U.C. Berkeley — Handout N10
CS294: Pseudorandomness and Combinatorial Constructions — September 29, 2005
Professor Luca Trevisan — Scribe: Varsha Dani

Notes for Lecture 10

Today we will study some conditions under which a very powerful pseudorandom generator can be shown to exist, and also some consequences of the existence of such a pseudorandom generator.

We will start by assuming the existence of a permutation $p : \{0,1\}^n \to \{0,1\}^n$ which is computable in poly(n) time and which, for some constant $\delta > 0$, is $(2^{\delta n}, 2^{-\delta n})$-one way. (This is an extremely strong assumption: we do not even know of any candidates for such permutations.)

Recall that as a consequence of the Goldreich-Levin theorem, we showed that given permutation p if the predicate $B(x,r) = \sum x_i r_i \mod 2$ is not (s,ε)-hard core for $x, r \mapsto p(x), r$ then p is not $(\frac{sn \log n}{\varepsilon^4}, \frac{\varepsilon}{4})$-one way.

Applying this with $s = 2^{\delta n/6}$ and $\varepsilon = 2^{-\delta n/6}$ we have in particular that if $x, r \mapsto \sum x_i r_i \mod 2$ is not (s,ε)-hard core for $x, r \mapsto p(x), r$ then p is not $(2^{5\delta n/6} n \log n, 2^{-2-\delta n/6})$-one way.

Thus, the existence of permutation p that is $(2^{\delta n}, 2^{-\delta n})$-one way implies that the predicate $B(x,r) = \sum x_i r_i \mod 2$ is $(2^{\delta n/6}, 2^{-\delta n/6})$-hard core for $x, r \mapsto p(x), r$.

Now recall the Blum-Micali-Yao theorem:

Theorem 1 *If $p : \{0,1\}^n \to \{0,1\}^n$ is a permutation, $B : \{0,1\}^n \to \{0,1\}$ is (s,ε)-hard core for p, and p and B are both computable by circuits of size t, then $G : \{0,1\}^n \to \{0,1\}^{n+k}$ defined by*

$$G(x) = B(x), B(p(x)), \ldots, B\left(p^{(k-1)}(x)\right), p^{(k)}(x)$$

is $(s - O(kt), k\varepsilon)$-pseudorandom.

Let $k = 2^{\delta n/12}$. Applying the Blum-Micali-Yao theorem to p and B above, we get a generator G that maps n bits to $2^{\delta n/12}$ bits, and is $(2^{\delta n/6} - 2^{\delta n/12}\text{poly}(n), 2^{-\delta n/12})$-pseudorandom. By change of variables $N = 2^{\delta n/12}$, we have a generator $G : \{0,1\}^{\log N/\delta} \to \{0,1\}^N$ which runs in $O(N\text{polylog}N)$ time and is $(N^2, 1/N)$-pseudorandom. Such a generator shall be called for the rest of this lecture an *ultimate* pseudorandom generator.

We have shown that the existence of $(2^{\Omega(n)}, 1/2^{\Omega(n)})$-one way permutations implies the existence of an ultimate pseudorandom generator.

What could we do with such a generator?

Definition 2 *A language L is in* **BPP** *if there is a probabilistic polynomial time algorithm A such that for all input strings x,*

$$\mathbf{Pr}\left[A(x,r) = L(x)\right] \geq \frac{3}{4}$$

where the probability is over the randomness r of the algorithm A.

Theorem 3 *If an ultimate pseudorandom generator exists then* $\mathbf{P} = \mathbf{BPP}$.

Proof: Suppose L is in **BPP**. Then there is an algorithm A that is realizable by a circuit of size $t(n) \leq n^{O(1)}$ and uses $r(n) \leq n^{O(1)}$ random bits on inputs of length n.

Let $x \in \{0,1\}^n$ be (arbitrarily) fixed. We need to show that with the help of the ultimate generator, we can decide whether or not $x \in L$ deterministically in time poly(n).

Let $N = \max\{\sqrt{t(n)}, r(n)\}$. Let G_N denote the ultimate pseudorandom generator that takes $O(\log N) = O(\log n)$ random bits and produces an N-bit pseudrandom string. G_N is $(N^2, 1/N)$ pseudrandom.

