CS225: Pseudorandomness Prof. Salil Vadhan

Lecture 18: PRGs from Average-Case Hardness

April 12, 2007 *Scribe: David Woodruff, David Troiano, and Denis Chebikin*

1 Noncryptographic Pseudorandom Generators Generators

Last time, Dan Gutfreund surveyed cryptographic pseudorandom generators, which numerous applications within and outside cryptography, including to derandomizing **BPP**. However, for derandomization, we can use generators with weaker properties. Specifically, we only need $G : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ such that:

1. G fools nonuniform distinguishers running in time n (as opposed to all probabilistic polynomial-time distinguishers).
2. G is computable in time $\text{poly}(n, 2^{\ell})$. In particular, *the PRG may take more time than the distinguishers it is trying to fool.*

Such a generator implies that every **BPP** algorithm can be derandomized in time $\text{poly}(n, 2^{\ell(n)})$.

The benefit of studying such generators is that we can hope to construct them under weaker assumptions than used cryptographic generators. In particular, a generator with the properties above no longer implies $\mathbf{P} \neq \mathbf{NP}$, much less the existence of one-way functions. Testing whether a string is an output of the generator is still an **NP** search problem, but even if we guess the seed properly, testing may take more time than the distinguishers are allowed. However, as you will show on Problem Set 6, such generators still imply nonuniform circuit lower bounds for exponential time, something that is still beyond the state of the art in complexity theory. Our goal in the next couple of lectures is to construct generators as above from assumptions that are as weak as possible.

2 Next-bit Unpredictability

In analyzing the pseudorandom generators that we construct, it will be useful to work with a reformulation of the pseudorandomness property, which says that, given a prefix of the output, it should be hard to predict the next bit.

For notational convenience, we deviate from our usual conventions use X to refer to an r.v. on $\{0,1\}^n$ which is part of an ensemble, and we use X_i for some $i \in [n] = \{1, \ldots, n\}$ to denote the ith bit of X. We have:

Definition 1 *Let $X = X^{(n)}$ be a random variable distributed on $\{0,1\}^n$. For functions $t : \mathbb{N} \to \mathbb{N}$ and $\varepsilon : \mathbb{N} \to [0,1]$, we say that X is (t, ε)* **next-bit unpredictable** *if for every nonuniform probabilistic algorithm P running in time $t(n)$ and every $i \in [n]$, we have:*

$$\Pr\left[P(X_1 X_2 \cdots X_{i-1}) = X_i\right] \leq \frac{1}{2} + \epsilon(n),$$

where the probability is taken over X and the coin tosses of P.

Note that the uniform distribution $X \equiv U_n$ is $(t, 0)$ next-bit unpredictable for every t. Intuitively, if X is pseudorandom, it must be next-bit unpredictable, as this is just one specific test one can perform on X. In fact the converse also holds, and this is the direction we will use.

Theorem 2 *Let X be a random variable distributed on $\{0,1\}^n$. If X is a (t, ε) pseudorandom, then X is $(t - O(1), \varepsilon)$ next-bit unpredictable. Conversely, if X is (t, ε) next-bit unpredictable, then it is $(t, n \cdot \varepsilon)$ pseudorandom.*

Proof: Here U denotes an r.v. uniformly distributed on $\{0,1\}^n$ and U_i denotes the i'th bit of U.

X pseudorandom $\Rightarrow$ X next-bit unpredictable. The proof is by reduction. Suppose for contradiction that X is not $(t - O(n), \varepsilon)$ next-bit unpredictable, so we have a predictor $P : \{0,1\}^{i-1} \to \{0,1\}$ that succeeds with probability at least $1/2 + \varepsilon$. We construct an algorithm $T : \{0,1\}^n \to \{0,1\}$ that distinguishes X from U_n as follows:

X next-bit unpredictable $\Rightarrow$ X pseudorandom. Also by reduction. Suppose X is not pseudorandom, so we have a nonuniform algorithm T running in time t s.t.

$$\Pr[T(X) = 1] - \Pr[T(U) = 1] > \epsilon,$$

where we have dropped the absolute values without loss of generality as in lecture 16.

