CS 880: Pseudorandomness and Derandomization 2/11/2013

Lecture 6: Small-Bias Generators

Instructors: Holger Dell and Dieter van Melkebeek Scribe: Kevin Kowalski

In the previous lectures, we discussed k-wise uniform generators, which can be viewed as ϵ-PRGs for k-wise uniformity with $\epsilon = 0$ – the pseudorandom distributions they induce look perfectly random if we restrict our attention to any k components. In this lecture we develop ϵ-PRGs for polynomials of low degree d with $\epsilon > 0$. Modulo the error ϵ, they can be viewed as generalizations of d-wise uniform generators. We focus on PRGs for the special case $d = 1$, which are known as small-bias generators. Based on a connection with error-correcting codes, we present a construction with seed length $O(\log(r/\epsilon))$. The notions and constructions apply to polynomials over any finite field $\mathbb{F}_q$ but we only discuss the case $q = 2$.

1 Pseudorandom Generators for Low-Degree Polynomials

A distribution D_r on $\{0,1\}^r$ is *ϵ-pseudorandom for polynomials of degree d* if, for every polynomial $p : \{0,1\}^r \to \{0,1\}$ of degree at most d,

$$d_{\text{stat}}\big(p(D_r), p(U_r)\big) \leq \epsilon\,. \tag{1}$$

An ϵ-PRG for polynomials of degree d is a collection of mappings $G = (G_r)_{r\in\mathbb{N}}$ with $G_r : \{0,1\}^{\ell(r)} \to \{0,1\}^r$ such that $G_r(U_\ell)$ is ϵ-pseudorandom for polynomials of degree d. Intuitively, a PRG for polynomials of degree d fools all algorithms that only use their bits of randomness after they are passed through a polynomial of degree at most d. In other words, the output $A(x,\rho)$ of the algorithm on input x and random bit sequence ρ can be written as $f(x, p_x(\rho))$, where f is an arbitrary function and p_x is a polynomial of degree at most d whose coefficients may depend on the input x in an arbitrary way.

Relationship to k-wise uniform generators. Distributions that are pseudorandom for polynomials of degree k are "almost" k-wise uniform, in a sense we will describe below.

Recall that for a k-wise uniform distribution D_r, any indices $i_1, \ldots, i_k \in [r]$, and bits $b_1, \ldots, b_k \in \{0,1\}$, we have that

$$\begin{aligned}
\frac{1}{2^k} &= \Pr_{\rho\leftarrow D_r}\Big[\rho_{i_1} = b_1 \wedge \cdots \wedge \rho_{i_k} = b_k\Big] \\
&= \Pr_{\rho\leftarrow D_r}\left[\prod_{j=1}^{k}(\rho_{i_j} - \overline{b_j}) = 1\right] \\
&= \Pr\left[p_b(D_r) = 1\right]\,,
\end{aligned}$$

where $p_b(\rho) \doteq \prod_{j=1}^{k}(\rho_{i_j} - \overline{b_j})$ is interpreted as a degree-k polynomial on the variables $\rho_1, \ldots, \rho_r$ (in which only the variables with indices $i_1, \ldots, i_k$ appear). What we have just shown is that every k-wise uniform generator D_r fools all degree-k polynomials of the form p_b without error, that is,

$$\Big|\Pr\left[p_b(D_r) = 1\right] - \Pr\left[p_b(U_r) = 1\right]\Big| = 0\,.$$

Conversely, if D_r is ϵ-pseudorandom for polynomials of degree k, then in particular it ϵ-fools the polynomials p_b, that is, it has the property that

$$\Big| \Pr\left[p_b(D_r) = 1\right] - \Pr\left[p_b(U_r) = 1\right] \Big| \leq \epsilon\,,$$

since the statistical distance $d_{\text{stat}}\big(p_b(D_r), p_b(U_r)\big)$ bounds the difference between the two distributions in the probability they assign to any event. When we sum over all possible choices of $b \in \{0,1\}^k$, we have that

$$d_{\text{stat}}(D_r|_{i_1,\ldots,i_k}, U_k) = \frac{1}{2}\sum_b \Big| \Pr\left[p_b(D_r) = 1\right] - \Pr\left[p_b(U_r) = 1\right] \Big| \leq 2^{k-1}\cdot\epsilon\,. \tag{2}$$

For $\epsilon = 0$, this implies that every ϵ-pseudorandom distribution for degree-k polynomials is also k-wise uniform. What we get for $\epsilon > 0$ is a generalization of k-wise uniform distributions. While the moniker "k-wise uniform" requires that the distribution be perfectly uniform when looking at any k-subset, "pseudorandom for degree-k polynomials" implies that the distribution is *close* to uniform. In the next lecture we will strengthen this connection in two ways: (i) it turns out that being ϵ-pseudorandom for degree-1 polynomials is sufficient, and (ii) we can improve the right-hand side of (2) to $\sqrt{2^k - 1}\cdot\epsilon$.

