Hardness vs. Randomness
Result by Nisan and Widgerson
Writeup by Gasarch

This is a write up of results of Nisan and Wigderson [1] along the lines of "if hard problems exist then randomized algorithms can be derandomized."

We DEFINE many terms, STATE how they relate, and then PROVE that that is how they relate. We parameterize everything and then later set the parameters to obtain the main result.

1 Intuition

We want to show that

If there exists f "hard" then P = BPP.

To do this we will need to be able to generate psuedo-random sequences. That is, given a short string of truly random bits we want a long string of psuedo-random bits.

Consider the following thought experiment. Let $f : \{0,1\}^m \rightarrow \{0,1\}$ be a hard function. Let $B_1, \ldots, B_M$ where be all $M = \binom{m}{m/2}$ $m/2$-sized subsets of $\{1, \ldots, m\}$. Let

$B_i = \{u_1^i < \cdots < u_{m/2}^i\}$

Now let $g : \{0,1\}^{2m} \rightarrow \{0,1\}^M$ be

$g(x_1 x_2 \cdots x_{2m}) =$

$f(x_{u_1^1} x_{u_2^1} \cdots x_{u_{m/2}^1}) \cdot f(x_{u_1^2} x_{u_2^2} \cdots x_{u_{m/2}^2}) \cdot \cdots \cdot f(x_{u_1^M} x_{u_2^M} \cdots x_{u_{m/2}^M})$.

If f is hard then is g psuedorandom?

Arguments for: Any one bit of $g(x_1 \cdots x_{28})$ is hard.

Arguments against: The B_i blocks overlap alot so that, say the 17th bit and the 24th bit of $g(x_1 \cdots x_{28})$ may be highly correlated.

The problem is that we picked ALL of the 14-element subsets of $\{1, \ldots, 28\}$. We want to pick a large number of subsets but with very little intersection. The next lemma allows us to do this.

Lemma 1.1 *For every $c \geq 2$, for every function $L : \mathsf{N} \rightarrow \mathsf{N}$ (think $\log n$) for almost all n, there exists sets $B_1, \ldots, B_n$ such that*

1. *$B_1, \ldots, B_n \subseteq \{1, \ldots, c^2 L(n)\}$.*
2. *For every i, $|B_i| = cL(n)$.*

3. *For every $i \neq j$, $|B_i \cap B_j| < L(n)$.*

Moreover, there is an algorithm that will, given n (in unary) produce the sets $B_1, \ldots, B_n$ in time poly in n and $2^{L(n)}$. The polynomial depends on c and is denoted $p_c(n, 2^{L(n)})$. If $L(n) = \log n$ then we just denote it $p_c(n)$.

Proof:
We give an algorithm and then prove that it works. The prove that it works is nonconstructive (probabilitistic) but the algorithm is not. We use L for $L(n)$ throughout.

First, list all $\binom{c^2L}{cL} < 2^{c^2L}$ subsets of $\{1, \ldots, c^2L\}$ of size cL. We list these as $D_1, \ldots, D_m$ where $m = \binom{c^2L}{cL}$

1. $B_1 = D_1$.
2. For $2 \leq i \leq n$ find the least $j \leq m$ such that D_j is not already picked and, for all $i' < i$, $|D_j \cap B_{i'}| \leq L$.

It is clear that the $B_1, \ldots, B_n$ satisfy the condition. We need to show that step 2 of the algorithm can always be carried out. A simple probabilistic argument shows that this can always be done. ∎

2 Conventions

Throughout the paper we use certain function symbols (e.g. $L(n)$, $f(n)$) for the same thing over and over again. The last section of this paper is a guide to such useage and the reader is advised to flip to it when needed.

Convention 2.1

1. Circuits are fanin-2 circuits.
2. All upper bounds like "$t(n)$" and "$s(n)$" actually mean "$O(t(n))$" and "$O(s(n))$".
3. We will refer to a circuit family $\{C_n\}_{n=1}^{\infty}$ by just C_n.
4. If f (or G) is a function with domain $\{0,1\}^*$ then f_m (or G_m) is that function restricted to $\{0,1\}^m$. The most common cases are $m = n$, $m = L(n)$, $m = cL(n)$, and $m = c^2L(n)$ where $c \geq 2$ and $L : \mathsf{N} \rightarrow \mathsf{N}$.

5. Note that $G_{cL(n)}$, f_n, and C_n are all restrictions of some object to a certain length. There is one crucial difference— C_n will be nonuniform but $G_{cL(n)}$ and f_n will both be uniform.

6. We will sometimes use an expressions like $\{G_{L(n)}(z) : z \in \{0,1\}^{L(n)}\}$. We will mean this to be a *multiset* not a set. The function $G_{L(n)}$ need not be 1-1, hence some elements will appear twice. We count them twice. This will matter in calculating probabilities.

Def 2.2 Let $t(n), r(n) : \mathsf{N} \rightarrow \mathsf{N}$ and $\text{err}(n) : \mathsf{N} \rightarrow \mathsf{Q} \cap (0, 1/2)$. (Think of $t(n), r(n)$ as poly and $\text{err}(n) = 1/4$.) Let $\text{BPP}(t(n), r(n), \text{err}(n))$ (Bounded Probabilistic time, parameterized) be the set of all $A \subseteq \{0,1\}^*$ such that there exists TM M that runs in time $t(n)$ on inputs of the form (x, y) where $|x| = n$ and $|y| = r(n)$ such that the following occurs. Let $x \in \{0,1\}^n$.

1. if $x \in A$ then, for at least $1 - \text{err}(n)$ of $y \in \{0,1\}^{r(n)}$, $M(x, y) = 1$.

2. if $x \notin A$ then, for at least $1 - \text{err}(n)$ of $y \in \{0,1\}^{r(n)}$, $M(x, y) = 0$.

Def 2.3 $\text{BPP} = \bigcup_{k=1}^{\infty} \text{BPP}(n^k, n^k, \frac{1}{4})$.