We now use a hybrid argument. Define $H_i = X_1 \circ X_2 \circ \cdots \circ X_i \circ U_{i+1} \circ U_{i+2} \circ \cdots \circ U_n$. Then $H_n = X$ and $H_0 = U$. We have:

$$\sum_{i=1}^{n} (\Pr[T(H_i) = 1] - \Pr[T(H_{i-1}) = 1]) > \varepsilon,$$

since the sum telescopes. Thus, there must exist an i such that

$$\Pr[T(H_i) = 1] - \Pr[T(H_{i-1}) = 1] > \epsilon/n.$$

This says that T is more likely to output 1 when we put X_i in the i'th bit than when we put a random bit U_i. We can view U_i as being X_i with probability 1/2 and being $\overline{X_i}$ with probability 1/2. The only advantage T has must be coming from the latter case, because in the former case, the two distributions are identical. Formally,

$$\Pr[T(X_1 \cdots X_{i-1} X_i U_{i+1} \cdots U_n) = 1] + 1 - \Pr[T(X_1 \cdots X_{i-1} \overline{X_i} U_{i+1} \cdots U_n) = 1] = 2 \cdot (\Pr[T(H_i) = 1] - \Pr[T(H_{i-1}$$

This motivates the following next-bit predictor:

$P(x_1 x_2 \cdots x_{i-1})$:

1. Choose random bits $u_i, \ldots, u_n \overset{\mathrm{R}}{\leftarrow} \{0,1\}$.
2. Compute $b = T(x_1 \cdots x_{i-1} u_i \cdots u_n)$.
3. If $b = 1$, output u_i, otherwise output $\overline{u_i}$.

The intuition is that T is more likely to output 1 when $u_i = x_i$ than when $u_i = \overline{x_i}$. Formally, we have:

$$\begin{aligned}
&\Pr[P(X_1 \cdots X_{i-1}) = X_i] \\
&= \frac{1}{2} \cdot (\Pr[T(X_1 \cdots X_{i-1} U_i U_{i+1} \cdots U_n) = 1 | U_i = X_i] + \Pr[T(X_1 \cdots X_{i-1} U_i U_{i+1} \cdots U_n) = 0 | U_i \neq X_i]) \\
&= \frac{1}{2} \cdot \left(\Pr[T(X_1 \cdots X_{i-1} X_i U_{i+1} \cdots U_n) = 1] + 1 - \Pr[T(X_1 \cdots X_{i-1} \overline{X_i} U_{i+1} \cdots U_n) = 1]\right) \\
&> \frac{1}{2} + \frac{\varepsilon}{n}.
\end{aligned}$$

Note that as described P runs in time $t + O(n)$. If we use circuit-size as our measure of nonuniform time, we can reduce its running time to t as follows. First, we may nonuniformly fix the coin tosses $u_i, \ldots, u_n$ of P while preserving its advantage. Then all P does is run T on $x_1 \cdots x_{i-1}$ concatenated with some fixed bits and and either output what T does or its negation (depending on the fixed value of u_i). Fixing some input bits and negation can be done without increasing circuit size. Thus we contradict the next-bit unpredictability of X. ■

We note that an analogue of this result holds for uniform distinguishers and predictors, provided that we change the definition of next-bit predictor to involve a random choice of $i \overset{\mathrm{R}}{\leftarrow} [n]$ instead of a fixed value of i, and change the time bounds in the conclusions to be $t - O(n)$ rather than $t - O(1)$ and t (we can't do tricks like in the final paragraph of the proof). In contrast to the multiple-sample indistinguishability result from last time, this result does not need X to be efficiently samplable for the uniform version.

3 Average-Case Hardness

We now turn to the assumptions under which we can construct pseudorandom generators suitable for derandomization. Today, we will construct them from boolean functions that are hard on average for nonuniform algorithms (eg. circuits) but be computed in (uniform) exponential time.

Definition 3 *For functions $f : \mathbb{N} \to \mathbb{N}$ and $\alpha : \mathbb{N} \to [0,1]$, we say that a Boolean function $f_\ell : \{0,1\}^\ell \to \{0,1\}$ is (t, α)* average-case hard *if for all nonuniform probabilistic algorithm A running in time $t(\ell)$,*

$$\Pr[A(X) = f(X)] < 1 - \alpha(\ell).$$

for sufficiently large ℓ.

Observe that the negation of this definition is that there is nonuniform algorithm A running in time $t(\ell)$ such that $\Pr[A(X) = f(X)] \geq 1 - \alpha(\ell)$ for *infinitely many* in put lengths ℓ. Normally, when we say that a function is easy, we mean that there is an algorithm that does well on *all* input lengths ℓ. The above definition is thus stronger than the usual definition of 'easiness'. If we only have a function f that is not easy, this means that it is hard for infinitely values of ℓ, and thus we will only be able to use it to construct generators G_n that are pseudorandom for infinitely many values of n (which in turn will lead to a deterministic simulation of **BPP** that works correctly for infinitely many input lengths n).

