CS 880: Pseudorandomness and Derandomization	4/10/2013

Lecture 21: Randomness and Space

Instructors: Holger Dell and Dieter van Melkebeek — Scribe: Adam Everspaugh

DRAFT

In this lecture we construct a family of psuedorandom generators that operate in logarithmic space and can ϵ-fool branching programs of constant width.

1 Branching Programs and Generators

Recall that a branching program B of length n and width w is a set of functions $(B_1, B_2, \ldots B_r)$ such that each $B_i : \{0,1\} \to [w]$. A branching program can also be viewed as a layered, acyclic graph in which every layer has the same number of vertices (representing states of the program), and every vertex (except the ones in the last layer) have exactly 2 outgoing edges labelled 0 or 1. These edges indicate moving to a new state based on an input of 0 or 1.

Also recall the INW-generator: $G_i(xy) = G_{i-1}(x)G_{i-1}(E(x,y))$ where E is an extractor $E_i : \{0,1\}^{is} \times \{0,1\}^s \to \{0,1\}^{2^i}$. The INW generator is ϵ-psuedorandom for the following types of constant-width branching programs and seed lengths:

BP	seed length
all	$\approx (\log r)(\log \frac{rw}{\epsilon})$
regular	$\approx (\log r)(\log \frac{w \log r}{\epsilon})$
permutation	$\approx (\log r)(poly(w) + \log \frac{1}{\epsilon})$

A *regular* branching program is one where every pair of consecutive layers forms a bipartite, regular graph. A *permutation* branching program is one where the mapping between any two consecutive layers is a permutation (a bijection).

Note that our seed length for permutation branching programs is not necessarily better than the seed length for regular branching programs because it contains a $poly(w)$ term. However, if we fix the width w to a constant value, then we can fully derandomize permutation branching programs in logarithmic space.

2 Logspace Generator

We'll define a new generator that stretches a seed of length $\approx \log w$ to a length of r (the length of our branching program). We'll do this in logarithmic space under the condition that $r \leq poly(\log w)$ (r is much smaller than it could be).

Use the following $(\frac{n}{2}, \beta)$-extractor:

$$E : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\frac{n}{4}}, \text{ where } d \leq O(\log \frac{n}{\beta}),$$

$$n \doteq \log w, \ t_i \doteq 8(\log w)^\gamma, \text{ with constant } \gamma, 0 < \gamma < 1.$$

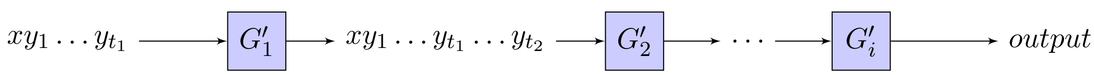
Note: The value γ simply allows us to control the error.

Definition 1 (Logspace Generator G_i). *Define our logspace generator* $G_i : \{0,1\}^{n+t_1 d} \to \{0,1\}^{2n^{1+i\gamma}}$ *and it's helper function* $G'_i : \{0,1\}^n \times \{0,1\}^d \times \ldots \times \{0,1\}^d \to \{0,1\}^{t_i \frac{n}{4}}$ *recursively as:*

$$G'_i(x\ y_1\ y_2 \ldots y_{t_i}) \doteq E(x, y_1)E(x, y_2) \ldots E(x, y_{t_i})$$

$$G_i \doteq G'_i \circ G'_{i-1} \circ \ldots \circ G'_2 \circ G'_1$$

We can visualize an application of G_i as a series of applications of the helper function G'_i where the output of G'_1 is fed into G'_2 and so on. Each application of the helper function G'_i expands the input to a larger output by rearranging the inputs that go into the extractor E. This gives G_i an output length that is exponentially longer than it's input.



Let's analyze how much G_i stretches it's input. Our input is of length $n + t_1 d$:

$$n + t_1 d \leq n + \delta n \gamma \log \frac{n}{\beta}$$

$$\text{If } \log \frac{1}{\beta} \leq n^{1-\gamma} \Rightarrow$$

$$n + \delta n \gamma \log \frac{n}{\beta} \leq O(n) = \log w$$

And our output is of length $t_i \frac{n}{4} = 2n^{1+i\gamma}$. So, as long as β is "not too small", our generator G_i stretches $O(\log w)$ bits to $\to poly(\log w)$ bits.

2.1 Space Requirements

A quick analysis of G_i shows that it can be computed in logarithmic space:

- $E(x, y)$ can be computed in $space = O(n) = O(\log w)$.
- Each G'_i runs E iteratively, so G'_i can be computed is $space = O(\log w)$.
- G_i can be comnputed in $space = i \cdot \log w \leq O(\log w)$

2.2 Psuedorandomness for Branching Programs

We'll prove inductively that the logspace generator G_i is ϵ-psuedorandom for branching programs of constant width and some bounded length.