Proposition 2.4 *Let* $t(n), r(n) : \mathsf{N} \rightarrow \mathsf{N}$ *and* $\text{err}(n) : \mathsf{N} \rightarrow \mathsf{Q} \cap (0, 1/2)$. *(Think of* $t(n), r(n)$ *as poly and* $\text{err}(n) = 1/4$.*)* $\text{BPP}(t(n), r(n), \text{err}(n)) \subseteq \text{DTIME}(2^{r(n)} t(n))$.

Proof: Assume $A \in \text{BPP}(t(n), r(n), \text{err}(n))$ via TM M. Here is an algorithm for A that runs in time $\text{DTIME}(2^{r(n)} t(n))$.

1. Input(x).

2. For every $y \in \{0,1\}^{r(n)}$ run $M(x, y)$ (This takes $2^{r(n)} t(n)$ steps.) Keep count of how many y's make $M(x, y) = 0$ and how many y's make $M(x, y) = 1$.

3. If the number of 0's is in the majority then output 0. If the number of 1's is in the majority then output 1. (One of the two must be in the majority since one of them is at least $1 - \text{err}(n) > \frac{1}{2}$ of the votes. Recall that $\text{err}(n) < \frac{1}{2}$.)

▮

In order to get BPP $\subseteq$ P we will need to find a set $S \subseteq \{0,1\}^{r(n)}$ such that

1. S behaves 'just like' $\{0,1\}^{r(n)}$ in that sampling from it gives approximately the same results as sampling from $\{0,1\}^{r(n)}$.
2. S is small.
3. S can be found in polynomial time.

If we could find such an S we would use it instead of $\{0,1\}^{r(n)}$ in the proof of the above proposition to get BPP $\subseteq$ DTIME$(|S|p(n))$.

Def 2.5 Let $L : \mathsf{N} \rightarrow \mathsf{N}$ (think $\log n$), $s : \mathsf{N} \rightarrow \mathsf{N}$ (think $2^{\epsilon n}$), and diff : $\mathsf{N} \rightarrow \mathsf{Q} \cap (0,1/2)$ (think $\frac{1}{\text{poly}(n)}$). Assume that, for all n, G maps $\{0,1\}^{L(n)}$ into $\{0,1\}^n$. G is $(L(n), s(n), \text{diff}(n))$-*pseudorandom* if

1. (Informally) For all n the set $\{G_{L(n)}(z) : z \in \{0,1\}^{L(n)}\}$ "looks like" $\{0,1\}^n$.
2. (Formally) For almost all n, for every $s(n)$-sized circuit C_n,

   $$|\Pr(C_n(y) = 1 : y \in \{0,1\}^n) - \Pr(C_n(G_n(z)) = 1 : z \in \{0,1\}^{L(n)}| < \text{diff}(n).$$

   (No $s(n)$-sized circuit can tell the two sets apart, up to diff(n). When assuming this is not true we freely use 0 intead of 1 and/or do not use the absolute value signs.)

Note 2.6 If we say that $G \in$ DTIME$(t(n))$ we mean that it runs in time $t(n)$ where n is the length of the *output*.

Def 2.7 Let $L : \mathsf{N} \rightarrow \mathsf{N}$ (think $\log n$), $s : \mathsf{N} \rightarrow \mathsf{N}$ (think $2^{\epsilon n}$), and eps : $\mathsf{N} \rightarrow \mathsf{Q} \cap (0,1/2)$ (think $\frac{1}{\text{poly}(n)}$). Assume that, for all n, G maps $\{0,1\}^{L(n)}$ into $\{0,1\}^n$. G is $(L(n), s(n), \text{eps}(n))$-*next bit predictable* if, for infinitely many n, there exists $i \in \{2,\ldots,n\}$ and a circuit $C_n : \{0,1\}^{i-1} \rightarrow \{0,1\}$ such that

1. C_n is a deterministic circuit of size $s(n)$.

2. For at least $\frac{1}{2} + \text{eps}(n)$ of strings $y \in \{G_{L(n)}(z)[1 : i-1] \mid z \in \{0,1\}^{L(n)}\}$, $C(y) = G_{L(n)}(z)[i]$. ($\{G_{L(n)}(z)[1 : i-1] \mid z \in \{0,1\}^{L(n)}\}$ is a multiset.)

Note 2.8 Since we define this with 'for infinitely many n', the negation is 'for almost all n'

Note 2.9 The probability is *not* over $\mathbf{IMG}(G_{cL(n)})[1 : i]$. The probability is over the strings $z \in \{0,1\}^{L(n)}$ that generate strings of length n, namely $G_{cL(n)}(z)$, but we then only look at the first i bits. Think of it as a strange distribution on $\mathbf{IMG}(G_{L(n)})[1 : i]$.

Example 2.10 Let $c = 1$, $n = 8$, $cL(n) = 3$. $G : \{0,1\}^3 \rightarrow \{0,1\}^8$, and
$G(000) = 01010101$
$G(001) = 10111000$
$G(010) = 01011101$
$G(011) = 11100101$
$G(100) = 00000011$
$G(101) = 01011100$
$G(110) = 11101011$
$G(111) = 11110111$
Say $i = 5$, so a next-bit-predictor would be given 4 bits and try to predict the fifth one. Lets look at the following next-bit-predictor C.
$C(0000) = 0$ (GOOD- 1 elt in im(G) begins 0000, and has next bit 0)
$C(0001) = 1$ (IRRELEVANT- no elt of im(G) begins 0001)
$C(0010) = 0$ (IRRELEVANT- no elt of im(G) begins 0010)
$C(0011) = 1$ (IRRELEVANT- no elt of im(G) begins 0011)
$C(0100) = 0$ (IRRELEVANT- no elt of im(G) begins 0100)
$C(0101) = 1$ (GOOD- 3 elts in im(G) begin 0101, and 2 have next bit 1)
$C(0110) = 0$ (IRRELEVANT- no elt of im(G) begins 0110)
$C(0111) = 1$ (IRRELEVANT- no elt of im(G) begins 0111)
$C(1000) = 0$ (IRRELEVANT- no elt of im(G) begins 1000)
$C(1001) = 1$ (IRRELEVANT- no elt of im(G) begins 1001)
$C(1010) = 1$ (IRRELEVANT- no element of im(G) begins 1010)
$C(1011) = 0$ (BAD- 1 element of im(G) begins 1011 and has next bit 1)
$C(1100) = 0$ (IRRELEVANT- no elt of the im(G) begins 1100)
$C(1101) = 0$ (IRRELEVANT- no elt of the im(G) begins 1101)

$C(1110) = 0$ (2 elts of $\text{im}(G)$ begin 1110, 1 has next bit 0, 1 has next-bit 1)

$C(1111) = 0$ (GOOD- one elt of $\text{im}(G)$ begins 1111 and next-bit is 0)

How well is C doing as a next-bit-predictor? We estimate this by looking at the multiset of prefixes of length 4 of images in G. For each one we also include in parenthesis what the next bit was.