From degree one to higher degree. There is another sense in which pseudorandomness for degree-1 polynomials is sufficient. Once we have a distribution D_r that is pseudorandom for degree-1 polynomials, we can obtain a distribution that is pseudorandom for degree d by taking d independent samples from D_r and outputting their component-wise XOR. The following theorem gives us an upper bound on how the error deteriorates in this construction.

Theorem 1. *Let D_r be a distribution on $\{0,1\}^r$ that is ϵ-pseudorandom for polynomials of degree 1. Then for every integer $d \geq 1$, the XOR of d independent samples from D_r is ϵ'-pseudorandom for polynomials of degree d, where $\epsilon' = 16\cdot\epsilon^{1/2^{d-1}}$.*

We refer to [Vio09] for a proof of Theorem 1. Note that the bound on the error ϵ' quickly converges to 1 as d increases. It is open whether this behavior is inherent or whether the bound can be improved. Theorem 1 is known to be optimal in terms of the number of samples: $d-1$ samples do not suffice [BV07].

2 Small-Bias Generators

We now consider pseudorandom generators for polynomials of degree 1, which are equivalent to so-called *small-bias generators*. We begin with the definition of the bias of a distribution, and then prove the just-mentioned connection to generators for polynomials of degree 1.

Definition 1 (Bias of a distribution). *Let D_r be a distribution on $\{0,1\}^r$ and $a \in \{0,1\}^r$. The* bias *of D_r with respect to a is*

$$\text{bias}_a(D_r) \doteq \Pr\Big[\langle a, D_r\rangle = 0\Big] - \Pr\Big[\langle a, D_r\rangle = 1\Big]\,.$$

We say that D_r has bias at most β if $\big|\text{bias}_a(D_r)\big| \leq \beta$ holds for all $a \in \{0,1\}^r \setminus \{0^r\}$.

Note that the bias of a distribution D_r with respect to 0^r is always 1, and that the uniform distribution has bias 0. In fact, as we will see next lecture, the uniform distribution is the only distribution that has bias 0, and having small bias implies being close to the uniform distribution in statistical distance, although there is a quantitative loss. As such, distributions with small bias are natural substitutes for the uniform distribution and, depending on how the algorithm uses its randomness, may work as well as the uniform one. Moreover, they can be generated from a small number of truly random bits, and thus form the basis for PRGs, which we refer to as *small-bias generators*.

Definition 2 (Small-bias generator). *A generator $G_r : \{0,1\}^\ell \to \{0,1\}^r$ whose output distribution $G_r(U_\ell)$ has bias at most β is called a β-bias generator.*

We are ready to state and prove the connection between small-bias generators and PRGs for polynomials of degree 1.

Proposition 2. *For all $\epsilon \geq 0$, a distribution D_r is ϵ-pseudorandom for polynomials of degree 1 if and only if D_r has bias at most $\beta = 2\epsilon$.*

Proof. First note that all polynomials $p : \{0,1\}^r \to \{0,1\}$ of degree at most one are of the form $p(\rho) = \langle a, \rho \rangle + b$ for some $a \in \{0,1\}^r$ and $b \in \{0,1\}$. Furthermore, p is constant if and only if $a = 0^r$, and in this case, (1) trivially holds for any $\epsilon \geq 0$.

