Computational Complexity Theory, Fall 2010 November 24

Lecture 21: Derandomization using PRG's

Lecturer: Kristoffer Arnsfelt Hansen Scribe: Kristoffer Arnsfelt Hansen

1 Cryptographic pseudo random generators

Last time we defined (nonuniform) cryptographic pseudo random generators.

Definition 1. A function $G : \{0,1\}^* \to \{0,1\}^*$ is a nonuniform pseudorandom generator if there is a *stretch function* $l : \mathbb{N} \to \mathbb{N}$, such $l(m) > m$ and $|G(x)| = l(|x|)$ for all x, satisfying the following properties.

- G is polynomial time computable.
- For every family $\{D_m\}$ of polynomial size Boolean circuits, we have

$$|\Pr[D_m(G(U_m)) = 1] - \Pr[D_m(U_{l(m)}) = 1]| \leq 1/m^k$$

  for any k and any sufficiently large m.

where U_m is chosen from $\{0,1\}^m$ uniformly at random and $U_{l(m)}$ is chosen from $\{0,1\}^{l(m)}$ uniformly at random.

We also say that G fools the circuit D_m with error at most $1/m^k$.

Such pseudo-random generators exists if and only if nonuniform one-way functions exists. We state one direction of this equivalence.

Theorem 2. *If nonuniform cryptographic one-way functions exists, then nonuniform cryptographic pseudo-random generators exists, with any polynomial strech function $l(m) = m^k$.*

Since $\mathbf{BPP} \subseteq \mathbf{P}/\text{poly}$ we are able to use cryptographic PRG's for the purpose of *derandomization*, i.e. completely eliminating the use of randomness of an algorithm.

Theorem 3. *If nonuniform one-way functions exists, then*

$$\mathbf{BPP} \subseteq \bigcap_{\epsilon>0} \text{DTIME}(2^{n^\epsilon}) \ .$$

Proof. Let A be a probabilistic algorithm running in time n^a deciding a decision problem with error probabilisty $\leq 1/3$. Let b be arbitrary. By our assumption we have a cryptographic PRG with stretch $l(m) = m^{ab}$. From this we construct the derandomized algorithm A' as follows. On input x of length $n = |x|$.

1. Iterate over all seeds $r \in \{0,1\}^{n^{1/b}}$ of length $n^{1/b}$.
2. For each r, run A on input x using $G(r)$ as the random bits.
3. Output majority vote of the $2^{n^{1/b}}$ runs of A.

Consider an input length n, and let $m = n^{1/b}$. Equivalent to A is a Boolean circuit of size $O(n^{2a}) = m^{O(1)}$. Thus for sufficiently large m, the generator will fool these circuits with error at most $1/m < 1/10$. If this is the case, then since $1/3 + 1/10 < 1/2$ the majority vote is the correct answer, and thus A' decides the same decision problem as A does.

The time complexity is $2^m m^{O(1)} = O(2^{n^\epsilon})$, whenever $1/b < \epsilon$. Since this holds for all ϵ, the result follows. □

Some remark are in order, motivating our definition of a notion of PRG's tailor-made for derandomization purposes:

- Since we enumerate over all seeds anyway, we can allow the generator to use *linear exponential time* to produce the pseudorandom bits.
- We only need to fool circuits of a *fixed* polynomial size, and the generator can depend on this size.
- We only need to fool circuits with a fixed constant error, and the generator can depend on this error.

2 Pseudo random generators for derandomization

WARNING: We are now going to reverse the meaning of the function l. It is no longer going to be a strech function, but rather going to specify the length of the seed. Also below, when we consider generators G, we really consider a *family* of generators, parametirized by m, or alternatively, restrictions of a function $G : \{0,1\}^* \to \{0,1\}^*$, as done above.

Definition 4. A function $G : \{0,1\}^{l(m)} \to \{0,1\}^m$ is an ϵ-PRG for circuits of size S if for all Boolean circuits of size S on m inputs we have

$$|\Pr[C(U_m) = 1] - \Pr[C(G(U_{l(m)})) = 1]| \leq \epsilon \ .$$

We say that G is a *quick* PRG if the function can be computed in time $2^{O(l(m))}$, i.e. linear exponential in the seed length.

We can use these for derandomization in a similar way as above. Namely, suppose we wish to derandomize a probabilistic algorithm A deciding a language L in time $T(n)$ with error at most $1/3$. Then there is an equivalent family of circuits computing L of size $m = O(T(n)^2)$. Suppose now we have a quick $1/10$-PRG G for circuits of size m on m inputs with seed length $l(m)$. We can then run over all seeds of G, run A using the pseudo-random bits instead of the random bits, and give the majority vote as answer. This gives a correct procedure for deciding L by our assumptions, similarly to the case of cryptographic PRG's. The running time becomes $2^{O(l(m))}T(n)$.

Typical scenarios one consider is the following:

"Low end" When we can get a PRG with $l(m) = m^\epsilon$, for any $\epsilon > 0$. In this case we can obtain the result that $\mathbf{BPP} \subseteq \bigcap_{\epsilon>0} \mathrm{DTIME}(2^{n^\epsilon})$.

"High end" When we can get a PRG with $l(m) = O(\log m)$. In this case we can obtain $\mathbf{BPP} = P$.

