# CS 744: Pseudorandomness Generators

Lecture 15: Pseudorandom Generators

February 8, 2015

**Definition 1.** *A deterministic polynomial-time algorithm $G : \Sigma^* \times \Sigma^* \to \Sigma^*$ is called a* pseudorandom generator *if there is a stretch function $\ell : \mathbb{N} \to \mathbb{N}$ such that the two probability families $\{G_n\}_{n\in\mathbb{N}}$ and $\{U_{\ell(n)}\}_{n\in\mathbb{N}}$ are computationally indistinguishable, where $G_n$ is the output of $G$ on a seed uniformly selected from $\Sigma^n$.*

Thus pseudorandom generators are a specific form of a derandomization technique, characterized by three factors [1] - 1. efficiency of construction, 2. the stretch function describing the relationship of the pseudorandom output length to the length of the seed, and 3. the output should be pseudorandom. The stretch function in particular, distinguishes the approach of derandomization via pseudorandom generators from the basic approaches we considered in the initial part of the course.

If we have a pseudorandom generator, then we can use its output on seeds of appropriate length instead of the uniform distribution as the source of randomness for a probabilistic polynomial-time algorithm.

## 1 Stretching by one bit

First, we give the construction of a pseudorandom generator which can stretch its input by one bit. The construction relies on the existence of one-way permutations and hard-core predicates for these permutations.

**Definition 2.** *A predicate $B : \Sigma^n \to \Sigma$ is an $(S, \varepsilon)$-hardcore for a permutation $f : \Sigma^n \to \Sigma^n$ if for every circuit $C$ of size at most $S$,*

$$U_n[x \in \Sigma^n : C(f(x)) = B(x)] \leq \frac{1}{2} + \varepsilon.$$

*A polynomial-computable family $B = \langle B_n \rangle_{n\in\mathbb{N}}$ of predicates is a* hard-core *for a family of polynomial-time computable permutations $f = \langle f_n \rangle_{n\in\mathbb{N}}$ if for all polynomials $p$ and $q$ and all large enough $n$, $B_n$ is a $(p(n), \frac{1}{q_n})$ hardcore for $f_n$.*

We have seen that the Goldreich-Levin theorem constructs such hard-core predicates for one-way permutations.

**Theorem 3.** *[Goldreich, Levin] Let $f : \Sigma^n \to \Sigma^n$ be a one-way permutation and $g : \Sigma^n \times \Sigma^n \to \Sigma^n \times \Sigma^n$ be defined by $g(x, r) = (f(x), r)$. Then if $B(x, r)$ is the inner product of $x$ and $r$ modulo 2, then $g$ is a one-way permutation and $B$ a hard-core predicate for $g$.*

Now we see how to use a hard-core predicate for a one-way permutation to construct a pseudo-random generator.

**Theorem 4.** *[Blum, Micali, Yao] If $B$ is an $(S, \epsilon)$ hard-core predicate for a one-way permutation $f : \Sigma^n \to \Sigma^n$, then $F : \Sigma^n \to \Sigma^{n+1}$ defined by $F(x) = p(x)^\frown B(x)$ is $(S, \varepsilon)$-pseudorandom.*

We will prove a slightly weaker theorem where we conclude that the output is $(S/2, \varepsilon)$ pseudo-random.

*Proof.* Let $D$ be a circuit of size $S$ such that

$$U_n[x \in \Sigma^n \ : \ D(p(x), B(x)) = 1] - U_{n+1}[x \in \Sigma^n, b \in \Sigma \ : \ D(p(x), b) = 1] \geq \varepsilon. \tag{1}$$

It suffices to construct a circuit $C$ of size $2S$ which satisfies

$$U_n[x \in \Sigma^n \ : \ C(x) = B(x)] \ \geq \ 1/2 + \varepsilon. \tag{2}$$

By assumption, $D$ has a greater tendency to output 1 when $B(x)$ is used in place of a pure random bit.

Consider the following algorithm which predicts $B(x)$. Run $D(p(x), 0)$ and $D(p(x), 1)$. There are four cases depending on the outputs.

If both outputs are the same, then $D(p(x), .)$ does not depend on the random bit. Since we have no useful information to exploit, we output $B(x)$ by flipping a fair coin.

