CS 710: Complexity Theory 3/11/2010

Lecture 16: Pseudorandomness

Instructor: Dieter van Melkebeek Scribe: Michael Correll, Tom Watson

In the last lecture we introduced the notion of a pseudorandom generator (PRG), showed how PRGs can be used for derandomization, and developed a construction of a PRG that fools space-bounded computations. In particular, we developed a PRG with seed length $O(\log^2 n)$ that fools BPL computations. In the time-bounded setting, no nontrivial unconditional PRG constructions are known, but there are constructions that are known to work under certain reasonable complexity-theoretic hypotheses. Under a sufficiently strong (but still reasonable) hypothesis, this PRG allows us to show that BPP = P.

1 Pseudorandom Generators for Time-Bounded Computations

1.1 Distinguishability

Recall that a PRG takes a truly random seed of length $\ell(r)$ and produces a "pseudorandom" string of length r. To be useful, a PRG should be efficiently computable by a deterministic machine. A PRG is called *quick* if it can be computed in time $2^{O(\ell(r))}$, i.e. in time linear exponential to its seed length. We will show that if there exists a language in E with large average-case circuit complexity, then there exists a quick PRG with short seed length that fools time-bounded randomized computations. We will formalize the notion of average-case complexity in Section 2.1. In the next lecture, we will see how to use error-correcting codes to relax our hypothesis from the existence of an average-case hard language to the existence of a worst-case hard language.

The notion of "quickness" may not seem to be efficient enough, and indeed in the cryptographic setting PRGs are typically required to be computable in time polynomial in the seed length. Also, this may not be efficient enough if our goal is merely to reduce the amount of randomness needed by a computation. However, our present focus is full derandomization, achieved by trying all possible seeds and explicitly computing the probability that our algorithm accepts under the pseudorandom distribution. In this setting, we need $2^{\ell(r)}$ time just to look at all possible seeds, and so the factor $2^{O(\ell(r))}$ overhead in computing the PRG's output is just a polynomial overhead in time.

To guage the quality of our PRG construction, we will need measures of how powerful the computations we are trying to fool are allowed to be, and how well we fool these computations. These measures are formalized by the parameters r and ϵ in the following definition.

Definition 1. *An ϵ-PRG for circuits of size r is a family of functions $(G_r)_{r\in\mathbb{N}}$ where $G_r : \{0,1\}^{\ell(r)} \to \{0,1\}^r$ such that for all circuits C that take r inputs and are of size at most r,*

$$\left| Pr_{\sigma\in\{0,1\}^{\ell(r)}}\left[C(G_r(\sigma)) = 1\right] - Pr_{\rho\in\{0,1\}^r}\left[C(\rho) = 1\right] \right| < \epsilon$$

where σ and ρ are chosen uniformly at random.

I.e., that every circuit of size at most r will have trouble distinguishing whether its input was sampled from the uniform distribution or from the pseudorandom distribution, since both distributions are "close" as described by some bound ϵ.

There are a few questions about the above definition that present themselves, viz.

(1) Why do we require that our PRG fool circuits when we're really interested in fooling uniform computations? Since BPTIME(t) computations can be mimicked by circuits of size polynomial in t we will also be able to use such a PRG to fool the uniform computations. We want our PRG to succeed in fooling the computations on all but finitely many inputs, and this is easily captured in the nonuniform setting by constructing a different circuit for each input where the input is hard-wired and the random bits are left as inputs to the circuit. Also, it turns out that our arguments critically use the nonuniformity of the circuits. There are also results that start from a uniform hardness assumption, but those results aren't as strong.

(2) Why do we only require that our PRG fool circuits of linear size? Using the same parameter r for the size of the circuit and its number of inputs will keep the arguments cleaner, and there's no harm in allowing the circuit to take more random bits than it needs. Mimicking a uniform computation with a circuit may yield a circuit that's larger than the number of random bits it needs, but the computation won't be affected by allowing the PRG to provide more random bits. This size requirement is not so stringent as it appears; we can achieve added complexity by simply not using all r input bits and using the savings in circuit size.

Recall from the last lecture that a PRG can be used for full derandomization by trying all possible seeds and explicity computing the probability of acceptance of the algorithm under the pseudorandom distribution. The running time becomes the time to run the PRG on a given seed plus the time to run a simulation of the algorithm, times $2^{\ell(r)}$ seeds. Thus if we can get a quick PRG with $O(\log r)$ seed length, then this full derandomization runs in polynomial time, implying that BPP = P. Our ultimate goal is to show that if a sufficiently hard function exists, then such a PRG exists.

