CS 880: Pseudorandomness and Derandomization 4/17/2013

Lecture 23: Pseudorandomness from Worst-Case Hardness

Instructors: Holger Dell and Dieter van Melkebeek Scribe: Kevin Kowalski

DRAFT

In the previous lecture, we discussed how to construct pseudorandom generators for any particular class given the existence of a function f with high average-case hardness for that class, and gave an explicit construction for the class $\mathcal{C}_r$. In this lecture, we will finish the proof of its correctness, and also show that we can relax the hardness assumption we need from average-case to worst-case hardness, so that the function f only needs to be difficult to compute at one point, as opposed to many points.

1 Preliminaries

First, recall that average-case circuit compexity $H_f(m)$ for a function $f : \{0,1\}^m \to \{0,1\}$ is defined as the smallest s such that for some circuit C of size $\leq s$,

$$\Pr_{x \leftarrow U_m}[C(x) = f(x)] \geq \frac{1}{2} + \frac{1}{s}.$$

In general, we would need two parameters to describe average-case circuit complexity fully–one for the maximum size of the circuit, and one for the probability of agreement–but this single parameter definition suffices for our purposes.

Average-case circuit complexity can be contrasted with the worst-case circuit complexity $C_f(m)$, which is defined as the smallest s such that for some circuit C of size $\leq s$,

$$\forall x \in \{0,1\}^m [C(x) = f(x)].$$

In other words, a function with average-case complexity s requires a circuit of size s to agree on most of the possible inputs, while a function with worst-case complexity s requires a circuit of the same size to agree on all possible inputs. Thus, for any particular s, $H_f(m) = s$ is a much stronger statement than $C_f(m) = s$.

Also recall that the class of algorithms we are working with is

$$\mathcal{C}_r \doteq \{C \mid C \text{ is a Boolean circuit of size} \leq r \text{ on} \leq r \text{ inputs}\},$$

which is important because the existence of a pseudorandom generator with seed length $O(\log r)$ for this class would automatically imply the existence of pseudorandom generators for BPP–any polynomial-time computation on a particular input and r random bits can be represented as a circuit in $\mathcal{C}_r$. Whether or not we can achieve this seed length depends on the hardness of the underlying function f, though as we will see, the existence of weaker f would still imply nontrivial results regarding the power of BPP.

2 Construction of the Generator

Our construction $G_r^f : \{0,1\}^\ell \to \{0,1\}^r$ from the previous lecture was based on a function $f : \{0,1\}^m \to \{0,1\}$ combined with a $(m, \log r)$-design $S_1, S_2, \ldots, S_r \subseteq [\ell]$ where for all $1 \le i < j \le r$,

- $|S_i| = m$, and
- $|S_j \cap S_j| \le \log r$.

In particular, the i-th bit of the output of G_r^f is produced by applying f to the m bit positions of the input indicated by S_i.

Proof of pseudorandomness. The intuition behind this construction is that if f has high average-case hardness, then it is difficult for any circuit in $\mathcal{C}_r$ to predict its output on any given input. In fact, we could concatenate the seed with the output to produce a longer output, but the advantage gained by doing this is small because the required seed turns out to be much smaller than the output. The naïve idea of drawing the S_i from disjoint parts of the seed doesn't produce a sufficiently long output, but allowing them to overlap to a limited extent allows us to generate many S_i while preserving the unpredictability argument, as we will see in the proof of the following theorem.

Theorem 1. *If $H_f(m) > r^2$, then G_r^f is $\frac{1}{r}$-pseudorandom for $\mathcal{C}_r$.*

Proof. Suppose for the sake of contradiction that G_r^f is not $\frac{1}{r}$-pseudorandom for $\mathcal{C}_r$, i.e., there exists a distinguisher $D \in \mathcal{C}_r$ such that

$$|\Pr_{\rho \leftarrow U_m}[D(\rho) = 1] - \Pr_{\rho \leftarrow U_\ell}[D(G_r^f(\sigma)) = 1]| > \epsilon \doteq \frac{1}{r}.$$

Recall from last lecture that this implies that there exists a predictor $P \in \mathcal{C}_r$ and some i such that

$$\Pr_{\sigma \leftarrow U_\ell}[P(G_r^f(\sigma))[1, \ldots, i-1] = G_r^f(\sigma)[i]] > \frac{1}{2} + \frac{\epsilon}{r} = \frac{1}{2} + \frac{1}{r^2}.$$