Intermidiate scenarios are of course possible. For instance, one could imagine getting a PRG with $l(m) = (\log m)^{O(1)}$. In this case we can obtain $\mathbf{BPP} \subseteq \mathrm{DTIME}(2^{(\log n)^{O(1)}})$.

3 PRG's and Circuit lower bounds

Theorem 5. *Suppose that there exists a quick* $1/10$*-PRG* $G : \{0,1\}^l \to \{0,1\}^m$ *for circuits of size* $S(l)$*. Then there is there is a language* L *in* $\mathbf{E}$ *that requires circuits of size* $S(n)$*.*

Proof. Define $L = \{G(y)_{[1:|y|+1]} \mid y \in \{0,1\}^*\}$, to be the language consisting of the prefixes of the output of the generator of length 1 larger than the length of the seed. Suppose now for contradiction that C is a circuit of size $S(n)$ that computes L on input length n. This means we have

$$\Pr[C(G(U_{n-1})) = 1] = 1$$

Now since G fools C we need to have

$$|\Pr[C(U_n) = 1] - \Pr[C(G(U_{n-1})) = 1]| \leq 1/10$$

However, since L can only contain a $1/2$ fraction of the strings of length n (since those are obtained as output of the generator on strings of length $n-1$), we have

$$\Pr[C(U_n) = 1] \leq 1/2$$

leading to contradiction. □

Thus existence of PRG's imply circuit lower bounds for languages in $\mathbf{E}$. A major result of complexity theory is that the reverse is also true: Circuit lower bounds for languages in $\mathbf{E}$ imply existence of PRG's. Before moving towards this result, we will consider PRG's based on a stronger assumption, more precisely an average case hard assumption.

4 Hardness based PRG's

We will regularly switch between languages $L \subseteq \{0,1\}^*$ and their characteristic (Boolean) function $f : \{0,1\}^* \to \{0,1\}$.

Define the *hardness* of a language L at input length n $H_L(n)$ by $H_L(n) = \max S$ such that no circuit of size S on n inputs can compute L on a $\frac{1}{2} + \frac{1}{S}$ fraction of the inputs of length n (equivalently $\Pr_{x \in \{0,1\}^n}[C(x) = f(x)] < \frac{1}{2} + \frac{1}{S}$). We may also write $H_f(n) = H_L(n)$.

We will next consider the following generator $G_1 : \{0,1\}^n \to \{0,1\}^{n+1}$ defined by $G_1(y) = (y, f(y))$, stretching n bits to $n+1$ bits. Although getting one additional pseudo-random bit is not very impressive, it is instrutive to see the proof that it is in fact pseudorandom.

To show that a generator is pseudorandom we will use an alternative characterization of pseudorandomness by the notion of *unpredictability*.

Consider a function $G : \{0,1\}^l \to \{0,1\}^m$. For some $i \in \{1, \ldots, m\}$ we say that a circuit P on $i-1$ inputs is an ϵ-*next bit* predictor for G if $\Pr[P(w_1 \ldots w_{i-1}) = w_i] > \frac{1}{2} + \epsilon$, where the probability is over $w = G(y)$ for $y \in \{0,1\}^l$ chosen uniformly at random.

Clearly if there exists an ϵ-next bit predictor for G of size S then G is not an ϵ-PRG for circuits of size S, since the predictor can be used to build a distinguishing circuit D, namely we let $D(w) = 1$ if and only if $P(w_1 \ldots w_{i-1}) = w_i$. Then $\Pr[D(G(U_l)) = 1] > \frac{1}{2} + \epsilon$, whereas $\Pr[D(U_m) = 1] = \frac{1}{2}$.

The next lemma shows that unpredictability also implies pseudorandomness.

Lemma 6. *Let $G : \{0,1\}^{l(m)} \to \{0,1\}^m$ be any function, and let C be a circuit of size S such that*

$$|\Pr[C(U_m) = 1] - \Pr[C(G(U_{l(m)})) = 1]| > \epsilon$$

Then there is an $\frac{\epsilon}{m}$-next bit predictor for G.

We will give the proof of this lemma in the next lecture. For now we will show how to use this lemma to show that the function G_1 is a pseudo random generator. Suppose that $G_1 : \{0,1\}^n \to \{0,1\}^{n+1}$ is not an $\frac{n+1}{H_f(n)}$-PRG for circuits of size $H_f(n)$. Then for some i we have a $\frac{1}{H_f(n)}$-next bit predictor P for G, for the ith bit of $w = G(y)$. Consider first $i \in \{1, \ldots, n\}$. Then we have $\Pr[P(w_1 \ldots w_{i-1}) = w_i] = \frac{1}{2}$ for *any* function P. Thus we must have $i = m + 1$, and $\Pr[P(w_1 \ldots w_m) = w_{m+1}] > \frac{1}{2} + \frac{1}{H_f(n)}$. Plugging in the definition of G_1 we have $\Pr[P(y) = f(y)] > \frac{1}{2} + \frac{1}{H_f(n)}$, which is a contradiction with the definition of H_f.

We can now also present a small preview of the idea behind the Nisan-Wigderson PRG we will present in the next lecture. Basically the idea is to evaluate a hard function f on a subset of the input bits that are in some sense "almost independent". This way one will be able to stretch the seed exponentially under sufficiently strong assumptions.