Now the mapping $r \mapsto A(x,r)$ is computable by a circuit of size $t(n) \leq N^2$ (the circuit for A with x hardwired in as input). Thus G_N is $1/N$-pseudorandom against $A(x,\cdot)$, so that

$$\left| \mathop{\mathbf{Pr}}_{r \sim \{0,1\}^{r(n)}} [A(x,r) = 1] - \mathop{\mathbf{Pr}}_{s \sim \{0,1\}^{O(\log n)}} [A(x, G_N(s)) = 1] \right| \leq \frac{1}{N}$$

Since L is in **BPP**, we have

$$\begin{aligned} x \in L &\Rightarrow \mathbf{Pr}[A(x,r) = 1] \geq 3/4 \\ x \notin L &\Rightarrow \mathbf{Pr}[A(x,r) = 1] \leq 1/4 \end{aligned}$$

Combining this and the pseudrandomness condition, we have

$$\begin{aligned} x \in L &\Rightarrow \mathbf{Pr}[A(x,G_N(s)) = 1] \geq 3/4 - 1/N > 1/2 \\ x \notin L &\Rightarrow \mathbf{Pr}[A(x,G_N(s)) = 1] \leq 1/4 + 1/N < 1/2 \end{aligned}$$

Thus we have reduced the number of random bits needed to only $O(\log n)$. But as there are only poly(n) bit strings of length $O(\log n)$, we can enumerate over all of the inputs to G_N. In other words, for each possible input s to G_N, we compute $A(x, G_N(s))$. This takes only poly(n) time. Since for a random s, $A(x, G_N(s))$ is the right answer with probability greater than 1/2, we output "$x \in L$" if $A(x, G_N(s))$ is 1 more often than it is 0 and we output "$x \notin L$" if $A(x, G_N(s))$ is 0 more often than it is 1. (Note that since the probability is strictly greater than 1/2, a tie cannot occur.) Thus we can deterministically decide whether $x \in L$ in poly$(|x|)$ time. It follows that $\mathbf{P} = \mathbf{BPP}$.
$\square$

The power of ultimate generators goes even further: they may also be used for derandomization in the setting where the problems are not decision problems, as we shall see in the next three results.

Theorem 4 *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function for which there is a probabilistic polynomial time algorithm A with the property that $\forall x \in \{0,1\}^*$*

$$\mathop{\mathbf{Pr}}_{r} [A(x,r) = f(x)] \geq 3/4$$

If ultimate pseudrandom generators exist then f is computable deterministically in polynomial time.

Proof: Let $t(n)$ denote the circuit size of A and $r(n)$ the number of random bits used by A on inputs of length n. Let $N = \Theta(\max\{\sqrt{t(n)}, r(n)\})$, and let $G_N : \{0,1\}^{O(\log N)} \to \{0,1\}^N$ be a $(N^2, 1/N)$-pseudorandom generator. Our first step is to show that feeding the output of G_N to A instead of a truly random string does not diminish its success probability by too much.

Let $x \in \{0,1\}^n$ be fixed. In this case, $f(x)$ is a fixed string of length at most $t(n)$. Let T be the circuit that on input r outputs 1 if $A(x,r) = f(x)$ and outputs 0 otherwise. Note that T has size $O(t(n)) < N^2$. By the pseudrandomness property, $|\mathbf{Pr}_r[T(r) = 1] - \mathbf{Pr}_s[T(G_N(s)) = 1]| \leq 1/N$. Therefore, $\mathbf{Pr}_s[A(x, G_N(s)) = f(x)] \geq 3/4 - 1/N > 1/2$. Thus we can use the output of G_N in A and still be guaranteed the right answer more than half the time.

So now to get a deterministic polytime algorithm, we enumerate over all inputs to G_N, as before, and we output the value that occurs the majority of times. $\square$

Theorem 5 *Let $f : \{0,1\}^* \to \mathbb{R}$ be a function for which there is a fully polynomial randomized approximation scheme,* i. e. *there is a probabilistic algorithm A such that for $\varepsilon > 0$, $\forall x \in \{0,1\}^*$*

$$\mathop{\mathbf{Pr}}_r [|A(x,\varepsilon,r) - f(x)| \leq \varepsilon f(x)] \geq 3/4$$

and A runs in poly$(|x|, 1/\varepsilon)$. If ultimate pseudrandom generators exist then there is a deterministic poly$(|x|, 1/\varepsilon)$ time algorithm to ε-approximate f.