By our construction, $G_r^f(\sigma)[i] = f(\sigma|_{S_i})$ and the other indices of $G_r^f(\sigma)$ are defined similarly, so we have that

$$\Pr_{\sigma \leftarrow U_\ell}[P(f(\sigma|_{S_1}), \ldots, f(\sigma|_{S_{i-1}})) = f(\sigma|_{S_i})] > \frac{1}{2} + \frac{1}{r^2},$$

which implies that for some setting of the bits of $\sigma|_{\overline{S_i}}$,

$$\Pr_{y \leftarrow U_m}[P(f(\sigma|_{S_1}), \ldots, f(\sigma|_{S_{i-1}})) = f(y)] > \frac{1}{2} + \frac{1}{r^2}.$$

Now, since $g(\sigma|_{S_1})$ and each other input to P can depend on at most $\log r$ bits of y, we can view each of these as a Boolean function on $\le \log r$ bits. Since any Boolean function on $\le \log r$ bits can be represented as a circuit of size $\le r$, $P \circ G_r^f[1, \ldots, i-1]$ can be represented as a circuit C of size at most

$$\underbrace{(r-1)}_{\text{max \# inputs}} \cdot \underbrace{r}_{\text{max input size}} + \underbrace{r}_{\text{max size of } P} = r^2.$$

This circuit C of size $\le r^2$ agrees with f on a fraction $> \frac{1}{2} + \frac{1}{r^2}$ of points in $\{0,1\}^m$, contradicting our assumption that $H_f(m) > r^2$. Thus, G_r^f is $\frac{1}{r}$-pseudorandom for $\mathcal{C}_r$, as desired. $\square$

Time complexity and seed length. All that remains is to show that G_r^f is efficiently computable and to characterize the best seed length we can achieve. In order to compute the i-th bit of the output, we need to construct S_i (which from te previous lecture takes polynomial time), and apply f to the indicated bits (which takes time $2^{O(m)}$). At first glance, the time it takes to compute f might seem to prevent us from being able to compute G_r^f efficiently, but if f is hard, the condition $H_f(m) > r^2$ still allows us to choose r to be much larger than m.

The polynomial-time construction of the $(m, \log r)$-design fom the previous lecture produces a seed length of $\ell = O(m^2)$ (or $\ell = m$ if $m \leq \log r$), so f takes time $2^{O(\ell)}$ to compute. In the context of full derandomization, this time complexity is perfectly acceptable because we will be cycling through all 2^ℓ seeds anyways–as long as the number of seeds is polynomial, each pseudorandom string only takes polynomial time to compute.

Implications for complexity theory. Assuming the existence of a hard f, we can use G_r^f to derandomize any probabilistic polynomial time computation. The seed length of G_r^f, however, depends on the hardness of f so Theorem 1 actually spawns a family of results that relate $H_f(m)$ to $\ell(r)$, and by extension to the time complexity of the derandomized procedure. Given the existence of harder functions f, we can construct generators with shorter relative seed length, so we can iterate through all possible seeds in a shorter amount of time. Some of the most significant implications of this nature are given in the theorem below.

Theorem 2. *Each of the following statements holds.*

- *If there exists* $f \in \mathrm{E}$ *such that* $H_f(m) \geq m^{\omega(1)}$*, then there exists a pseudorandom generator for* $\mathcal{C}_r$ *with seed length* $\ell(r) = r^{o(1)}$*, and* $\mathrm{BPP} \subseteq \mathrm{DTIME}(2^{n^{o(1)}})$.
- *If there exists* $f \in \mathrm{E}$ *such that* $H_f(m) \geq 2^{m^{\Omega(1)}}$*, then there exists a pseudorandom generator for* $\mathcal{C}_r$ *with seed length* $\ell(r) = (\log r)^{O(1)}$*, and* $\mathrm{BPP} \subseteq \mathrm{QP}$.
- *If there exists* $f \in \mathrm{E}$ *such that* $H_f(m) \geq 2^{\Omega(m)}$*, then there exists a pseudorandom generator for* $\mathcal{C}_r$ *with seed length* $\ell(r) = O(\log r)$*, and* $\mathrm{BPP} \subseteq \mathrm{P}$.