$\{0101(0), 1011(1), 0101(1), 1110(0), 0000(0), 0101(1), 1110(1), 1111(0)\}$

Since $C(0101) = 1$, C is correct on 2 of the elements, but wrong on 1.

Since $C(1011) = 0$, C is wrong on 1 of the elements.

Since $C(1110) = 0$, C is correct on 1 of the elements, but wrong on 1.

Since $C(0000) = 0$, C is correct on 1 of the elements.

Since $C(1111) = 0$, C is correct on 1 of the elements.

So C is correct on 5 strings: $\{0101, 0101, 1110, 0000, 1111\}$. and wrong on 3 strings $\{0101, 1011, 1110\}$.

It is easy to have a circuit that is correct for $\frac{1}{2}$ of the elements of the domain. Let C_0 be the circuit that always outputs 0. Let C_1 be the circuit that always outputs 1. We are tempted to say "One of these is correct at least half of the elements of $\mathbf{IMG}(G_{L(n)})[1 : i-1]$." This is correct, but not useful— we are not using the uniform distribution on $\mathbf{IMG}(G_{L(n)})[1 : i-1]$. We are using a weird distribution. We could say "One of these is correct at least 'half' of the time based on the weird distribution." This is correct, but not quite as rigorous as we'd like. We do the following thought experiment: Map every $z \in \{0,1\}^{L(n)}$ to $G_{L(n)}(z)[i]$. There exists $a \in \{0,1\}$ such that at least half of the x's map to a. The circuit C_a works 'half' the time via our distribution. Asking for a circuit to be correct $\frac{1}{2} + \text{eps}(n)$ of the time does not seem that hard to satisfy. It is just a little bit better than taking C_0 or C_1.

Def 2.11 Let $f : \{0,1\}^* \rightarrow \{0,1\}$. Let f_n be the restriction of f to $\{0,1\}^n$. f is $(S(n), \text{eps}(n))$*–HARD* if there does not exist an $s(n)$-sized circuit C_n that computes, for almost all n, f_n correctly on $\frac{1}{2} + \text{eps}(n)$ of the strings in $\{0,1\}^n$.

3 BOLDFACE DEFS and OUR PLAN

Def 3.1

1. $\mathbf{HARD}_{S(n),\mathrm{eps}(n),T(n)}$: there exists f such that f

$$f \text{ is } (S(n), \mathrm{eps}(n))\textbf{-HARD} \wedge \; f \in \mathrm{DTIME}(T(n)).$$

2. $\mathbf{NNBP}_{L(n),s(n),\mathrm{eps}(n),t(n)}$: there exists G such that

$$G \text{ is } \mathbf{N}\text{ot } (L(n), s(n), \mathrm{eps}(n))\text{-}\mathbf{N}\text{ext } \mathbf{B}\text{it } \mathbf{P}\text{redictable} \wedge G \in \mathrm{DTIME}(t(n).$$

3. $\mathbf{PSRAND}_{L(n),s(n),\mathrm{diff}(n),t(n)}$: there exists G such that

$$G \text{ is } (L(n), s(n), \mathrm{diff}(n))\text{-}\mathbf{PS}\text{eudo}\mathbf{RAND}\text{om} \wedge G \in \mathrm{DTIME}(t(n)).$$

4. $\mathbf{CONTAINS}_{t(n),r(n),\mathrm{err}(n),u(n)}$:

$$\mathrm{BPP}(t(n), r(n), \mathrm{err}(n)) \subseteq \mathrm{DTIME}(u(n)).$$

We want to prove **HARD** $\Rightarrow$ **CONTAINS** with appropriate parameters. The plan is to prove
HARD $\Rightarrow$ **NNBP**,
NNBP $\Rightarrow$ **PSRAND**, and
PSRAND $\Rightarrow$ **CONTAINS**

4 Helpful Easy Lemmas

Lemma 4.1 *If h is a Boolean function on m variables then there is a circuit for h of size $m2^m$*

Proof: We first build an unbounded fan-in circuit and then from that construct a fan-in 2 circuit.

For every row of the truth table that says YES we construct the appropriate AND gate with m inputs. Have all of the outputs of that go to an OR gate with at most 2^m inputs.

There are 2^m AND gates. Each one has m inputs. Each one can be rewritten as $\leq (m-1)$ fanin 2 AND gates. Hence we can replace the AND gate with $(m-1)2^m$ fanin 2 AND gates.

There is 1 2^m-input OR gate. This can be rewritten as $\leq 2^m - 1$ fanin 2 OR gates.

Hence the total number of fan-in 2 gates is $(m-1)2^m + 2^m - 1 = m2^m$. ▮

Example 4.2 If the number of variables is $m = c\log n$ then the circuit is of size $(c\log n)2^{c\log n} = cn^c \log n$. Later on in the paper the cases of $c = 1$ and $c \geq 2$ will be very important.

Lemma 4.3 *Let $c, d \in \mathsf{N}$ and $0 < \alpha < 1$. Let R be a $c \times d$ array of 0's and 1's. If at least αcd of the entries in R are 1's then there exists a column where at least αd of the entries in the column are 1.*

Proof: Assume, by way of contradiction, that for every column $< \alpha c$ or the entries are 1. Then the entire array would have $< \alpha cd$ of its entries being 1. This contradicts the premise. ▮

The following Lemma will not be used until Proposition 6.1; hence you can put off reading it for now.