In the case that $a \neq 0^r$, we have

$$\begin{aligned}
d_{\text{stat}}\big(p(D_r), p(U_r)\big) &= \Big| \Pr[p(D_r) = 0] - \Pr[p(U_r) = 0] \Big| \\
&= \Big| \Pr[p(D_r) = 0] - 1/2 \Big| \\
&= \left| \Pr[p(D_r) = 0] - \frac{\Pr[p(D_r) = 0] + \Pr[p(D_r) = 1]}{2} \right| \\
&= \frac{1}{2} \Big| \Pr[p(D_r) = 0] - \Pr[p(D_r) = 1] \Big| \\
&= \frac{1}{2} \big| \text{bias}_a(D_r) \big|
\end{aligned}$$

In particular, this implies that, for all $a \neq 0^r$ and all b,

$$d_{\text{stat}}\big(p(D_r), p(U_r)\big) \leq \epsilon \iff \big| \text{bias}_a(D_r) \big| \leq 2\epsilon = \beta.$$

Since this holds for all non-constant polynomials p of degree at most 1 and the case of constant polynomials p is trivial, the statement follows from the definition of ϵ-pseudorandomness for polynomials of degree 1, and the definition of the bias of a distribution. □

3 Connection with Error-Correcting Codes

Just as in the case of k-wise uniform generators, small-bias generators have a close connection to error-correcting codes. Given a β-bias generator $G : \{0,1\}^\ell \to \{0,1\}^r$, we can write the function table of G_r as a $2^\ell \times r$ matrix M over $\{0,1\}$ such that the i-th row is the bit vector $G(\sigma_i)$, where σ_i

denotes the i-th seed in some ordering of $\{0,1\}^\ell$. For example, the generator G with the following function table would produce the corresponding matrix M:

$$\begin{array}{c|c} \sigma & G(\sigma) \\ \hline \sigma_1 & 110\ldots1 \\ \sigma_2 & 011\ldots0 \\ \vdots & \vdots \\ \sigma_{2^\ell} & 101\ldots1 \end{array} \qquad \longrightarrow \qquad M = \begin{bmatrix} 1 & 1 & 0 & \cdots & 1 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 1 & \cdots & 1 \end{bmatrix}$$

We can view this matrix as the generator matrix of some linear error-correcting code. Given a plain-text word $a \in \{0,1\}^r$, the corresponding codeword under the above code is

$$M \cdot a = \Big[\langle a, G(\sigma)\rangle\Big]_{\sigma\in\{0,1\}^\ell}.$$

This implies that a can be viewed as a mask, and applying M to a computes the parity of each row of M over the positions at which a is 1. Since G is at most β-biased, the relative weight of Ma (i.e., the number of 1's in Ma divided by the length of the vector) must be in the range $[\frac{1}{2} - \frac{\beta}{2}, \frac{1}{2} + \frac{\beta}{2}]$. Thus, the requirement that G has small bias is equivalent to the condition that the code generated by M is such that the relative weight of every nonzero codeword is $\beta/2$-close to $1/2$. A code with the latter property is called $\beta/2$-*balanced*.

Proposition 3. *Let $G : \{0,1\}^\ell \to \{0,1\}^r$ be a generator and M be the $(2^\ell \times r)$-matrix over $\{0,1\}$ consisting of the rows $G(\sigma)$ for $\sigma \in \{0,1\}^\ell$. Then G has bias at most β if and only if all nonzero codewords of the linear code generated by M have a relative weight in the range $[\frac{1}{2} - \frac{\beta}{2}, \frac{1}{2} + \frac{\beta}{2}]$.*

4 Construction

The connection with error-correcting codes given by Proposition 3 suggests that we can construct a small-bias generator by finding an appropriate error-correcting code. We now develop two closely related small-bias generators based on that connection.

Choosing a suitable code. The Hadamard code has the property that every non-zero codeword has relative weight exactly $1/2$, so at first glance it might seem like an attractive choice. However, it is inappropriate here because the generator matrix has size $2^r \times r$, which means that the seed length of the PRG is $\ell = r$. Indeed, the induced generator would output "pseudorandom" strings that are exactly as long as the seed and, in fact, implement the identity mapping. Since the purpose of a pseudorandom generator is to reduce the number of random bits required, the Hadamard code clearly will not suffice.

Concatenating the Hadamard code with the Reed–Solomon code, however, does give a useful pseudorandom generator. Recall from Lecture 3 that the Hadamard and Reed–Solomon codes are linear, with characteristics

$$\begin{array}{rl} \text{Reed–Solomon} & [q, k, q-k+1]_q \text{ and} \\ \text{Hadamard} & [q, \log_2 q, q/2]_2\,. \end{array}$$

The concatenated code is constructed by first applying the Reed–Solomon generator matrix to each block of size k in the input (interpreted as a vector of elements in $\mathbb{F}_q$), and then applying

the Hadamard generator matrix to each block of size $\log_2 q$ in the intermediate code (interpreted as a vector of elements in $\mathbb{F}_2$) to get the output. In particular, this code has characteristics $[q^2, k \log_2 q, \delta \cdot q^2]_2$, where

$$\delta \geq \frac{q-k+1}{q} \cdot \frac{1}{2} = \frac{1}{2} - \frac{k-1}{2q}.$$

The fact that the relative distance of the concatenated code is close to $1/2$ means that no codeword has relative weight much below $1/2$. In principle, this still leaves the possibility that some codewords have relative weight significantly larger than $1/2$. However, since every codeword of the Hadamard code has relative weight at most $1/2$, every codeword of a concatenation with the Hadamard code has relative weight at most $1/2$.