An important observation to make here is that all of the above statements are conditional–the pseudorandom generators only exist if some function with the necessary hardness exists. At present, no function in E is known to satisfy any of the above hardness conditions. Though our current construction does not give us any unconditional results about pseudorandom generators for $\mathcal{C}_r$, a similar approach will give us the unconditional existence of a pseudorandom generator for bounded-depth circuits in the next lecture.

3 Average-Case to Worst-Case Complexity

As it turns out, we can obtain a strengthening of Theorem 2 by relaxing each of the average-case complexity requirements on f to a worst-case complexity requirement. We will demonstrate this by constructing a transformation of $g : \{0,1\}^n \to \{0,1\}$ with high worst-case complexity $C_g(n)$ into a function $f : \{0,1\}^m \to \{0,1\}$ with high average-case complexity $H_f(m)$ that satisfies the following properties:

(1) $g \in \mathrm{E} \Rightarrow f \in \mathrm{E}$.

(2) $H_f(m) \leq s \Rightarrow C_g(n) \leq \text{poly}(n, s)$.

(3) $m = O(n)$.

The last condition is particularly important because the hardness of a function can be strongly dependent on the length of its input–we can trivially make a function require a larger circuit to compute by increasing its input length, but if the f we construct has a much higher input length than g, then we would need a much harder g to achieve a nontrivial complexity result with Theorem 2.

To construct this transformation, we will turn to error-correcting codes. The main idea is as follows: let $N \doteq 2^n$, and define the length N sequence $\chi_g \doteq (g(i))_{i \in \{0,1\}^n}$, i.e., χ_g is the output of g on every possible input. We will apply an error-correcting code to χ_g to obtain the similarly defined sequence χ_f of length $M = 2^m$. We would like to choose a code so that if there exists a small circuit C (which computes the function h) that agrees with f on most possible inputs, then the decoding of χ_h has a correponding small circuit that agrees with g on all inputs.

Now, note that conditions (1) and (3) are automatically met if the encoding function Enc of the code is computable in polynomial time. If g is computable in linear exponential time, then χ_f can be constructed in time $\text{poly}(2^{O(n)}) = 2^{O(n)}$, so f is also computable in linear exponential time. Similarly, since χ_f contains $2^{O(n)}$ elements, we must have that $m = O(n)$.

Thus, all we need is a polynomial-time computable code that satisfies condition (2). Intuitively, this condition will be satisfied if the code is locally decodable and can correct up to almost a fraction $\frac{1}{2}$ of errors. Local decodability allows us to estimate an element of χ_g from a small number of elements of χ_h, which limits the blowup in required circuit size to compute the decoding of h relative to h itself, and the ability to correct a large number of errors is necessary to ensure that χ_h is decoded to exactly χ_g. However, this isn't possible for an error- correcting code–the relative distance of a binary code is at most $\frac{1}{2}$, so it can correct errors in at most a fraction $\frac{1}{4}$ of all bit positions.

Instead, we can use list decoding to obtain all possible χ_g that encode to a sequence close to χ_h, and then there must exist some small circuit that chooses the correct one. Formally, this is captured in the following theorem:

Theorem 3. *There exists a function* $\text{Enc}_{n,s} : \{0,1\}^N \to \{0,1\}^{M(n,s)}$ *such that*

(A) $\text{Enc}_{n,s}$ *is computable in time* $\text{poly}(N, s)$.

(B) For any n, s*, there exists a list of circuits* $D_1, D_2, \ldots$ *each of size* $\text{poly}(n, s)$ *with oracle access to* χ_h *such that for every* $h : \{0,1\}^m \to \{0,1\}$ *and every* $g : \{0,1\}^n \to \{0,1\}$ *such that the relative Hamming distance between* $\text{Enc}_{n,s}(\chi_g)$ *and* χ_h *is at least* $\frac{1}{2} + \frac{1}{s}$*, there exists an* i *such that* D_i^f *computes* g.

Condition (A) here guarantees that $\text{Enc}_{n,s}$ satisfies conditions (1) and (3) from the beginning of the section, and condition (B) guarantees that $\text{Enc}_{n,s}$ satisfies condition (2).