Lemma 4.4 *Let $i \in \{2, \ldots, n\}$, $D : \{0,1\}^n \rightarrow \{0,1\}$, $f : \{0,1\}^{i-1} \rightarrow 2^{\{0,1\}}$, $H_n : X \rightarrow \{0,1\}^{i-1}$ (X any set), and $\alpha(n) : \mathsf{N} \rightarrow (0,1)$. Assume*

$$\Pr_{x\in X, w_1 = H_n(x), w_2 \in \{0,1\}^{n-i+1}}(D(w_1w_2) \in f(x)) \geq \alpha(n)).$$

(where $x \in X$ uniformly and $w_2 \in \{0,1\}^{n-i+1}$ uniformly). Then there exists $w_o \in \{0,1\}^{n-i+1}$ such that

$$\Pr_{w_1 = H_n(x), x\in X}(D(w_1bw_o) \in f(x)) \geq \alpha(n).$$

Proof: Let R be the $2^{i-1} \times 2^{n-i+1}$ rectangle indexed by $w_1 \in \{0,1\}^{i-1}$ and $w_2 \in \{0,1\}^{n-i+1}$ defined via

$$R(w_1, w_2) = \begin{cases} 1 & \text{if } D(w_1w_2) \in f(x); \\ 0 & \text{if } D(w_1w_2) \notin f(x). \end{cases}$$

By the premise at least $\alpha(n)$ of the entries of R are 1. By Lemma 4.3 there exists a column, indexed by w_o, such that at least $\alpha(n)$ 0f the entries in that column are 1. This w_o suffices. ▮

5 HARD $\Rightarrow$ NNBP

Proposition 5.1 *Assume that there exists $f \in \mathrm{DTIME}(T(n))$ such that, for almost all n, f_n is $(S(n), \mathrm{eps}(n))$–hard. Let $c \geq 2$ and let $s_c(n)$ be such that $s_c(n) << S(cL(n)) - nL(n)2^{L(n)}$. Then there exists G such that the following hold.*

1. *$G \in \mathrm{DTIME}(p_c(n, 2^{L(n)}) + nT(cL(n)))$ (p_c is from Lemma 1.1).*
2. *G restricted to $\{0,1\}^{c^2L(n)}$ maps to $\{0,1\}^n$.*
3. *G is not $(c^2L(n), s(n), \mathrm{eps}(n))$-next bit predictable,*

This can be restated as

$$\mathbf{HARD}_{S(n),\mathrm{eps}(n),T(n)} \Rightarrow (\forall c \geq 2)[\mathbf{NNBP}_{c^2L(n),s_c(n),\mathrm{eps}(n),nT(cL(n))+p_c(n,2^{L(n)})}].$$

Proof: We will keep the $s_c(n)$ term and later note when the condition on it is needed.

We use $f_{cL(n)}$. Let $B_1, \ldots, B_n$ be from Lemma 1.1 with our value of c. Every time we run the algorithm we will need to compute these, so that adds $p_c(n, 2^{L(n)})$ to the run time.

We define G by, for each n, defining $G_{c^2L(n)}$, the restriction of G to $\{0,1\}^{c^2L(n)}$. Let $G_{c^2L(n)}(z_1z_2 \cdots z_{c^2L(n)}) = b_1b_2 \cdots b_n$ where

b_i is obtained by taking $B_i = \{u_1 < \cdots < u_{cL(n)}\}$ and letting

$$b_i = f_{cL(n)}(z_{u_1} \cdots z_{u_{cL(n)}}).$$

Clearly G can be computed in $p_c(n, 2^{L(n)}) + nT(cL(n))$ steps, $p_c(n, 2^{L(n)})$ to generate the B_i's, and then, n times, we compute $f_{cL(n)}$ on $cL(n)$ variables, which takes $T(cL(n))$ steps.

Assume, by way of contradiction, that G is $(c^2L(n), s_c(n), \mathrm{eps}(n))$-next-bit-predictable. Let C_n be a deterministic $s_c(n)$-sized circuit, and $2 \leq i \leq n$ be such that

for over $\frac{1}{2} + \mathrm{eps}(n)$ of the $z' \in \{0,1\}^{c^2L(n)}$,

$$C_n(G_{c^2L(n)}(z')[1 : i-1]) = b_i \text{ where } G_{c^2L(n)}(z) = b_1 \cdots b_n.$$

We rewrite this (for later use) as

for over $\frac{1}{2} + \mathrm{eps}(n)$ of the $(z, w) \in \{0,1\}^{cL(n)} \times \{0,1\}^{c^2L(n)-cL(n)}$,

$$C_n(G_{c^2L(n)}(zw)[1:i-1]) = b_i \text{ where } G_{c^2L(n)}(zw) = b_1 \cdots b_n.$$

Let b_i depend on (renamed) variables $\{z_1, \ldots, z_{cL(n)}\}$, so $b_i = f_{cL(n)}(z_1 z_2 \cdots z_{cL(n)})$. We will use C_n to build a deterministic circuit

$$D(z_1 \cdots z_{cL(n)} \cdot w_{cL(n)+1} \cdots w_{c^2L(n)}).$$

We will then find a restriction of the w-bits so that the remaining circuit is a small deterministic circuit that computes $f(z_1 \cdots z_{cL(n)})$ enough of the time to yield a contradiction.

We construct D in stages.

1. Let $1 \le j \le i-1$. Recall that $b_j = f_{cL(n)}(z_{u_1} \cdots z_{u_{cL(n)}})$ where $B_j = \{u_1 < \cdots < u_{cL(n)}\}$. For $1 \le k \le cL(n)$ if $u_k \notin \{1, \ldots, cL(n)\}$ then replace z_k by w_k. Since, for all $j \le i-1$, $|B_i \cap B_j| \le L(n)$, there are at most $L(n)$ u-variables.

   Construct a circuit D^j that computes this function. This is the circuit we will use for now. When we later restrict all of the w_k's, we will have b_j as a function of $L(n)$ variables. By Lemma 4.1 such a circuit has size $L(n)2^{L(n)}$. Since there are $i-1$ of them the size of all the restricted D^j's together is $\le (i-1)L(n)2^{L(n)} \le nL(n)2^{L(n)}$. (Note that if the B_i's were chosen less carefully we could have $\le ncL(n)2^{cL(n)}$. We comment on why this would not have sufficed our purposes after the proposition and after the corollary that follows it.)