Thus, by choosing q and k so that $(k-1)/2q \leq \beta/2$, or equivalently, $(k-1)/q \leq \beta$, the generator matrix of the code described above gives us a function table for a generator with bias β.

Selecting suitable parameters. Now, we would like to formulate our choice of q and k in terms of β (the bound on the bias) and r (the length of the output string) while keeping ℓ (the length of the random seed we need) as small as possible.

Recall that our final code has characteristic $[q^2, k \log_2 q, \delta \cdot q^2]_2$, so the generator matrix of the code has dimensions $q^2 \times k \log_2 q$. Matching the dimensions of the generator matrix with the dimensions of the function table, we obtain the equalities

$$\begin{aligned} q^2 = 2^\ell &\iff \ell = 2 \log_2 q\,, \\ k \log_2 q = r &\iff r = k \log_2 q\,. \end{aligned} \tag{3}$$

Since we do not have to make full use of the pseudorandom bits generated by the generator, the second equality can be rewritten as the inequality $r \leq k \log_2 q$. For simplicity of analysis, we will fix $k = r$, though we could get a shorter relative seed length by using all the pseudorandom bits. For future reference, note that dropping the factor $\log_2 q$ corresponds to only using binary strings $\{0,1\}^k$ rather than all of $\mathbb{F}_q^k$ as the messages to which we apply the Reed–Solomon code.

With these equalities, we can then write the condition $\frac{k-1}{q} \leq \beta$ as

$$\begin{aligned} &\frac{k-1}{q} \leq \beta \\ \iff &q \geq \frac{r-1}{\beta}, \end{aligned}$$

so if we choose q to be the smallest power of two that is at least $(r-1)/\beta$, we get that $\ell \in O(\log(r/\beta))$. Since all the underlying computations can be performed efficiently, we conclude the following.

Theorem 4. *For every positive integer r and positive real β, there exists an efficiently computable β-bias generator $G : \{0,1\}^\ell \to \{0,1\}^r$ with $\ell = O\big(\log(r/\beta)\big)$.*

By the fact that the XOR of d small-bias generators fools degree d polynomials (Theorem 1) and the equivalence between small-bias generators and PRGs for polynomials of degree 1 (Proposition 2), Theorem 4 yields the following corollary.

Corollary 5. *For every positive integers d and r, and every real $\epsilon > 0$, there exists an efficiently computable ϵ-pseudorandom generator $G : \{0,1\}^\ell \to \{0,1\}^r$ for polynomials of degree d with $\ell = O\big(d \cdot \log(r) + d2^d \cdot \log(1/\epsilon)\big)$.*

Note that the seed length in Corollary 5 is only nontrivial for sub-logarithmic degree d.

In order to figure out just how "efficiently computable" the small-bias generator from Theorem 4 is, we explicitly construct the generator matrix of the corresponding code.

Explicit construction. We first analyze the exact construction as described above, i.e., the one where we do not drop the factor $\log_2 q$ in (3) as we did in our simplified analysis. We need to explicitly construct the generator matrix of the concatenation of the Reed–Solomon code with the Hadamard code. Since both codes are linear, we might expect the final generator matrix to be a simple product of matrices for each of the two codes, but the larger alphabet of the Reed–Solomon code (i.e., $\mathbb{F}_q$ as opposed to $\{0,1\}$) makes this more complicated.

Recall that the generator matrix G_{RS} for the Reed–Solomon code is a $q \times k$ Vandermonde matrix over $\mathbb{F}_q$, where there is one row for every element of $\mathbb{F}_q$, and the row corresponding to $u \in \mathbb{F}_q$ consists of the elements

$$\begin{bmatrix} u^0 & u^1 & \cdots & u^{k-1} \end{bmatrix}, \tag{4}$$

and all multiplication is performed over the field $\mathbb{F}_q$. Let $m = \log_2 q$, so that each element $u \in \mathbb{F}_q$ can be represented as a binary vector of size m. The $[q,k]_q$ Reed–Solomon code can also be viewed as a $[qm, km]_2$ code. We first need to figure out the generator matrix for the latter code.