Theorem 1 (G_i ϵ-fools branching programs). *G_i is ϵ-psuedorandom for branching programs of constant width and length $t_i \frac{n}{4}$ where $\epsilon = t_i(\beta + \frac{1}{w})$.*

For our proof, we'll start with the following lemma.

Lemma 2 (Basis for G_1). *G_1 is ϵ-psuedorandom for branching programs of constant width and length $t_1 \frac{n}{4}$ where $\epsilon = \epsilon_1 = t_1(\beta + \frac{1}{w})$.*

Note:

$$\epsilon = t_1(\beta + \frac{1}{w}) \leq \frac{\log w}{w} + \beta \log w.$$

So, we require that $\beta = \frac{1}{poly(w)}$ to ensure our generator can be computed in *space* $= \log w$.

Consider the sequence of bits output from $G_1(x\ y_1 \dots y_{t_1})$. Each bit takes us to a new layer in a branching program B, but we'll ignore intervening layers and only examine the layers that we arrive at after applying the $\frac{n}{4}$ bits from each extractor call $E(x, y_i)$. Call this sequence of layers: $L_1, L_2, \dots L_{l-1}, L_l$.

Let P_l be the distribution on layer L_l after the psuedorandom walk with input $G_1(xy_1 \dots y_{t_1})$. Let T_l be distribution of the same definition with uniformly random input. Notice that:

$$P_l^u(v) = \Pr[\text{psuedorandom walk starting at u ends at v}]$$

$$T_l^u(v) = \Pr[\text{uniformly random walk starting at u ends at v}]$$

$$\text{where } u \in L_i, v \in L_l, \text{ for some layer } L_i \text{ in branching program } B.$$

Claim. $d_{stat}(T_l, P_l) \leq l \cdot (\frac{1}{w} + \beta)$

Proof (Claim).

$$d_{stat}(T_l, P_l)$$

$$= d_{stat}(\Sigma_{u \in L_{l-1}} T_{l-1} \cdot T_l^u, \Sigma_{u \in L_{l-1}} P_{l-1} \cdot P_l^u)$$

By the triangle inequality $d(A, C) \leq d(A, B) + d(B, C)$:

$$\leq d_{stat}(\Sigma_u T_{l-1}(u) \cdot T_l^u, \Sigma_u P_{l-1}(u) \cdot T_l^u) + d_{stat}(\Sigma_u P_{l-1}(u) \cdot T_l^u, \Sigma_u P_{l-1}(u) \cdot P_l^u)$$

Terms $d(A, B)$ only differ by their distributions, and terms $d(B, C)$ can be factored:

$$\leq d_{stat}(T_{l-1}, P_{l-1}) + \Sigma_u(P_{l-1}(u) \cdot d_{stat}(T_l^u, P_l^u))$$

We can split our sum based on the "hard to reach" vertices and the remaining vertices. Let set of "hard to reach" vertices be $H = \{u | u \in L_{l-1}, P_{l-1}(u) < \frac{1}{w^2}\}$. And let the remaining "easier to reach" vertices be the set $E = \{u | u \in L_{l-1}, P_{l-1}(u) \geq \frac{1}{w^2}\}$.

$$\leq (l-1)(\frac{1}{w} + \beta) + \Sigma_{u \in H}(\frac{1}{w^2} \cdot 1) + \Sigma_{u \in E}(P_{l-1}(w)) \cdot d_{stat}(T_L^u, P_l^u)$$

Our sum of "easier to reach" vertices in E is the sum over a probability distribution, so it has a value of at most 1 and our statistical distance is at most β by the guarantee of our extractor, so:

$$\leq (l-1)(\frac{1}{w} + \beta) + \frac{1}{w} + \beta$$

$$\leq l(\frac{1}{w} + \beta)$$

□

Lemma 3 (Inductive Step). *If G_i is ϵ-psuedorandom for a branching program B of constant width w and bounded length, then G_{i+1} is also ϵ-psuedorandom for B.*

Proof. First, observe that:

$$\begin{aligned} G_i &= (G'_i \circ G'_{i-1} \circ \dots G'_1) \\ G_{i+1} &= G'_{i+1} \circ (G'_i \circ G'_{i-1} \circ \dots G'_1) \\ G_{i+1} &= G'_{i+1} \circ (G_i) \end{aligned}$$

Let $B' = (B \circ G_i)$ and observe that B' is also a branching program of constant width and $space = O(w)$. By our claim, G'_{i+1} is ϵ-psuedorandom for B' and this implies that $G'_{i+1} \circ G_i$ is ϵ-psuedorandom for B. Since $G_{i+1} = G'_{i+1} \circ G_i$, then we have:

$$G_i \ \epsilon\text{-psuedorandom for } B \Longrightarrow G_{i+1} \ \epsilon\text{-psuedorandom for } B$$

□