2. We construct a circuit D on $z_1 \cdots z_{cL(n)} \cdot w_{cL(n)+1} \cdots w_{c^2L(n)}$ as follows. First the circuit computes $D_1, \ldots, D_{i-1}$. The outputs of these are fed into C_n. The size of C_n is $s_c(n)$ hence after we restrict the w_k's, the final circuit will have size $\le nL(n)2^{L(n)} + s_c(n)$.

We examine what D does. Let the input be $z_1 \cdots z_{cL(n)} \cdot w_{cL(n)+1} \cdots w_{c^2L(n)}$. Let $G_{c^2L(n)}(z_1 \cdots z_{cL(n)} w_{cL(n)+1} \cdots w_{c^2L(n)}) = b_1 \cdots b_n$. For $j \le i-1$,

$$D^j(z_1 \cdots z_{cL(n)} w_{cL(n)+1} \ldots, w_{c^2L(n)}) = b_j.$$

So D will compute $C_n(b_1 \cdots b_{i-1})$. By the definition of C_n, as we rewrote it above,

for over $\frac{1}{2} + \text{eps}(n)$ of the $(z, w) \in \{0,1\}^{cL(n)} \times \{0,1\}^{c^2L(n)-cL(n)}$

$$C_n(G_{c^2L(n)}(zw)[1:i-1]) = b_i \text{ where } G_{c^2L(n)}(zw) = b_1 \cdots b_n.$$

Since for all $(z,w) \in \{0,1\}^{cL(n)} \times \{0,1\}^{c^2L(n)-cL(n)}$ we have $b_i = f_{cL(n)}(z)$ we have
for over $\frac{1}{2} + \text{eps}(n)$ of the $(z,w) \in \{0,1\}^{cL(n)} \times \{0,1\}^{c^2L(n)-cL(n)}$,

$$C_n(G_{c^2L(n)}(zw)[1:i-1]) = f_{cL(n)}(z).$$

By Lemma 4.3 there exists $w_o \in \{0,1\}^{c^2L(n)-cL(n)}$ such that
for over $\frac{1}{2} + \text{eps}(n)$ of the $z \in \{0,1\}^{cL(n)}$

$$C_n(G_{c^2L(n)}(zw_o)[1:i-1]) = f_{cL(n)}(z).$$

The circuit that uses this value of w_o is of size $\le nL(n)2^{L(n)} + s_c(n)$. Since $s_c(n) << S(cL(n)) - nL(n)2^{L(n)}$ this circuit is of size $<< S(c(L(n)))$. this contradicts the hardness assumption on $f_{cL(n)}$. ▮

Note 5.2 We used a carefully chosen set of B_i's such that $|B_i \cap B_j| \le L(n)$. What is we had been less careful and only demanded that $|B_i \cap B_j| \le cL(n)$ which is trivial to obtain? Then the theorem would have been weakened in only one place- we would need $s_c(n) << S(cL(n)) - cnL(n)2^{cL(n)}$ instead of $s_c(n) << S(cL(n)) - nL(n)2^{L(n)}$. See the note after the next corollary to see why this matters.

Corollary 5.3 *Let $0 < \epsilon < 1$ and $c > \frac{2}{\epsilon}$. By setting $L(n) = \log n$, $S(n) = 2^{\epsilon n}$, $T(n) = 2^n$, $\text{eps}(n) = \frac{1}{4n}$, and $s_c(n) = n^{\epsilon c/2}$:*

$$\mathbf{HARD}_{2^{\epsilon n}, \frac{1}{4n}, 2^n} \Rightarrow (\forall c > \frac{2}{\epsilon})[\mathbf{NNBP}_{c^2 \log n, n^{\epsilon c}, \frac{1}{4n}, n^{c+1}+p_c(n)}].$$

Proof: The only condition to check is $s(n) << S(cL(n)) - nL(n)2^{L(n)}$. Note that
$S(c(L(n)) = 2^{\epsilon c \log n} = n^{\epsilon c}$, and
$nL(n)2^{L(n)} = n^2 \log n$
Since $c > \frac{2}{\epsilon}$ we have $\epsilon c > 2$ so
$S(cL(n)) - nL(n)2^{L(n)} = n^{\epsilon c} - n^2 \log n \ge \Omega(n^{\epsilon c})$. Since $s_c(n) = n^{\epsilon c/2}$ we have $s_c(n) << S(cL(n)) - nL(n)2^{L(n)}$. ▮

Note 5.4 In the proof of Proposition 5.1 we used a carefully chosen set of B_i's such that $|B_i \cap B_j| \le L(n)$. What if we had been less careful and only demanded that $|B_i \cap B_j| \le cL(n)$ (which is trivial to obtain)? Then the theorem would have been weakened in only one place- we would need $s_c(n) << S(cL(n)) - cnL(n)2^{cL(n)}$ instead of $s_c(n) << S(cL(n)) - nL(n)2^{L(n)}$. The function $s_c(n)$ in Corollary 5.3 would not have satisfied this condition. What function would have? We would still have $S(cL(n)) = n^{\epsilon c}$; however now we have $cnL(n)2^{cL(n)} = (cn \log n)n^c = cn^{c+1} \log n$. So $S(cL(n)) - cnL(n)2^{L(n)} = n^{\epsilon c} - cn^{c+1} \log n < 0$. Hence no $s_c(n)$ works.

6 NNBP $\Rightarrow$ PSRAND

Proposition 6.1 *Let $G : \{0,1\}^* \to \{0,1\}^*$ such that, for all n, G restricted to $\{0,1\}^{L(n)}$ maps to $\{0,1\}^n$. If G is not $(L(n), s(n), \mathrm{diff}(n))$-pseudorandom then G is $(L(n), s(n), \frac{\mathrm{diff}(n)}{n})$-next bit predictable. By taking the contrapositive this can be restated as*

$$\mathbf{NNBP}_{L(n),s(n),\mathrm{diff}(n)/n,t(n)} \Rightarrow \mathbf{PSRAND}_{L(n),s(n),\mathrm{diff}(n),t(n)}.$$

Proof: Since G is not $(L(n), s(n), \mathrm{diff}(n))$-pseudorandom there exists an $s(n)$-sized circuit C_n such that

$$\Pr_{x \in \{0,1\}^n}(C_n(x) = 1) - \Pr_{z \in \{0,1\}^{L(n)}}(C_n(G_{L(n)}(z)) = 1) > \mathrm{diff}(n).$$

For $0 \le i \le n$ let

$$p_i = \Pr_{z \in \{0,1\}^{L(n)}, w \in \{0,1\}^{n-i}, b_1 \cdots b_i = G_{L(n)}(z)[1:i]}(C_n(b_1 \cdots b_i w) = 1)$$

Note that

$p_0 = \Pr(C_n(w) = 1 : w \in \{0,1\}^n) = \Pr(C_n(x) = 1 : x \in \{0,1\}^n)$,

$p_n = \Pr(C_n(G_{L(n)}(z)) = 1 : z \in \{0,1\}^{L(n)})$.

