1 Derandomizing Logspace Computations (Cont.)

In today’s lecture we will finish the derandomization of Logspace-bounded computations. Our algorithm runs in space $S$, time $2^S$, using only $O(S^3)$ random bits. In the last lecture we proved

**Lemma 1** There exists a constant $k > 0$ such that for all $r, c$, there exists a polynomial time computable $c-$generator $g: \{0,1\}^{r+kc} \rightarrow \{0,1\}^r \times \{0,1\}^r$.

such that $\forall A_1$ and $A_2$

\[
A_1: \{0,1\}^r \rightarrow \{0,1\}^c
\]

\[
A_2: \{0,1\}^r \times \{0,1\}^r \rightarrow \{0,1\}^c
\]

$\forall b_2 \in \{0,1\}^c$

\[
\left| \operatorname{Prob}_{x_1, x_2 \in \{0,1\}^r} [A_2(A_1(x_1), x_2) = b_2] - \operatorname{Prob}_{z \in \{0,1\}^{r+c}} [A_2(A_1(g(z)), g^c(z)) = b_2] \right| < 2^{-c}
\]

We are going to use Lemma 1 recursively, see Figure 1

**Definition 2** $c = 2S^2$

\[
g_i : \{0,1\}^{2(i+1)ks^2} \rightarrow \{0,1\}^{2iks^2} \times \{0,1\}^{2iks^2} \quad \forall i = 1, \ldots S
\]

Define the functions $G_i : \{0,1\}^{2(i+1)ks^2} \rightarrow \{0,1\}^{2iks^2}$ inductively as follows:

\[
G_i(z) = g_i(z) \circ g_i^c(z)
\]

\[
G_i(z) = G_{i-1}(g_i(z)) \circ G_{i-1}(g_i^c(z)) \quad \forall i = 2, \ldots S
\]

**Claim 3** $G_i$ is an $\epsilon_i$-generator for space $S$ and time $2^S \cdot 2^i$ where

\[
\epsilon_i = \frac{2^{(S+1)i} - 1}{2^{S+1} - 1} \cdot 2^{-2S^2}
\]

Note: In the case of $i = S$, the running time is $2^S \cdot 2^S$. $G_S$ takes $O(S^3)$ random bits. $\epsilon_S \approx 2^{-S^2}$, which is pretty small.

**Proof** We proof by induction.

The base case is $G_1(z) = g_1(z) \circ g_1^c(z) : \{0,1\}^{4ks^2} \rightarrow \{0,1\}^{2ks^2} \times \{0,1\}^{2ks^2}$. Thus $r = 2ks^2$, $c = 2S^2 > S$ and $\epsilon_1 = 2^{-2S^2}$.

Inductive part (Figure 2): we assume that

\[
\forall b_1 \left| \operatorname{Prob}_{x_1 \in \{0,1\}^{2^i-1}, x_2 \in \{0,1\}^{2^i}, y_1} [A_1(x_1) = b_1] - \operatorname{Prob}_{y_1} [A_1(G_{i-1}(y_1)) = b_1] \right| < \epsilon_{i-1}
\]

\[
\forall b_1, b_2 \left| \operatorname{Prob}_{x_2 \in \{0,1\}^{2^{i-1}}, y_2} [A_2(b_1, x_2) = b_2] - \operatorname{Prob}_{y_2} [A_2(b_1, G_{i-1}(y_2)) = b_2] \right| < \epsilon_{i-1}
\]
Figure 1: Recursive Pattern. time = $2^n$, $S = O(\log n)$
Then we sum over $b_1 (|b_1| = 2^S)$ and apply the previous two inequalities to get

$$\forall b_2 \quad \left| \text{Prob} \left[ A_2(A_1(x_1), x_2) = b_2 \right] - \text{Prob} \left[ A_2(G_{i-1}(y_1)), G_{i-1}(y_2) = b_2 \right] \right| < 2 \cdot 2^S \epsilon_{i-1}$$

Since $g_i$ is a $2 \cdot S^2$-generator, we have

$$\forall b_2 \quad \left| \text{Prob} \left[ A_2(G_{i-1}(y_1)), G_{i-1}(y_2) = b_2 \right] - \text{Prob} \left[ A_2(A_1(G_{i-1}(y_1)), G_{i-1}(g_i^z(z))) = b_2 \right] \right| < 2^{-2S^2}$$

By triangular inequality,

$$\forall b_2 \quad \left| \text{Prob} \left[ A_2(A_1(x_1), x_2) = b_2 \right] - \text{Prob} \left[ A_2(A_1(G_{i-1}(g_i^z(z))), G_{i-1}(g_i^z(z))) = b_2 \right] \right| < 2 \cdot 2^S \epsilon_{i-1} + 2^{-2S^2} = \epsilon_i$$

Another approach is the recursion using hash functions. We can achieve $S^3$ random bits, and more efficiently only $S^2$ bits by randomly choose hash functions.

Constructs hash functions:

$$h_1, \ldots, h_s = \{0, 1\}^S \to \{0, 1\}^s$$

**Figure 2: Inductive Picture**
And defines

\[ G_1(x) = x \circ h_1(x) \]
\[ G_2(x) = G_1(x) \circ G_1(h_2(x)) \]
\[ \ldots \]
\[ G_i(x) = G_{i-1}(x) \circ G_{i-1}(h_i(x)) \]

2 Hardness vs. Randomness by Nissan Wigderson

**Definition 4** for any \( f : \{0,1\}^n \rightarrow \{0,1\} \), hardness is \( h(n) \) if \( \forall \) circuit \( C \) of size \( \leq h(n) \)

\[ \Pr_{x \in \{0,1\}^n} [ f(x) = C(x) ] < \frac{1}{2} + \frac{1}{h(n)} \]

Assume a function of hardness \( n^{\log n} \) is computable in time \( 2^{n^k} \) for some \( k \), we can prove

\[ \text{BPP} \subseteq \text{DTIME}(2^{n^\epsilon}) \forall \epsilon > 0 \]

\[ \text{hardness } h(n) = 2^{-n^k} \text{ computable in EXP } \Rightarrow \text{BPP} = \text{P}. \]

We will continue this topic in the next lecture.