Note that when $\alpha = 0$ and we consider *deterministic* algorithms A, then the above definition says that f is hard in the worst case. A definition of worst-case hardness for probabilistic algorithms will be given next time.

Today we consider $\alpha = \frac{1}{2} - \epsilon(\ell)$, where $\epsilon(\ell) = 1/t(\ell)$. That is, no efficient algorithm can compute f much better than random guessing. A typical setting of parameters we use is $t(\ell)$ somewhere in range from $\ell^{\omega(1)}$ (slightly superpolynomial) to $t(\ell) = 2^{\delta\ell}$ for a constant $\delta > 0$. (Note that every function is computable by a nonuniform algorithm running in time roughly 2^{ℓ}, so we cannot take $t(\ell)$ to be any larger.) We will also require f is computable in (uniform) time $2^{O(\ell)}$ so that our pseudorandom generator will be computable in time exponential in its seed length. The existence of such an average-case hard function does seem to be quite a strong assumption, but next time we will see how to relax it to a worst-case hardness assumption.

Now we show how to obtain a pseudorandom generator from average-case hardness.

Proposition 4 *If $f : \{0,1\}^{\ell} \to \{0,1\}$ is $(t, 1/2 - \varepsilon)$ average-case hard, then $G(x) = x \circ f(x)$ is a (t, ε) pseudorandom generator.*

Proof: This follows from the equivalence of pseudorandomness and next-bit unpredictability. Considering uniformly random seed X, we certainly can't predict the first ℓ bits with any advantage whatsoever, so the only hope is to predict $f(x)$ from x, but f is $(\frac{1}{2}-\epsilon)$-hard. A black-box application of Theorem 2 would lose a factor of $\ell + 1$ in the advantage ε, but we do not need to pay it here because the first ℓ bits are perfectly uniform. (Following the proof of Theorem 2, we would have $\Pr[T(H_i) = 1] - \Pr[T(H_{i-1}) = 1] = 0$ for $i = 1, \ldots, \ell$.) ■

Note that this generator includes its seed in its output. This is impossible for cryptographic pseudorandom generators, but is feasible (as shown above) when the generator can have more resources than the distinguishers it is trying to fool.

Of course, this generator is quite weak, stretching by only one bit. We would like to get many bits out. Here are two attempts:

- Define $G(x_1 \cdots x_k) = x_1 \cdots x_k f(x_1) \cdots f(x_k)$. This is a $(t, k\varepsilon)$ pseudorandom generator because we have k independent samples of a pseudorandom distribution so nonuniform computational indistinguishability is preserved. Note that already here we are relying on *nonuniform* indistinguishability, because the distribution $(U_\ell, f(U_\ell))$ is not samplable (in time that is feasible for the distinguishers).

- Use composition. For example, try to get two bits out using the same seed length by defining $G'(x) = G(G(x)_1 \cdots G(x)_\ell)G(x)_\ell$. This works for cryptographic pseudorandom generators, but not for the generators we are considering here. Why not?

4 The Nisan–Wigderson Generator

Our goal now is to show the following: Given $f \in \mathbf{E} = \mathbf{DTIME}(2^{O(\ell)})$ [1] that is $(\frac{1}{2} - \frac{1}{t(\ell)})$-hard for nonuniform time $t(\ell)$, construct an $(n, 1/n)$ pseudorandom generator $G : \{0,1\}^{O(\ell)} \to \{0,1\}^n$ with $n = t(\ell)^\delta$ for constant $\delta > 0$. (This is analogous to the parameters achieved for constructing pseudorandom generators from one-way permutations, as surveyed last time, but we are now using a weaker assumption.) Actually the generator we construct will have a slightly worse seed length than $O(\ell)$.

The idea is to apply f on *slightly dependent* inputs, i.e. x_i and x_j share very few bits. The sets of seed bits used for each output bit will be given by a *design*, as on Problem Set 2:

Definition 5 *$S_1, \cdots, S_m \subseteq [d]$ is an (ℓ, a)-design if*

1. *$\forall i, |S_i| = \ell$*
2. *$\forall i \neq j, |S_i \cap S_j| \leq a$*

We want lots of sets having small intersections over a small universe.