Hence $p_0 - p_n > \mathrm{diff}(n)$.

Note that

$(p_0 - p_1) + (p_1 - p_2) + \cdots + (p_{n-1} - p_n) = p_0 - p_n = p_0 - p_n > \mathrm{diff}(n)$.

Hence there exists i such that

$p_{i-1} - p_i > \frac{\mathrm{diff}(n)}{n}$.

Though Experiment: Take a random $z_1 \cdots z_{L(n)} \in \{0,1\}^{L(n)}$ and let $C_n(z_1 \cdots z_{L(n)}) = b_1 \cdots b_n$. Now take a random $w_i \cdots w_n \in \{0,1\}^{n-i}$. The probability that $C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1$ is 'large'. BUT if you take a random $w_{i+1} \cdots w_n \in$

$\{0,1\}^{n-i-1}$ then the probability that $C_n(b_1 \cdots b_i w_{i+1} \cdots w_n) = 1$ is 'small'. This means that if you extend $b_1 \cdots b_{i-1}$ with the correct next bit then C_n is less likely to be 1 (on an extension of this extended string) then if you had extended by the incorrect next bit.

We now produce a circuit $D : \{0,1\}^n \to \{0,1\}$ based on the above intuition. The first $i-1$ bits will be inputs, the remaining $n-i+1$ bits will be chosen at random. We will then apply Lemma 4.4 to set values of the random bits to obtain a deterministic circuit.

1. Input($b_1 \cdots b_{i-1}$). (We only care when $b_1 \cdots b_{i-1} \in \mathbf{IMG}(G_{L(n)})$.
2. Pick $w_i \cdots w_n \in \{0,1\}^{n-i}$ at random.
3. Compute $b = C_n(b_1 \cdots b_{i-1} w_i \cdots w_n)$.
4. If $b = 1$ then output $1 - w_i$. If $b = 0$ then output w_i.

Picture the random variable: Take a random $z_1 \cdots z_{L(n)} \in \{0,1\}^{L(n)}$ and let $C_n(z_1 \cdots z_{L(n)}) = b_1 \cdots b_n$. Now take a random $w_i \cdots w_n \in \{0,1\}^{n-i}$. Run $D(b_1 \cdots b_{i-1})$ with random bits $w_i \cdots w_n$. Output the answer.

Let E be the probability that the output is b_i (that is, the probability that D outputs the correct next bit).

$$\begin{aligned}
\Pr(E) = \quad & \Pr(w_i = b_i \wedge C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 0) + \\
& \qquad \Pr(w_i = \overline{b_i} \wedge C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1) \\
\\
= \quad & \Pr(w_i = b_i)\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 0 | w_i = b_i) + \\
& \qquad \Pr(w_i = \overline{b_i})\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i})
\end{aligned}$$

We need to determine these various quantities.

1) $\Pr(w_i = b_i) = \Pr(w_i = \overline{b_i}) = \frac{1}{2}$.

2) $\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 0 | w_i = b_i)$. Note that

$$\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = b_i) = \Pr(C_n(b_1 \cdots b_{i-1} b_i w_{i+1} \cdots w_n) = 1) = p_i.$$

Hence

$$\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 0 | w_i = b_i) = 1 - p_i.$$

3) $\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i})$.

Note that

$$\begin{aligned}
p_{i-1} = {} & \Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1) \\
= {} & \Pr(w_i = b_i \wedge C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1) + \\
& \qquad \Pr(w_i = \overline{b_i} \wedge C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1) \\
= {} & \Pr(w_i = b_i)\Pr(C_n(b_1 \cdots b_{i-1} b_i w_{i+1} \cdots w_n) = 1 | w_i = b_i) + \\
& \qquad \Pr(w_i = \overline{b_i} \wedge C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1) \\
= {} & \tfrac{1}{2} p_i + \Pr(w_i = 1 = b_i)\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i}) \\
= {} & \tfrac{1}{2} p_i + \tfrac{1}{2}\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i})
\end{aligned}$$

We can use this equation to easily find
$\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i})$.
Hence

$$\begin{aligned}
\tfrac{1}{2} p_i + \tfrac{1}{2}\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i}) = {} & p_{i-1} \\
p_i + \Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i}) = {} & 2p_{i-1} \\
\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i}) = {} & 2p_{i-1} - p_i
\end{aligned}$$

Now that we have $\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i})$. We have all the pieces we need to calculate $\Pr(E)$.

Hence

$$\begin{aligned}
\Pr(E) = {} & \Pr(w_i = b_i)\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 0 | w_i = b_i) + \\
& \qquad \Pr(w_i = \overline{b_i})\Pr(C_n(b_1 \cdots b_{i-1} w_i \cdots w_n) = 1 | w_i = \overline{b_i}) \\
= {} & \tfrac{1}{2}(1 - p_i) + \tfrac{1}{2}(2p_{i-1} + p_i) \\
= {} & \tfrac{1}{2} + p_{i-1} - p_i \\
= {} & \tfrac{1}{2} + \tfrac{\text{diff}(n)}{n}
\end{aligned}$$

We apply Lemma 4.4 with D, i as above, $X = \{0,1\}^n$, $H_n = \mathbf{IMG}(G_n[1 : i-1])$, and

$$f(b_1 \cdots b_{i-1}) = \{b_i \mid b_1 \cdots b_i \in \mathbf{IMG}(G_n[1 : i]).$$