In order to do so, we associate to each $u \in \mathbb{F}_q$ the polynomial

$$u(\xi) \doteq \sum_{p=1}^{m} u_p \xi^{p-1},$$

where each $u_p \in \{0,1\}$. These polynomials have the nice property that for any $a, b, c \in \mathbb{F}_q$,

$$a \cdot b = c \iff a(\xi) \cdot b(\xi) = c(\xi),$$

where the multiplication on the left takes place over $\mathbb{F}_q$, and the polynomial multiplication on the right takes place over $\mathbb{F}_2[\xi]$ modulo some fixed irreducible polynomial of degree m. Note that for any fixed b, the coefficients of $c(\xi)$ are linear combinations over $\mathbb{F}_2$ of the coefficients of $a(\xi)$. Thus, seen as vectors of size m over $\mathbb{F}_2$, we can write $c = M_b \cdot a$, where M_b is an $(m \times m)$-matrix over $\mathbb{F}_2$ that corresponds to multiplication with b (for the chosen irreducible polynomial).

Given a plain-text word $x \in \mathbb{F}_q^k$, its encoding y under G_{RS} is given by

$$y = \left(\sum_{j=1}^{k} x_j u^{j-1} \right)_{u \in \mathbb{F}_q}.$$

When the components y_u of y are interpreted as vectors of length m over $\mathbb{F}_2$, they can be written as

$$y_u = \sum_{j=1}^{k} M_{u^{j-1}} \cdot x_j \,,$$

where the matrix-vector product is computed over $\mathbb{F}_2$. Thus, for every row (4) of the generator matrix for the $[q,k]_q$ Reed–Solomon code, there are m rows in the generator matrix for the corresponding $[qm,km]_2$ code, and they are obtained by replacing every entry $v \doteq u^{j-1}$ in (4) by M_v. For a given ordering of the elements $u \in \mathbb{F}_q$, this gives us explicit matrices $T_{ij} \in \mathbb{F}_2^{m\times m}$ such that

$$y = \begin{bmatrix} T_{11} & T_{12} & \cdots & T_{1k} \\ T_{21} & T_{22} & \cdots & T_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ T_{q1} & T_{q2} & \cdots & T_{qk} \end{bmatrix} \cdot x,$$

so the revised Reed–Solomon matrix is equal to the original one, except with each (i,j)-entry replaced with the corresponding matrix T_{ij}.

All that remains is to use the Hadamard code to encode y. We apply the $[q,m]_2$ Hadamard code to each of the q components y_i of y, where we view y_i as a vector of length m over $\mathbb{F}_2$. Let H be the generator matrix for the $[q,m]_2$ Hadamard code. Since the Hadamard code is linear over $\mathbb{F}_2$, we have that

$$\begin{aligned} \mathrm{Had}(y) = \begin{bmatrix} Hy_1 \\ Hy_2 \\ \vdots \\ Hy_q \end{bmatrix} &= \begin{bmatrix} H \cdot \left(\sum_{j=1}^k T_{1j}x_j\right) \\ H \cdot \left(\sum_{j=1}^k T_{2j}x_j\right) \\ \vdots \\ H \cdot \left(\sum_{j=1}^k T_{qj}x_j\right) \end{bmatrix} \\ &= \begin{bmatrix} \sum_{j=1}^k HT_{1j}x_j \\ \sum_{j=1}^k HT_{2j}x_j \\ \vdots \\ \sum_{j=1}^k HT_{1j}x_j \end{bmatrix}, \end{aligned}$$

so our final generator matrix is the $q^2 \times mk$ matrix

$$G = \begin{bmatrix} HT_{11} & HT_{12} & \cdots & HT_{1k} \\ HT_{21} & HT_{22} & \cdots & HT_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ HT_{q1} & HT_{q2} & \cdots & HT_{qk} \end{bmatrix},$$

where T_{ij} equals the multiplication matrix $M_{u^{j-1}}$ for u the ith element of $\mathbb{F}_q$ in the underlying ordering.