1.2 Predictability

When provided with a truly random seed, a PRG produces an output according to some distribution. Since the seed length $\ell(r)$ is ideally much smaller than the output length r, it follows that the pseudorandom distribution can have mass on at most $2^{\ell(r)}$ out of the 2^r strings of length r and will thus be, in some sense, far from the uniform distribution. Thus given sufficient power and complexity we can *always* distinguish our pseudorandom distribution from the uniform distribution. This is not an issue for us, however; we only want the pseudorandom distribution to be *computationally indistinguishable* from the uniform distribution. The circuits in Definition 1 can be viewed as statistical tests, and we only require that our pseudorandom distribution "pass" certain tests, namely those computable by relatively small circuits.

Our first step will be to show that we can restrict our class of statistical tests even further. We will be interested in circuits that attempt to *predict* the ith bit of a pseudorandom string given the first $i-1$ bits. No predictor exists for the uniform distribution; all circuits succeed in predicting the next bit of the sample with probability exactly $1/2$. In principle, an advantage in predicting the next bit can be gained by the fact that the input is sampled from a pseudorandom distribution; however, intuitively it seems like a lot of computation would be required to do this prediction. We leave it as an exercise to show that a circuit that succeeds with probability at least $1/2+\epsilon$ in predicting the ith bit from the first $i-1$ bits of a sample from a pseudorandom distribution yields a circuit of essentially the same size that can distinguish between the pseudorandom distribution and the uniform distribution by at least an ϵ amount in the sense of Definition 1. Thus an indistinguishable distribution is also unpredictable. It is conceivable that distinguishing is a much easier task than predicting, but we will now show that, actually, unpredictable distributions are also indistinguishable from the uniform distribution in a certain sense. Thus we will be able to focus our efforts on constructing a PRG with an unpredictable output distribution.

Theorem 1. *Unpredictability $\implies$ Computational Indistinguishability*

We will show the implication by proving the contrapositive, i.e. constructing a circuit and then show that this entails the existence of a predictor. It turns out that this relationship is biconditional. The proof of the opposite direction is similar to the proof below, and is left as an exercise.

Lemma 1. *If there exists a circuit C of size at most r such that*

$$\Big| Pr_{\sigma \in \{0,1\}^{\ell(r)}}\big[C(G_r(\sigma)) = 1\big] - Pr_{\rho \in \{0,1\}^r}\big[C(\rho) = 1\big]\Big| \geq \epsilon$$

then there exists an $i \in \{1, \dots, r\}$ and a circuit P of size at most r such that

$$Pr_{\sigma \in \{0,1\}^{\ell(r)}}\big[P\big((G_r(\sigma))_1, \dots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\big] \geq \frac{1}{2} + \frac{\epsilon}{r}.$$

Proof. Using the distinguisher C, we would like to construct a predictor P. Our first task will be to determine which bit position i will be predicted by P. Consider the hybrid distributions D_i $(i = 0, \dots, r)$ where D_i consists of samples where the first i bits are chosen according to the output distribution of G_r and the remaining bits are chosen uniformly at random. Then D_0 is the uniform distribution on strings of length r, and D_r is the output distribution of G_r. Intuitively, seeing how the circuit C behaves on distributions D_i and D_{i-1} should give us some idea of how good C is at predicting the ith bit from the first $i-1$ bits of a pseudorandom sample, because these distributions are very similar, differing only in the ith component. We can argue this formally. Using the shorthand $Pr_{D_i}[C = 1]$ for the probability that C outputs 1 on a sample from distribution D_i, we have

$$\begin{aligned}
\epsilon &\leq \Big| Pr_{D_r}[C = 1] - Pr_{D_0}[C = 1]\Big| \\
&= \Big| \sum_{i=1}^{r} \big(Pr_{D_i}[C = 1] - Pr_{D_{i-1}}[C = 1]\big)\Big| \\
&\leq \sum_{i=1}^{r} \Big| Pr_{D_i}[C = 1] - Pr_{D_{i-1}}[C = 1]\Big|
\end{aligned}$$