Hence we obtain fixed values of w such that, if they are fixed as such, then, for $\frac{1}{2} + \frac{\text{diff}(n)}{n}$ of the $z \in \{0,1\}^{L(n)}$ we have $D(G_n(z)[1 : i-1]) = b_i$ where $G_n(z) = b_1 \cdots b_n$. Hence G is $(L(n), s(n), \frac{\text{diff}(n)}{n})$-next-bit-predicable. ▮

Corollary 6.2 *Let $0 < \epsilon < 1$. Let $c > \frac{2}{\epsilon}$.*

1. *By setting $L(n) = c^2 \log n$, $s_c(n) = n^{\epsilon c/2}$, $t(n) = n^{c+1} + p_c(n)$, $\mathrm{diff}(n) = \frac{1}{4}$ we obtain*

   $$(\forall c \geq \frac{2}{\epsilon})[\mathbf{NNBP}_{c^2 \log n, n^{\epsilon c/2}, \frac{1}{4n}, n^{c+1}+p_c(n)} \Rightarrow (\forall c \geq \frac{2}{\epsilon})[\mathbf{PSRAND}_{c^2 \log n, n^{\epsilon c/2}, \frac{1}{4}, n^{c+1}+p_c(n)}].$$

   Hence

   $$[(\forall c \geq \frac{2}{\epsilon})[\mathbf{NNBP}_{c^2 \log n, n^{\epsilon c/2}, \frac{1}{4n}, n^{c+1}+p_c(n)}]] \Rightarrow [(\forall c \geq \frac{2}{\epsilon})[\mathbf{PSRAND}_{c^2 \log n, n^{\epsilon c/2}, \frac{1}{4}, n^{c+1}+p_c(n)}]].$$

2. *Combining the above with Corollary 5.3 we obtain*

   $$\mathbf{HARD}_{2^{\epsilon n}, \frac{1}{4n}, 2^n} \Rightarrow (\forall c \geq \frac{2}{\epsilon})[\mathbf{PSRAND}_{c^2 \log n, n^{\epsilon c}, \frac{1}{4}, n^{c+1}+p_c(n)}].$$

7 PSRAND $\Rightarrow$ CONTAINS

Proposition 7.1 *Assume there exists $G : \{0,1\}^* \rightarrow \{0,1\}^*$ such that the following occur.*

1. *$G \in \mathrm{DTIME}(t(n))$.*
2. *For all n, if G is restricted to $\{0,1\}^{L(n)}$ then it maps to $\{0,1\}^n$.*
3. *G is $(L(n), s(n), \mathrm{diff}(n))$-pseudorandom.*

Assume that $t' : \mathsf{N} \rightarrow \mathsf{N}$ is such that, for any constant c, $t'(cn) = O(t(n))$. Then $\mathrm{BPP}(t'(n), r(n), \mathrm{err}(n)) \subseteq \mathrm{DTIME}(u(n))$ where

1. *$2^{L(r(n))}(t(r(n)) + t'(n)) \leq u(n)$,*
2. *$(t'(n))^2 \leq s(n)$,*
3. *$\mathrm{diff}(n) + \mathrm{err}(n) \leq \frac{1}{2}$.*

This can be restated as

$$\mathbf{PSRAND}_{L(n), s(n), \mathrm{diff}(n), t(n)} \Rightarrow \mathbf{CONTAINS}_{t'(n), r(n), \mathrm{err}(n), u(n)}.$$

Proof: We will keep everything parameterized and note when the conditions give us what we need.

Let $A \in \text{BPP}(t'(n), r(n), \text{err}(n))$. Let G be an $(L(n), s(n), \text{diff}(n))$-pseudorandom generator such that $G_{L(n)} \in \text{DTIME}(t(n))$ and $G_{L(n)} : \{0,1\}^{L(n)} \rightarrow \{0,1\}^n$.

Let $G_{r(n)} : \{0,1\}^{L(r(n))} \rightarrow \{0,1\}^{r(n)}$. Note that $G_{r(n)}$ is in $\text{DTIME}(t(r(n)))$. (Also note that this is NOT our standard use of subscripts.)

The following algorithm shows that $A \in \text{DTIME}(u(n))$. The time bound will be easy; however, proving that it is correct will be interesting.

1. Input(x). $|x| = n$.
2. For every $z \in \{0,1\}^{L(r(n))}$ compute $y = G_{r(n)}(z)$ and run $M(x,y)$ (This takes $2^{L(r(n))}(t(r(n)) + t'(n))$ steps.) Keep a count of for how many y's we get a 0 and for how many y's we get a 1.
3. If the number of 0's is in the majority then output 0. If the number of 1's is in the majority then output 1.

This algorithm runs in time $2^{L(r(n))}(t(r(n))+t'(n)) \le u(n)$, so the algorithm runs in the desired time. Our concern is, is it correct? Assume, by way of contradiction, that it is incorrect. Assume on input $x_0 \in \{0,1\}^n$ the algorithm gives the wrong answer. We assume $x_0 \in A$ but the algorithm outputs 0 (the other case, where $x_0 \notin A$ but the algorithm outputs 1, is symmetric.) We have the following:

1. Since $x_0 \in A$, for $\le \text{err}(n)$ of the $y \in \{0,1\}^{r(n)}$ $M(x_0, y) = 0$. Hence

   $\Pr(M(x_0, y) = 0 : y \in \{0,1\}^{r(n)}) \le \text{err}(n)$.
2. Since the algorithm outputs 0, for $\ge \frac{1}{2}$ of the $z \in \{0,1\}^{L(r(n))}$, $M(x_0, G_{r(n)}(z)) = 0$. Hence

   $\Pr(M(x_0, G_{r(n)}(z)) = 0 : z \in \{0,1\}^{L(r(n))}) \ge \frac{1}{2}$.