Lemma 6 *For every constant $\gamma > 0$ and every $\ell, m \in \mathbb{N}$, there exists an (ℓ, a)-design $S_1, \cdots, S_m \subseteq [d]$ with $d = O\left(\frac{\ell^2}{a}\right)$ and $a = \gamma \cdot \log m$. Such a design can be constructed deterministically in time* $\text{poly}(m, d)$.

This follows from the results on Problem Set 2. There you did the calculation for $\gamma = 1$ and $m = 2^{\epsilon \ell}$ for constant $\epsilon > 0$. For this case, the lemma gives $d = O\left(\frac{\ell^2}{\epsilon \ell}\right) = O(\ell)$, matching what you showed.

Given an (ℓ, a)-design $S_1, \cdots, S_m \subseteq [d]$ and average-case hard $f : \{0,1\}^\ell \to \{0,1\}$, define $G : \{0,1\}^d \to \{0,1\}^m$ as

$$G(x) = f(x|_{S_1}) f(x|_{S_2}) \cdots f(x|_{S_m})$$

where if x is a string in $\{0,1\}^d$ and $S \subseteq [d]$, $|S| = \ell$, $x|_S$ is the string of length ℓ obtained from x by selecting the bits indexed by S.

Theorem 7 *Suppose that $f : \{0,1\}^\ell \to \{0,1\}$ is $(\frac{1}{2} - \frac{1}{t})$-hard for nonuniform time t, and $S_1, \cdots, S_m \subseteq [d]$ is an (ℓ, a)-design with $m = t^{1/3}$ and $a = \frac{1}{3} \cdot \log t$, then $G : \{0,1\}^d \to \{0,1\}^m$ is an $(m, 1/m)$ pseudorandom generator.*

By construction of the design, the seed length is $d = O\left(\ell^2 / \log t\right)$, which is slightly worse than our original aim of $d = O(\ell)$, but we do achieve our aim for the important case when $t(\ell) = 2^{\Omega(\ell)}$. (We have seen earlier that this exponential hardness corresponds to what is needed to show $\mathbf{BPP} = \mathbf{P}$ case.)

Proof: Suppose G is not an (m, ε) pseudorandom generator for $\varepsilon = 1/m$. By Theorem 2, there is a nonuniform time m next-bit predictor P such that

$$\Pr[P(f(X|_{S_1}) f(X|_{S_2}) \cdots f(X|_{S_{i-1}})) = f(X|_{S_i})] > \frac{1}{2} + \frac{\varepsilon}{m}, \tag{1}$$

[1] $\mathbf{E}$ should be contrasted with the larger class $\mathbf{EXP} = \mathbf{DTIME}(2^{poly(\ell)})$

for some $i \in [m]$. From P, we construct A that computes f with probability $\frac{1}{2} + \frac{\epsilon}{m}$.

Let $Y = X|_{S_i}$. By averaging, we can fix all bits of $X|_{\bar{S}_i} = z$ such that the prediction probability is at least $\frac{1}{2} + \frac{\epsilon}{m}$ (over Y and the coin tosses of the predictor P). Define $f_j(y) = f(x|_{S_j})$ for $j \in \{1, \cdots, i-1\}$. (That is, $f_j(y)$ forms x by placing y in the positions in S_i and z in the others, and then applies f to $x|_{S_i}$). Then

$$\Pr_Y[P(f_1(Y) \cdots f_{i-1}(Y)) = f(Y)] > \frac{1}{2} + \frac{\epsilon}{m}.$$

Note that $f_j(y)$ depends only on $|S_i \cap S_j| \leq a$ bits of y. We cannot afford to compute f in the forward direction, but we can represent each f_j with a look-up table, which we can include in the advice to our nonuniform algorithm. Indeed, every function on a bits can be computed by a boolean circuit of size at most $a \cdot 2^a = \tilde{O}(t^{1/3})$. (In fact, size at most $O(2^a/a)$ suffices.)

Then, defining $A(y) = P(f_1(y) \cdots f_{i-1}(y))$, we have:

- $A(y)$ can be computed in nonuniform time time$(P) + m \cdot a \cdot 2^a = \tilde{O}(t^{2/3}) < t$.
- The advantage of A in computing f at least $\varepsilon/m = 1/t^{2/3} > 1/t$.

This contradicts the hardness of f. Thus, we conclude G is an $(m, \frac{1}{m})$ pseudorandom generator. ■

Corollary 8 *Suppose that* $\mathbf{E}$ *has a* $(t(\ell), 1/2 - 1/t(\ell))$ *average-case hard function* $f : \{0,1\}^\ell \to \{0,1\}$.

