Φ distinguishes between g_k and a random function, contradicting the fact that g_k came from a pseudorandom function generator. $\square$

References

[1] M. Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[2] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, Apr. 1984.

[3] A. A. Razborov and S. Rudich. Natural proofs. In *Proceedings of the 26th Annual Symposium on the Theory of Computing*, pages 204–213, New York, May 1994. ACM Press.