Complexity Theory II	Course Instructor: V. Arvind
Lecture 1 : Aug 2, 2006	
Lecturer: V. Arvind	*Scribe: Kazim Bhojani*

1 Outline of the course

In this course we will discuss following topics

- Hardness vs. Randomness
  - Pseudorandom generators
  - Error correcting codes
- Extractors
- Expander graphs
  - Reingold's Logspace algorithm for undirected connectivity
  - New proof of PCP theorem

Topics discussed in the first part are taken from a survey by Nisan and Wigderson, *Hardness vs. Randomness.*

2 Notion of Pseudorandomness and constructing PRG's

In this lecture we shall discuss two notions of randomness as formulated by Blum-Micali and Yao during 80's ,show that the two definitions are equivalent and then go on to construct a PRG using one way functions.

2.1 Two notions of Pseudorandomness

If a *resource bounded* observer cannot distinguish between Ideal random source and another deterministic source A , then A can be used in place of ideal source and it is as good as an ideal source from the observer's point of view.

Such an A will be called a *Pseudorandom Generator*.

Blum and Micali for the first time in 1980 formalized the notion of randomness and gave the following definition of Pseudorandomness.

Definition 1. *Blum-Micali definition(Next bit prediction test)*

Let $G = \{G_n : \{0,1\}^{l(n)} \to \{0,1\}^n\}$ *be such that* $l(n) << n$ *, that is ,* G *takes a seed of length* $l(n)$ *and generates a string of length* n *.*

G *is called a* Pseudorandom Generator *for a class* C *(of algorithms) , if for every polynomial* $p(n)$ *and each* i *and for all the algorithms* $A \in C$ *the following holds*

$$\left| Prob_{x \in \{0,1,\}^{l(n)}}[A(y_1 y_2 \dots y_{i-1}) = y_i] - \frac{1}{2} \right| < \frac{1}{p(n)}$$

where $x \in \{0,1\}^{l(n)}$ *and* $G_n(x) = y_1 y_2 \dots y_n$.

Yao in 1982 formulated a different definition of Pseudorandomness and showed that the two definitions are equivalent

Definition 2. *Yao's Definition(Distinguisher test)*

A function $G : l(n) \to n$ *is a PRG* [1] *for a class* C *of algorithms if for every polynomial* $p(n)$ *and each* i *and for all the algorithms* $A \in C$ *the following holds*

$$\left| Prob_{y \in \{0,1,\}^n}[A(y) = 1] - Prob_{x \in \{0,1,\}^{l(n)}}[A(G(x)) = 1] \right| < \frac{1}{p(n)}$$

Theorem 3 (Yao)**.** *The two definitions of Pseudorandomness given above are equivalent*

2.2 Constructing PRG's

First we begin with the definition of *One way functions.*

Definition 4. *One way functions*

A function $f = \{f_n : \{0,1\}^n \to \{0,1\}^m\}$ *is a* 1-way function *if for every polynomial* $p(n)$ *and every circuit* C *of size* $p(n)$ *the following holds*

$$Prob_{x \in \{0,1\}^n}[C(f(x)) \notin f^{-1}(x)] \geq \frac{1}{p(n)}$$

and f *should be computable in polynomial time.*

[1] Henceforth we will use the abbreviation PRG for Pseudorandom generator(s)

The following are two examples of functions are that are believed to be 1-way and no *fast* algorithms are known for inverting these functions

- Function f that computes product of two n bit primes
-
$$f : (\mathbb{Z}_{p-1}, +) \rightarrow (\mathbb{Z}_p^*, *)$$
$$x \mapsto a^x$$

Now we state a theorem due to Yao which relate 1-way functions with PRGs.

Theorem 5. *(Yao's theorem).*
If there is a 1-way function f then for every $\epsilon > 0$ there is a PRG

$$G : n^\epsilon \rightarrow n$$

such that G runs in polynomial time and is secure against all polynomial size circuits.