Proof: Fix x of length n and $\varepsilon > 0$. Let $T(r)$ be the predicate that $|A(x,\varepsilon,r) - f(x)| \leq \varepsilon f(x)$. Choose N large enough so that N exceeds the number of random bits used by A on inputs of length n for desired accuracy parameter ε, and so that the size of the circuit needed to compute T is at most N^2. We can pick N large enough for this and yet have $N = \text{poly}(n)$ so that $\log N = O(\log n)$. Let G_N be the corresponding pseudrandom generator. We argue as usual that

$$\mathop{\mathbf{Pr}}_s [|A(x,\varepsilon,G_N(s)) - f(x)| \leq \varepsilon f(x)] \geq 3/4 - 1/N.$$

To derandomize the algorithm, we enumerate over all inputs to G_N, and the only thing that remains to be said is how to select an output value from the list of outputs that we get.

Say s is *bad* if $|A(x,\varepsilon,G_N(s)) - f(x)| > \varepsilon f(x)$. Then at most a $1/4 + 1/N$ fraction of all inputs to G_N are bad. Moreover, if we sort the list of values $\{A(x,\varepsilon,G_N(s))\}_s$ then the values corresponding to bad strings s must occur at the ends, *i. e.* if s is bad then either all values smaller than $A(x,\varepsilon,G_N(s))$ or all values larger than $A(x,\varepsilon,G_N(s))$ correspond to bad strings. In particular, the middle third of the list of values cannot contain any values corresponding to bad strings. Any value in here such as, say, the median of the list, will do as output of the algorithm. $\square$

Theorem 6 *Suppose we have a proof by the probabilistic method that an object with property $\mathcal{P}$ exists. (For simplicity let us say that by this we mean that under the uniform distribution $\mathcal{P}$ is true with probability $\geq 1/100$. In fact we can get by with weaker assumptions both on the distribution and on the lower bound on the probability). Suppose further that $\mathcal{P}$ is decidable in polynomial time. If ultimate pseudrandom generators exist then there is a deterministic polytime algorithm that generates objects with property $\mathcal{P}$.*

Note: The assumption that $\mathcal{P}$ is decidable in polynomial time is important, as we shall see in the proof. In particular, this derandomization technique does not work for Ramsey graphs.

Proof: Let $\mathcal{P}(r)$ denote the predicate that the object generated on random string r has property $\mathcal{P}$. Let s denote the size of the circuit to decide $\mathcal{P}$. We have assumed that this is polynomial in the size of the object to be generated. If N is chosen just large enough that G_N fools circuits of s, then N is also polynomial in the size of the object. Since for (uniformly) random r $\mathcal{P}(r)$ is true with probability at least $1/100$ by the pseudrandomness property, for strings output by the generator, $\mathcal{P}$ is true with probability at least $1/100 - 1/N > 0$. So if we generate an object corresponding to each output of G_N we are guaranteed that at least one of them has property $\mathcal{P}$. Since there are only polynomially many of them and we can test $\mathcal{P}$ in polytime, we can simply test each one. Thus we have a deterministic polytime algorithm to generate objects with property $\mathcal{P}$. $\square$

We will now take another look at the hypotheses that imply the existence of an ultimate pseudrandom generator. We have already shown their existence if $(2^{\Omega(n)}, 1/2^{\Omega n})$-one way permutations exist. However we would like to be able to prove their existence from weaker hypotheses.

Definition 7 *A function $f : \{0,1\}^n \to \{0,1\}$ is (s, ε)-hard if for all circuits C of size at most s, $\mathbf{Pr}_x[C(x) = f(x)] \leq 1/2 + \varepsilon$.*

Theorem 8 (Nisan, Wigderson) *Suppose there is a language L decidable in time $2^{O(n)}$ and $\exists \delta > 0$ such that L on inputs of length n is $(2^{\delta n}, 2^{-\delta n})$-hard. Then ultimate pseudrandom generators exist.*

Note that this is a strictly harder theorem: if p is a $(2^{\delta n}, 2^{-\delta n})$-one way permutation and B is $(2^{\delta n}, 2^{-\delta n})$-hard core for p then the language L defined by $L = \{y \mid B(p^{-1}(y)) = 1\}$ is $(2^{\delta n}, 2^{-\delta n})$-hard.

Nevertheless this is still a very strong requirement: the hypothesized language L must be hard to decide on *all* inputs. We state another theorem that further weakens the requirements.

Theorem 9 (Impagliazzo, Wigderson) *Suppose there is a language L decidable in time $2^{O(n)}$ and $\exists \delta > 0$ such that L on inputs of length n cannot be solved by circuits of size at most $2^{\delta n}$. Then there is a language L' decidable in time $2^{O(n)}$ and $\exists \delta' > 0$ such that L' on inputs of length n is $(2^{\delta' n}, 2^{-\delta' n})$-hard.*