Simpler explicit construction. Next we analyze the variant of the construction that corresponds to dropping the $\log_2 q$ factor in (3), which corresponds to only considering plain-text words $x \in \{0,1\}^k$ rather than $\mathbb{F}_q^k$. Again, we start with the Reed–Solomon generator G_{RS}, but instead of replacing each entry u^{j-1} by the $(m \times m)$-matrix $M_{u^{j-1}}$ over $\mathbb{F}_2$ that represents multiplication with u^{j-1} over $\mathbb{F}_q$, we replace it by the $(m \times 1)$-matrix over $\mathbb{F}_2$ that represents u^{j-1} as a column vector of length m over $\mathbb{F}_2$. This gives us a $(qm \times k)$-matrix over $\mathbb{F}_2$. Applying the Hadamard code as before gives us the final generator matrix where the q rows corresponding to $u \in \mathbb{F}_q$ are given by

$$\begin{bmatrix} H \cdot 1 & H \cdot u & H \cdot u^2 & \cdots & H \cdot u^{k-1} \end{bmatrix},$$

where each u^{j-1} represents a vector of length m over $\mathbb{F}_2$.

This variant only produces a $(q^2 \times k)$-matrix rather than a $(q^2 \times km)$-matrix. In terms of the corresponding small-bias generator given by Proposition 3, this means a reduction in the number of pseudorandom bits r by a factor of $m = \log_2 q$. However, the entries of the matrix and therefore the output of the generator are considerably easier to describe. Recall that the Hadamard encoding of a binary vector of length m consists of the inner product with every vector $v \in \mathbb{F}_2^m$ over $\mathbb{F}_2$. Thus, we can index the rows of the generator matrix by $(u, v) \in \mathbb{F}_2^m \times \mathbb{F}_2^m$. Equivalently, the small-bias generator G' takes a seed of length $2m$ and parses it as two random strings u, v, each of length m. The corresponding output is then

$$\begin{aligned} G'(u,v) &= \left(\text{the } v\text{-th entry of } Hu^{j-1}\right)_{j=1}^{k} \\ &= \left(\langle u^{j-1}, v\rangle\right)_{j=1}^{r}. \end{aligned} \tag{5}$$

Note that u is first viewed as an element of $\mathbb{F}_{2^m}$ and raised to the power $(j-1)$ in that field.[1] Then u^{j-1} is viewed as an element of $\mathbb{F}_2^m$ and we take the inner product of that vector with $v \in \mathbb{F}_2^m$.

The construction is elegant enough to merit its own statement.

Theorem 6. *Let $G' : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}^r$ be defined by (5), where u^{j-1} denotes u, viewed as an element of $\mathbb{F}_{2^m}$, raised to the power $j-1$. Then G' has bias at most β, where $\beta = \frac{r-1}{2^m}$.*

The earlier derivation proves Theorem 6. However, it is instructive to revisit the argument and directly show that G' has small bias. Let $D = G'(U_{2m})$ and consider any nonzero vector $a \in \{0,1\}^r$. We want to show that $\left|\text{bias}_a(D)\right| = \left|\Pr[\langle D, a\rangle = 0] - \Pr[\langle D, a\rangle = 1]\right| \leq \beta$ holds. By linearity of the inner product we have that

$$\sum_{j=1}^{r} a_j G'(u,v)_j = \sum_{j=1}^{r} a_j \langle u^{j-1}, v\rangle = \langle p_a(u), v\rangle,$$

where $p_a(u) = \sum_{j=1}^{r} a_j u^{j-1}$. Any u for which $p_a(u) \neq 0$ has no net contribution to $\text{bias}_a(D)$ since the number of v's for which $\langle p_a(u), v\rangle = 0$ is the same as the number of v's for which $\langle p_a(u), v\rangle = 1$. Any u for which $p_a(u) = 0$ contributes $\frac{1}{2^m}$ to $\text{bias}_a(D)$. Since p_a is a polynomial of degree at most $r-1$, there are at most $r-1$ values u (out of 2^m possible) for which $p_a(u) = 0$, and therefore, $\text{bias}_a(D)$ is in the range $[0, \frac{r-1}{2^m}]$. Since this holds for every $a \neq 0$, it follows that D has bias at most $\beta = \frac{r-1}{2^m}$.

References

[BV07] Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. In *Proceedings of the 48th Annual Symposium on Foundations of Computer Science, FOCS 2007*, pages 41–51, 2007.

[Vio09] Emanuele Viola. The sum of d small-bias generators fools polynomials of degree d. *Computational Complexity*, 18(2):209–217, 2009.

[1] This presumes the choice of some underlying irreducible polynomial of degree m over $\mathbb{F}_2$. Any choice will do.