If $D(p(x), 0)$ is 1 and $D(p(x), 1)$ is 0, then $B(x)$ is likelier to be 0. So we guess 0 as the value of $B(x)$.

If $D(p(x), 0)$ is 1 and $D(p(x), 1)$ is 1, then we output $B(x)$ is 1.

Clearly this algorithm can be implemented by a circuit using two copies of $D$ (and a constant number of gates to implement the case analysis on the outputs, but we will ignore this). Observe that when the outputs are different, $C(x)$ matches the output $D(p(x), 1)$ in both cases. To show that inequality (2) holds, we can equivalently show

$$U_n[x \in \Sigma^n \ : \ C(x) = B(x)] - U[x \in \Sigma^n \ : \ C(x) \neq B(x)] \geq 2\varepsilon. \tag{3}$$

Let $E_{00}$ denote the event that both $D(p(x), 0) = 00$ and $D(p(x), 1) = 0$, and $E_{11}$ the probabilty that both are 1. Observe that when the outputs are different, always $C(x) = D(p(x), 1)$. The case

when $D(p(x),1) = B(x)$ (hence $C$ guesses correctly) is denoted t $C_1$ (for "input 1 is correct") and when it does not as $W_1$ (for "input 1 is wrong").

To handle the first term in inequality (1), observe that

(1) when both $D(p(x),0)$ and $D(p(x),1)$ are 1, then $D(p(x),B(x)) = 1$ and

(2) when both are different and $D(p(x),1) = B(x)$, then $D(p(x),B(x)) = 1$ always — *i.e.* When $D(p(x),1) = 1 = B(x)$, then $D(p(x),B(x)) = D(p(x),1) = 1$ and when $D(p(x),1) = 0 = B(x)$, then $D(p(x),B(x)) = D(p(x),0) = 1$.

Thus, we can write the first term as (!)

$$P(E_{00}) + P(C_1). \tag{4}$$

The second term is $P(E_{11}) + 1/2P(C_1) + 1/2P(W_1)$, and substituting in inequality (1), we obtain $1/2(Pr(C_1) - Pr(W_1)) \geq \varepsilon$, which is equivalent to (3). (Why?) □

We will be interested in *optimal* pseudorandom generators. Optimal pseudorandom generators are those which can derandomize BPP. In the following three lectures, we will cover the conditional construction of a pseudorandom generator capable of derandomizing BPP.

## 2 The Nisan-Wigderson Pseudorandom Generator

The motivating question we consider in the course is the derandomization of BPP algorithms, showing that randomness gives no additional power over deterministic polynomial-time computation. Even though an unconditional full derandomization is the ideal, we may obtain a partial answer to the question by seeing whether a widely believed conjecture about complexity classes leads to the construction of a pseudorandom generator.

### 2.1 Sketch of construction

Nisan and Wigderson construct a pseudorandom generator under the assumption that certain functions in E which are ill-predicted on average by sub-exponential size circuits. We first define what sense we measure the hardness of a function for a circuit.

**Definition 5.** *A function $f : \Sigma^n \to \Sigma$ is $(s,\varepsilon)$-hard if for all circuits of size $s$,*

$$U_{2^n \otimes 2^r}[r \mid C(x;r) = f(x)] < \frac{1}{2} + \varepsilon.$$

Note that this is a stronger assumption than worst-case complexity, which only says that there are instances $x \in \Sigma^n$ on which $C(x)$ and $f(x)$ disagree. This assumption says that *on average* over the inputs and the random choices made, $C(x) \neq f(x)$ with very high probability.

This was later improved by Impagliazzo and Wigderson, who construct optimal pseudorandom generators under the weaker assumption that there ate languges in $E$ which cannot be solved *in*

*the worst case* by subexponential size circuits (they may be doing well on average, but there are some worst case instances which they are unable to decide.)

We now prove the Nisan-Wigderson theorem. [1]

**Theorem 6.** *(Nisan, Wigderson) Suppose there is a language $L \in E$ and a positive $\delta$ such that $L \cap \Sigma^n$ is (size:$2^{\delta n}$, tolerance:$2^{-\delta n}$)-hard, then optimal pseudorandom generators exist.*

*Proof.* First, □

## References

[1] O. Goldreich. *Foundations of Cryptography*, volume 1. Springer, 2001.

[1] The material in this section is adapted from Luca Trevisan's notes, Chapter 11, and Salil Vadhan's book.