Make a circuit out of $M(x_0, -)$ that will, on input $y \in \{0,1\}^{r(n)}$ compute $M(x_0, y)$. This circuit is of size $(t'(n + r(n)))^2 \le (t(2r(n))^2 = O((t'(r(n)))^2)$. (We are using $r(n) \le n$ and the condition on t'.) By the above

$$|\Pr(C(y) = 0 : y \in \{0,1\}^{r(n)}) - \Pr(C(G_{L(n)}(z)) = 0 : z \in \{0,1\}^{L(r(n))}| > \frac{1}{2} - \text{err}(n)$$

This will lead to a contradiction if

$(t'(r(n)))^2 \le s(r(n))$ (we achieve by $(t'(n))^2 \le s(n)$)
and
$\frac{1}{2} - \mathrm{err}(n) \ge \mathrm{diff}(n)$ (we achieve by $\mathrm{diff}(n) - \mathrm{err}(n) \le \frac{1}{2}$). ▮

Corollary 7.2 *Let $0 < \epsilon < 1$ and let $r(n)$ be any polynomial.*

1.

$$\textbf{PSRAND}_{L(n),s(n),\mathrm{diff}(n),t(n)} \Rightarrow \textbf{CONTAINS}_{\sqrt{s(n)},r(n),\frac{1}{2}-\mathrm{diff}(n),2^{L(r(n))}(t(r(n))+\sqrt{s(n)})}.$$

2. *Let $c \ge 2/\epsilon$. By setting $L(n) = c^2 \log n$, $s(n) = n^{\epsilon c/2}$, $\mathrm{eps}(n) = \frac{1}{4n}$, and $t(n) = n^{c+1} + p_c(n)$ we obtain the following.*

$$\textbf{PSRAND}_{c^2\log n, n^{\epsilon c/2}, \frac{1}{4}, n^{c+1}+p_c(n)} \Rightarrow \textbf{CONTAINS}_{n^{c\epsilon/4}, r(n), \frac{1}{4}, r(n)(r(n)^{c+1}+p_c(r(n)))+n^{\epsilon c/4}}.$$

3. *Combining the above with Corollary6.2.2 we obtain the following.*

$$\textbf{HARD}_{2^{\epsilon n}, \frac{1}{4n}, 2^n} \Rightarrow (\forall c \ge \frac{2}{\epsilon})[\textbf{CONTAINS}_{n^{c\epsilon/4}, r(n), \frac{1}{4}, r(n)(r(n)^{c+1}+p_c(r(n)))+n^{\epsilon c/4}}].$$

The last part of the last corollary is really what we want. We restate it without as much jargon.

Theorem 7.3 *Let $0 < \epsilon < 1$. Assume there exists a function $f \in \mathrm{DTIME}(2^n)$ such that f is $(2^{\epsilon n}, \frac{1}{4n})$-hard. Then for almost all k there exists a polynomial q_k such that $\mathrm{BPP}(n^k, n^k, \frac{1}{4}) \subseteq \mathrm{DTIME}(q_k(n))$. In particular $\mathrm{BPP} = \mathrm{P}$.*

Proof:

We are assuming the premise of Corollary 7.2.3. Hence we have the conclusion which we write as

$$(\forall c \ge \frac{2}{\epsilon})[\mathrm{BPP}(n^{c\epsilon/4}, r(n), \frac{1}{4}) \subseteq \mathrm{DTIME}(r(n)(r(n)^{c+1} + p_c(r(n)) + n^{\epsilon c/4}].$$

Let $A \in \mathrm{BPP}(n^k, n^k, \frac{1}{4})$. Let c be such that $c \ge \frac{2}{\epsilon}$ and $k \le c\epsilon/2$. Let $r(n) = n^k$. Then we have

$$\mathrm{BPP}(n^k, n^k, \frac{1}{4}) \subseteq \mathrm{BPP}(n^{c\epsilon/4}, r(n), \frac{1}{4}) \subseteq \mathrm{DTIME}(r(n)(r(n)^{c+1}+p_c(r(n))+n^{\epsilon c/4}].$$

Note that the time bound is polynomial in n, which is all we need. ▮

8 Notation Used Throughout the Paper

Notation 8.1 Throughout this paper the following hold.

1. $L(n) : \mathsf{N} \rightarrow \mathsf{N}$ (think log). c will be a constant. $cL(n)$ will be used alot.

2. $s(n), S(n) : \mathsf{N} \rightarrow \mathsf{N}$ (think poly, $2^{\epsilon n}$). Bounds on circuit size.

3. $r(n) : \mathsf{N} \rightarrow \mathsf{N}$ (think poly). We require $r(n) \geq n$. The random string that a BPP machine uses.

4. $t(n), T(n) : \mathsf{N} \rightarrow \mathsf{N}$ (think poly, 2^n). Run times.

5. $G : \{0,1\}^* \rightarrow \{0,1\}^*$. At different places we will also require that either (1) for all n $\{0,1\}^{L(n)}$ maps to $\{0,1\}^n$, or (2) for all n $\{0,1\}^{cL(n)}$ maps to $\{0,1\}^n$, or (3) for all n $\{0,1\}^{c^2L(n)}$ maps to $\{0,1\}^n$. We denote the subfunction that maps $\{0,1\}^m$ to $\{0,1\}^n$ by G_m. (m will be $L(n)$ or $cL(n)$ or $c^2L(n)$). A potential psuedorandom generator.

6. $f : \{0,1\}^* \rightarrow \{0,1\}$. We denote the subfunction that maps $\{0,1\}^n$ to $\{0,1\}$ by f_n. A "hard" function.

7. $\mathrm{err}(n) : \mathsf{N} \rightarrow \mathsf{Q} \cap (0, 1/2)$ (think $\frac{1}{4}$) An error, so the smaller it is the less chance of error.

8. $\mathrm{diff}(n) : \mathsf{N} \rightarrow \mathsf{Q} \cap (0, 1/2)$ (think $\frac{1}{\mathrm{poly}}$). $\mathrm{diff}(n)$ is decreasing. How much two distributions differ. The smaller it is, the less they differ.

9. $\mathrm{eps}(n) : \mathsf{N} \rightarrow \mathsf{Q} \cap (0, 1/2)$ (think $\frac{1}{\mathrm{poly}}$). $\mathrm{eps}(n)$ is decreasing. How much more than $\frac{1}{2}$ of the elements of some domain a function is computed correctly. The larger $\mathrm{eps}(n)$ the large the domain we can compute the function on.

References

[1] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49, 1994. Prior version in FOCS88.