and thus $|Pr_{D_i}[C = 1] - Pr_{D_{i-1}}[C = 1]| \geq \epsilon/r$ for some i. We will choose this index i for our predictor. Now we have that

$$Pr\big[C\big((G_r(\sigma))_1, \dots, (G_r(\sigma))_{i-1}, (G_r(\sigma))_i, \rho_{i+1}, \dots, \rho_r\big) = 1\big]$$

differs from

$$Pr\big[C\big((G_r(\sigma))_1, \dots, (G_r(\sigma))_{i-1}, \rho_i, \rho_{i+1}, \dots, \rho_r\big) = 1\big]$$

by at least ϵ/r, where the probabilities are taken over σ and $\rho_i, \rho_{i+1}, \dots, \rho_r$ chosen uniform) value being 1 than in the uniform distribution. The circuit C appears to be doing a good job of detecting when the ith bit of its input came from the pseudorandom distribution, but it is not quite a predictor yet. In particular, it still takes r input bits, whereas a predictor is only given the first $i-1$ bits of a sample. However, by an averaging argument, there must be some setting $\widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r$ to the inputs $\rho_{i+1}, \ldots, \rho_r$ such that

$$Pr\big[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, (G_r(\sigma))_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r\big) = 1\big]$$

differs from

$$Pr\big[C\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, \rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r\big) = 1\big]$$

by at least ϵ/r, where the probabilities are taken over σ and ρ_i chosen uniformly at random. We can hard-wire these inputs without affecting the circuit size. Note that we are critically using the fact that we are dealing with nonuniform circuits, and so we can handle each value of r separately.

Now for some bit b, our circuit is at least ϵ/r more likely to output b when provided with the first i bits of a sample from the pseudorandom distribution than when provided with the first $i-1$ bits plus a truly random bit. This suggests how to construct a *randomized predictor* P': given the first $i-1$ bits $\pi_1, \ldots, \pi_{i-1}$ of a sample from the pseudorandom distribution, flip a coin to determine ρ_i, evaluate $C(\pi_1, \ldots, \pi_{i-1}, \rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r)$, and if it evaluates to b, assume that our guess was correct and output ρ_i, and otherwise output $\overline{\rho_i}$. More formally,

$$P'(\pi_1, \ldots, \pi_{i-1}) = \rho_i \oplus C(\pi_1, \ldots, \pi_{i-1}, \rho_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r) \oplus b.$$

Claim 1. *$Pr\big[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, \rho_i\big) = (G_r(\sigma))_i\big] \geq \frac{1}{2} + \frac{\epsilon}{r}$ where the probability is over σ and ρ_i.*

Proof. Consider two cases:

(1) The output of C does not depends on ρ_i, i.e. C will always output 1 or 0 no matter what we guess for ρ_i. If this is the case then we have we will guess with accuracy 1/2.

(2) The output of C *does* depend on ρ_i. If the actual value x_i will produce an output of 1 on C then P' will always guess correctly (outputs 1 $\iff$ $x_i = 1$). If x_i's true value will make C output 0, then our predictor is always wrong.

In either case the contribution of the $|Pr\big[P'\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}, \rho_i\big) = (G_r(\sigma))_i\big]| - 1/2$ term will be greater than $\frac{\epsilon}{r}$. $\square$

This is exactly the behavior we want from our predictor, but P' still draws on one random bit. In order to create a deterministic P we can again take advantage of the fact that we're in the nonuniform setting and hard-wire ρ_i to some value $\widetilde{\rho}_i$, either 0 or 1, such that the circuit retains its advantage of ϵ/r in predicting the ith bit. This yields a predictor P where $P(\pi_1, \ldots, \pi_{i-1})$ is just

$$\widetilde{\rho}_i \oplus C(\pi_1, \ldots, \pi_{i-1}, \widetilde{\rho}_i, \widetilde{\rho}_{i+1}, \ldots, \widetilde{\rho}_r) \oplus b,$$

which can be expressed as a circuit of the same size as C (possibly with an additional NOT gate, which we assume doesn't increase the size of the circuit). This predictor satisfies

$$Pr\big[P\big((G_r(\sigma))_1, \ldots, (G_r(\sigma))_{i-1}\big) = (G_r(\sigma))_i\big] \geq \frac{1}{2} + \frac{\epsilon}{r},$$

as desired. $\square$