But how do we get hold of G , given a 1-way function ?
Here is a rough procedure how we can use 1-way function to get hold of a PRG

- f is a 1-way function.
- Get hold of g which is 1-way with amplified hardness
- Then compute $x, g(x), g(g(x)), \ldots, g^{(n)}(x)$.
- Extract one bit from each of the above strings and this $x_0x_1\ldots x_n$ is the output of the PRG

One immediate corollary of the Yao's theorem is the following result

Corollary 6. *If 1-way functions exist then $BPP \subseteq DTIME(2^{n^\epsilon})$ for some $\epsilon > 0$.*

Also the converse of Yao's of theorem is also true.

Theorem 7. *(Converse of Yao's Theorem).*
If there is a PRG
$$G : n^\epsilon \rightarrow n$$
then 1-way functions exist.

2.3 Nisan-Wigderson design

First we begin with definition of quick PRG.

Definition 8. *(quick PRG).*

$G : l(n) \rightarrow n$ is called a quick PRG if it is computable in time $2^{O(l(n))}$ (deterministic) and for every circuit C of size n^2 the following holds

$$\left| Prob_{y \in \{0,1,\}^n}[C(y) = 1] - Prob_{x \in \{0,1,\}^{l(n)}}[C(G(x)) = 1] \right| < \frac{1}{n}$$

Lemma 9. *If a quick PRG $G : l(n) \rightarrow n$ exists then for every time constructible function $t(n)$*

$$BPTIME(t(n)) \subseteq DTIME(2^{O(l(t(n)))}$$

Proof. Let $L \in BPTIME(t(n))$ and M be a machine which accepts language L and x be an input instance.

Let $|x| = t(n)$, we can assume that the r random bits used in the computation are given with input.

Look at the tableau of computation it has $t(n)$ configurations.

Now $G_{t(n)} : O(l(t(n))) \rightarrow O(t(n))$, so we can cycle over all string s of length $l(t(n))$ and use $G(s)$ as a random string in computation so the whole simulation can be done in $DTIME(2^{O(l(t(n)))}$

Hence proved.

□

Now we define the notion of hardness of a boolean function.

Definition 10. *A boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$ is called (ϵ, S)-hard , if for all circuits of size S the following holds,*

$$\left| Prob_{x \in \{0,1,\}^{l(n)}}[C(x) = f(x)] - \frac{1}{2} \right| < \epsilon$$

Also a function f is said to have hardness $H(f) = m$, where is m is the largest number such that f is $(m, \frac{1}{m})$-hard.

We can now use hard functions to build PRGs using the Nisan-Wigderson design

2.3.1 The (n,l,m,k) design

- Let $S_1, S_2, \ldots, S_n \subseteq \{1, 2, \ldots, l\}$ such that $|S_i| = m$
- $i \neq j \implies |S_i \cup S_j| \leq k$.
- Let $f : \{0,1\}^m \to \{0,1\}$ be the given hard function.
- Let $x = x_1 x_2 \ldots x_l$ be the seed where l is the seed length.
- Now project x onto each coordinate of S_i to get a string of length m and apply f to that string to obtain bit y_i , that is $y_i = f(x|S_i)$ for $i = 1, 2, \ldots, n$.
- $y = y_1 y_2 \ldots y_n$ is the output of the Pseudorandom generator.

So we have a function $G_f : l \to n$.

Theorem 11. *If $f : \{0,1\}^m \to \{0,1\}$ is a boolean function with hardness n^2 and G_f is built from a $(n, l, m, \log n)$ design , then G_f is a quick PRG.*

Proof. Suppose C is a circuit of size n^2 that distinguishes G_f's output from random source , that is ,

$$Prob_{y \in \{0,1,\}^n}[C(y) = 1] - Prob_{x \in \{0,1,\}^{l(n)}}[C(G_f(x)) = 1] > \frac{1}{n}$$

Let E_1 be the uniform distribution and E_n be the distribution of $G(x)$, now we advance a hybrid argument to obtain a contradiction.