1. *If* $t(\ell) = 2^{\Omega(\ell)}$, *then* $\mathbf{BPP} = \mathbf{P}$.
2. *If* $t(\ell) = 2^{\ell^{\Omega(1)}}$, *then* $\mathbf{BPP} \subseteq \tilde{\mathbf{P}}$.
3. *If* $t(\ell) = \ell^{\omega(1)}$, *then* $\mathbf{BPP} \subseteq \mathbf{SUBEXP}$.

We make the following additional remarks:

1. This is a very general construction that works for any average-case hard function f. We only used $f \in \mathbf{E}$ to deduce G is computable in $\mathbf{E}$.
2. The reduction works for any nonuniform class of algorithms $\mathcal{C}$ where functions of logarithmically many bits can be computed efficiently.

Indeed, in the next section we will use the same construction to obtain an *unconditional* pseudorandom generator following constant-depth circuits.

5 Constant-depth circuits

We consider circuits with gates of type AND, OR, and NOT, and with unbounded fan-in (i.e. the number of incoming arrows) at gates of type AND and OR. The *depth* of such a circuit is the maximum length of a path from an input variable to the output. Typically negations are not counted in the depth (since they can all be moved to the bottom using De Morgan's laws).

Theorem 9 *For all constants d, the function $Par_\ell : \{0,1\}^\ell \to \{0,1\}$ defined by $Par_\ell(x_1, \ldots, x_\ell) = \bigoplus_{i=1}^{\ell} x_i$ is $(t, 1/2 - 1/t)$-average-case hard for depth d circuits of size $t = 2^{\Omega(\ell^{1/d})}$.*

Lemma 10 *Every function $g : \{0,1\}^a \to \{0,1\}$ can be computed by a depth 2 circuit of size 2^a.*

The following theorem by Nisan uses the two facts above.

Theorem 11 *For every constant d and every m, there exists a* $\mathrm{poly}(m)$*-time computable $(m, 1/m)$-pseudorandom generator $G_m : \{0,1\}^{\log^{O(d)} m} \to \{0,1\}^m$ fooling depth d circuits (of size m).*

Proof: Put $d' = d + 10$. Choose ℓ such that $m = 2^{\Omega(\ell^{1/d'})}$. Let G be the Nisan-Wigderson generator for $f = \mathrm{Par}_\ell$ and $t(\ell) = m^3 = 2^{\Omega(\ell^{1/d'})}$. Set $a = \log m$. Then the seed length of G is $O(\ell^2/a) < O(\ell^2) = O(\log^{d'} m)^2 = \log^{O(d')} m$.

We now follow the steps of the proof of Theorem 7 to go from an adversary T breaking the pseudorandomness of G to a circuit A calculating the parity function Par_ℓ.

If T has depth d, then the corresponding next-bit predictor P has depth $d_P = d + c$ for some small constant c. Recall that, in the proof of Theorem 7, we obtain A from P by $A(y) = P(f_1(y)f_2(y) \cdots f_{i-1}(y))$ for some $i \in \{1, \ldots, m\}$ and where each f_i depends on at most a bits of y. Now we observe that A can be computed by a small constant-depth circuit (if P can). Specifically, applying Lemma 10 to each f_i, the size of A is at most $O(m \cdot 2^a) = O(m^2)$ plus the size of P and the depth of A is at most $d_P + 2$. This contradicts the hardness of Par_ℓ (Theorem 9). ■

We define $\mathbf{BPAC_0}$ as the class of uniform languages decided by probabilistic constant depth circuits of polynomial size, where uniformity means that there is a polynomial time algorithm M such that $M(1^n) = C_n$ is the circuits of inputs of length n.

Corollary 12 $\mathbf{BPAC_0} \subseteq \tilde{\mathbf{P}}$.

With more work, this can be strengthened to actually put $\mathbf{BPAC_0}$ in $\widetilde{\mathbf{AC_0}}$. (The difficulty is that we use majority voting in the derandomization, but small constant-depth circuits cannot compute majority. However, they can compute an "approximate" majority, and this suffices.)

While $\mathbf{BPAC_0}$ may not seem a very natural class of randomized algorithms, on Problem Set 6, you will see how to use the above PRG to derandomize one of the actual randomized algorithms we have seen (namely the approximate DNF-counting algorithm).