Define the intermediate distributions $E_2, \ldots, E_{(n-1)}$ as follows ,

- E_0 has distribution $r_1, r_2, \ldots, r_n$ where r_i's are true random bits.

  $\vdots$

- E_i has distribution $u_1, \ldots, u_i, r_{i+1}, \ldots, r_n$, where $u_1, \ldots, u_i$ are first i bits of output of pseudorandom generator G.

  $\vdots$

- E_n has distribution $u_1, u_2, \ldots, u_n$ where u_i's are output of G.

So we have ,

$$Prob_{y\in E_0}[C(y)=1]-Prob_{y\in E_n}[C(y)=1]>\frac{1}{n}$$

$$\implies \sum_{i=0}^{n-1}\left[Prob_{y\in E_i}[C(y)=1]-Prob_{y\in E_{i+1}}[C(y)=1]\right]>\frac{1}{n}$$

$$\implies \exists\ i: Prob_{y\in E_i}[C(y)=1]-Prob_{y\in E_{i+1}}[C(y)=1]>\frac{1}{n^2}$$

So now we have a randomized algorithm $D(y_1,y_2,\ldots,y_n)$ where input is first i bits of G's output and outputs y_{i+1} with probability $\geq \frac{1}{2}+\frac{1}{n^2}$.

- Pick $r_{i+1},\ldots,r_n$ ideal random bits.
- Let $C(y_1,\ldots,y_i,r_{i+1},\ldots,r_n)=b$
- if $b=1$ then predict $y_{i+1}=r_{i+1}$ else $y_{i+1}=\overline{r_{i+1}}$

Claim

$$Prob_{x,r_{i+1},\ldots,r_n}[D(y_1,\ldots,y_i)=y_{i+1}]\geq\frac{1}{2}+\frac{1}{n^2}$$

This shows that functions obtained using Nisan-Wigderson design are Pseudorandom generators if the function used is a hard function , assuming the claim.

□

Now we prove the claim stated above.

Proof of Claim.

$$\begin{aligned}
\text{Required probability} &= Prob[D(y_1,\ldots,y_i)=y_{i+1}|r_{i+1}=y_{i+1}].\frac{1}{2} \\
&+ Prob[D(y_1,\ldots,y_i)=y_{i+1}|r_{i+1}=\overline{y_{i+1}}].\frac{1}{2} \\
&= Prob[D(y_1,\ldots,y_i)=y_{i+1}|r_{i+1}=y_{i+1}].\frac{1}{2}+\frac{1}{2}\alpha
\end{aligned}$$

$$\begin{aligned}
Prob[D(y_1,\ldots,y_i)=0] &= 1-p_i \\
&= Prob[D(y_1,\ldots,y_i,y_{i+1},\ldots,r_n)=y_{i+1}|r_{i+1}=y_{i+1}].\frac{1}{2} \\
&+ Prob[D(y_1,\ldots,y_i,\overline{y_{i+1}},\ldots,r_n)=y_{i+1}|r_{i+1}=\overline{y_{i+1}}].\frac{1}{2}
\end{aligned}$$

$$\begin{aligned}
1-p_i &= \frac{1}{2}p_{i+1}+\frac{\alpha}{2} \\
\frac{\alpha}{2} &= 1-p_i-\frac{1}{2}(1-p_{i+1}) \\
\text{Required probability} &= 1-p_i+\frac{1}{2}p_{i+1}-\frac{1}{2}(1-p_{i+1}) \\
&= \frac{1}{2}+(p_{i+1}-p_i) \\
&\geq \frac{1}{2}+\frac{1}{n^2}
\end{aligned}$$

This also proves that a distinguisher circuit can also be used to construct a next bit predictor which proves one direction of theorem 1 due to Yao stated in the beginning.

□